**EXHIBIT D-1**
Defendant's Preliminary Invalidity Contentions
*Orckit Corporation v. Cisco Systems, Inc.*, 2:22-cv-00276-JRG-RSP

---

**Chart for U.S. Patent 10,652,111 ("the '111 Patent")**
**U.S. Patent Publication No. 2012/0300615 to Kempf et al. ("Kempf")**

As shown in the chart below, all Asserted Claims of the '111 Patent are invalid under (1) AIA-35 U.S.C. § 102 (a) because Kempf meets each element of those claims, and/or (2) 35 U.S.C. § 103 because Kempf renders those claims obvious either alone, or in combination with the knowledge of a person having ordinary skill in the art, and in further combination with the references specifically identified below and in the following claim chart and/or one or more references identified in Defendant's Preliminary Invalidity Contentions. The following quotations and diagrams come from Kempf titled "Implementing EPC In A Cloud Computer With OpenFlow Data Plane", which was filed on June 28, 2012, and published on November 29, 2012.

Motivations to combine the disclosures in Kempf with disclosures in other publications known in the art, as explained in this chart, include at least the similarity in subject matter between the references to the extent they concern methods relating to routing certain network traffic to entities for further analysis and inspection. Insofar as the references cite other patents or publications, or suggest additional changes, one of ordinary skill in the art would look beyond a single reference to other references in the field.

These invalidity contentions are based on Defendant's present understanding of the Asserted Claims, and Orckit's apparent construction of the claims in its November 3, 2022 Disclosure of Asserted Claims and Infringement Contentions Pursuant to P.R. 3-1, and Orckit's January 19, 2023 First Amended Disclosure of Asserted Claims and Infringement Contentions Pursuant to P.R. 3-1 (Orckit's "Infringement Disclosures"), which is deficient at least insofar as it fails to cite any documents or identify accused structures, acts, or materials in the Accused Products with particularity. Defendant does not agree with Orckit's application of the claims, or that the claims satisfy the requirements of 35 U.S.C. § 112. Defendant's contentions herein are not, and should in no way be seen as, admissions or adoptions as to any particular claim scope or construction, or as any admission that any particular element is met by any accused product in any particular way. Defendant objects to any attempt to imply claim construction from this chart. Defendant's prior art invalidity contentions are made in a variety of alternatives and do not represent Defendant's agreement or view as to the meaning, definiteness, written description support for, or enablement of any claim contained therein.



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1 of 1100

The following contentions are subject to revision and amendment pursuant to Federal Rule of Civil Procedure 26(e), the Local Rules, and the Orders of record in this matter subject to further investigation and discovery regarding the prior art and the Court's construction of the claims at issue.

| No. | '111 Patent Claim 1 | Kempf |
|---|---|---|
| 1[preamble] | A method for use with a packet network including a network node for transporting packets between first and second entities under control of a controller that is external to the network node, the method comprising: | Kempf discloses a method for use with a packet network including a network node for transporting packets between first and second entities under control of a controller that is external to the network node, the method comprising.<br><br>For example, Kempf discloses a method in which a network element such as a router, switch, or bridge communicatively interconnects other elements of a network for data packet transport. Kempf further discloses a method in which the network element is controlled by an external OpenFlow controller. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met.<br><br>Kempf at Abstract ("A method implements a control plane of an evolved packet core (EPC) of a long term evolution (LTE) network in a cloud computing system. A cloud manager monitors resource utili-zation of each control plane module and the control plane traffic handled by each control plane module. The cloud man-ager detects a threshold level of resource utilization or traffic load for one of the plurality of control plane modules of the EPC. A new control plane module is initialized as a separate virtual machine by the cloud manager in response to detecting the threshold level. The new control plane module signals the plurality of network elements in the data plane to establish flow rules and actions to establish differential routing of flows in the data plane using the control protocol, wherein flow matches are encoded using an extensible match structure in which the flow match is encoded as a type-length-value (TLV).")<br><br>Kempf at [0004] ("The GPRS tunneling protocol (GTP) is an important communication protocol utilized within the GPRS core net-work. GTP enables end user devices ( e.g., cellular phones) in a GSM network to move from place to place while continuing to connect to the Internet. The end user devices are connected to other devices through a gateway GPRS support node (GGSN). The GGSN tracks the end user device's data from the end user device's serving GPRS support node (GGSN) that is handling the session originating from the end user device.") |

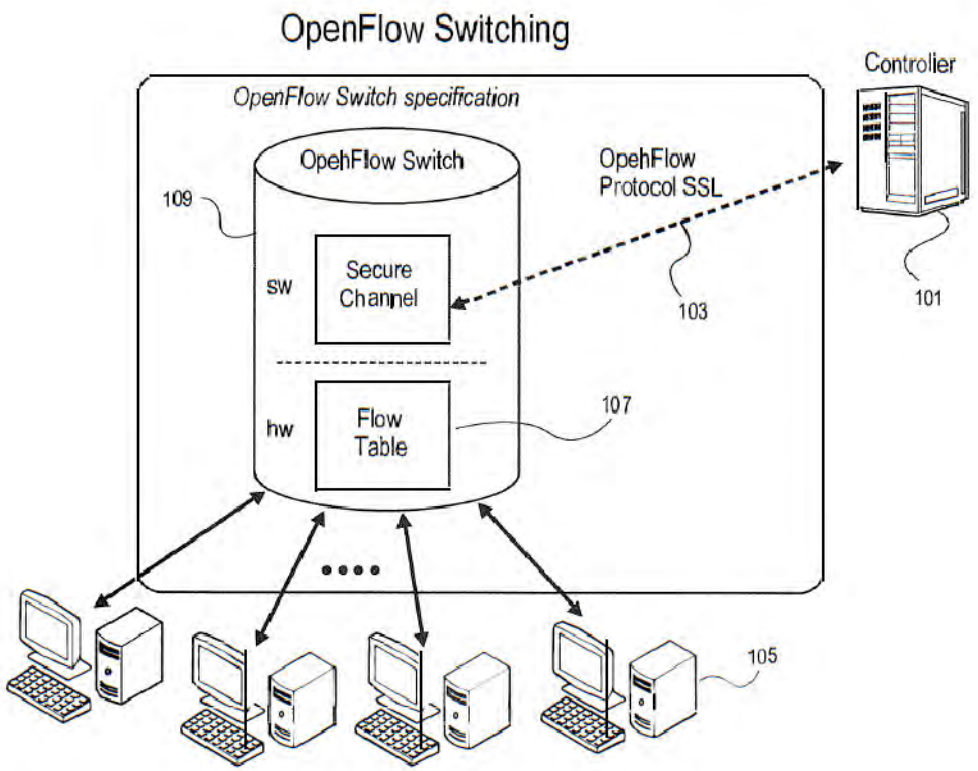


~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 2 of 1100

| No. | '111 Patent Claim 1 | Kempf |
|---|---|---|
| | | Kempf at [0033] ("As used herein, a network element (e.g., a router, switch, bridge, etc.) is a piece of networking equipment, including hardware and software, that communicatively interconnects other equipment on the network (e.g., other network elements, end stations, etc.). Some network elements are "multiple services network elements" that provide sup-port for multiple networking functions (e.g., routing, bridg-ing, switching, Layer 2 aggregation, session border control, multicasting, and/or subscriber management), and/or provide support for multiple application services (e.g., data, voice, and video). Subscriber end stations ( e.g., servers, worksta-tions, laptops, palm tops, mobile phones, smart phones, mul-timedia phones, Voice Over Internet Protocol (VOIP) phones, portable media players, GPS units, gaming systems, set-top boxes (STBs), etc.) access content/services provided over the Internet and/or content/services provided on virtual private networks (VPN s) overlaid on the Internet. The content and/or services are typically provided by one or more end stations ( e.g., server end stations) belonging to a service or content provider or end stations participating in a peer to peer service, and may include public web pages (free content, store fronts, search services, etc.), private web pages (e.g., username/pass-word accessed web pages providing email services, etc.), corporate networks over VPNs, IPTV, etc. Typically, sub-scriber end stations are coupled (e.g., through customer premise equipment coupled to an access network (wired or wirelessly)) to edge network elements, which are coupled (e.g., through one or more core network elements to other edge network elements) to other end stations (e.g., server end stations).")<br><br>Kempf at [0044] ("FIG. 1 is a diagram of one embodiment of an example network with an OpenFlow switch, conforming to the OpenFlow 1.0 specification. The OpenFlow 1.0 protocol enables a controller 101 to connect to an OpenFlow 1.0 enabled switch 109 using a secure channel 103 and control a single forwarding table 107 in the switch 109. The controller 101 is an external software component executed by a remote computing device that enables a user to configure the Open-Flow 1.0 switch 109. The secure channel 103 can be provided by any type of network including a local area network (LAN) or a wide area network (WAN), such as the Internet.")<br><br>Kempf at Figure 1 |

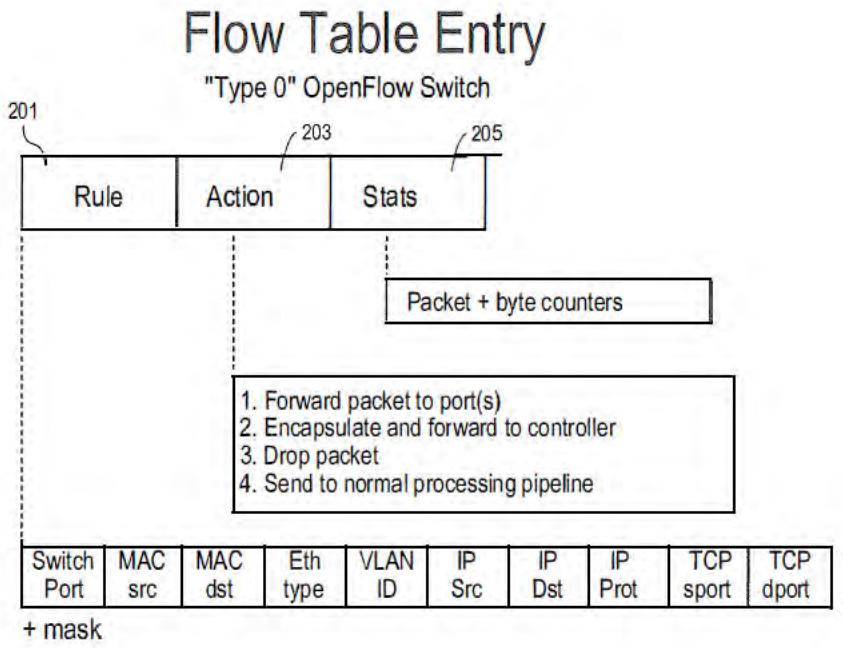


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 3 of 1100

| No. | '111 Patent Claim 1 | Kempf |
|---|---|---|
| | |  <br>FIG. 1<br>Kempf at Figure 2 |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 4 of 1100

| No. | '111 Patent Claim 1 | Kempf |
|---|---|---|
| | |  <br>FIG. 2 |
| 1[a] | sending, by the controller to the network node over the packet network, an instruction and a packet-applicable criterion; | Kempf discloses sending, by the controller to the network node over the packet network, an instruction and a packet-applicable criterion.<br><br>For example, Kempf discloses sending by the OpenFlow controller to the network element a rule defining matches for fields in packet headers. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met.<br><br>Kempf at [0044] ("FIG. 1 is a diagram of one embodiment of an example network with an OpenFlow switch, conforming to the OpenFlow 1.0 specification. The OpenFlow 1.0 |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 5 of 1100

| No. | '111 Patent Claim 1 | Kempf |
|---|---|---|
| | | protocol enables a controller 101 to connect to an OpenFlow 1.0 enabled switch 109 using a secure channel 103 and control a single forwarding table 107 in the switch 109. The controller 101 is an external software component executed by a remote computing device that enables a user to configure the Open-Flow 1.0 switch 109. The secure channel 103 can be provided by any type of network including a local area network (LAN) or a wide area network (WAN), such as the Internet.")<br><br>Kempf at [0045] ("FIG. 2 is a diagram illustrating one embodiment of the contents of a flow table entry. The forwarding table 107 is populated with entries consisting of a rule 201 defining matches for fields in packet headers; an action 203 associated to the flow match; and a collection of statistics 205 on the flow. When an incoming packet is received a lookup for a matching rule is made in the flow table 107. If the incoming packet matches a particular rule, the associated action defined in that flow table entry is performed on the packet.")<br><br>Kempf at [0046] ("A rule 201 contains key fields from several headers in the protocol stack, for example source and destination Ethernet MAC addresses, source and destination IP addresses, IP protocol type number, incoming and outgoing TCP or UDP port numbers. To define a flow, all the available matching fields may be used. But it is also possible to restrict the matching rule to a subset of the available fields by using wildcards for the unwanted fields.")<br><br>Kempf at [0047] ("The actions that are defined by the specification of OpenFlow 1.0 are Drop, which drops the matching packets; Forward, which forwards the packet to one or all outgoing ports, the incoming physical port itself, the controller via the secure channel, or the local networking stack (if it exists). OpenFlow 1.0 protocol data units (PDU s) are defined with a set of structures specified using the C programming language. Some of the more commonly used messages are: report switch configuration message; modify state messages (in-cluding a modify flow entry message and port modification message); read state messages, where while the system is running, the datapath may be queried about its current state using this message; and send packet message, which is used when the controller wishes to send a packet out through the datapath.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 6 of 1100

| No. | '111 Patent Claim 1 | Kempf |
|---|---|---|
| | | Kempf at [0050] ("FIG. 4 illustrates one embodiment of the processing of packets through an OpenFlow 1.1 switched packet pro-cessing pipeline. A received packet is compared against each of the flow tables 401. After each flow table match, the actions are accumulated into an action set. If processing requires matching against another flow table, the actions in the matched rule include an action directing processing to the next table in the pipeline. Absent the inclusion of an action in the set to execute all accumulated actions immediately, the actions are executed at the end 403 of the packet processing pipeline. An action allows the writing of data to a metadata register, which is carried along in the packet processing pipe-line like the packet header.")<br><br>Kempf at [0051] ("FIG. 5 is a flowchart of one embodiment of the OpenFlow 1.1 rule matching process. OpenFlow 1.1 contains support for packet tagging. OpenFlow 1.1 allows matching based on header fields and multi-protocol label switching (MPLS) labels. One virtual LAN (VLAN) label and one MPLS label can be matched per table. The rule matching process is initiated with the arrival of a packet to be processed (Block 501 ). Starting at the first table 0 a lookup is performed to determine a match with the received packet (Block 503). If there is no match in this table, then one of a set of default actions is taken (i.e., send packet to controller, drop the packet or continue to next table) (Block 509). If there is a match, then an update to the action set is made along with counters, packet or match set fields and meta data (Block 505). A check is made to determine the next table to process, which can be the next table sequentially or one specified by an action of a matching rule (Block 507). Once all of the tables have been processed, then the resulting action set is executed (Block 511). FIG. 6 is a diagram of the fields, which a matching process can utilize for identifying rules to apply to a packet.")<br><br>Kempf at [0074] ("The operation of the EPC cloud computer system as follows. The UE 1317, E-NodeB 1317, S-GW-C 1307, and P-GW-C signal 1307 to the MME, PCRF, and HSS 1307 using the standard EPC protocols, to establish, modify, and delete bearers and GTP tunnels. This signaling triggers pro-cedure calls with the OpenFlow controller to modify the routing in the EPC as requested. The OpenFlow controller configures the standard OpenFlow switches, the Openflow S-GW-D 1315, and P-GW-D 1311 with flow rules and actions to enable the routing requested by the control plane entities. Details of this configuration are described in further detail herein below.") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 7 of 1100

| No. | '111 Patent Claim 1 | Kempf |
|---|---|---|
| | | Kempf at [0079] ("FIG. 16 is a diagram of one embodiment of a process for EPC peering and differential routing for specialized ser-vice treatment. The OpenFlow signaling, indicated by the solid lines and arrows 1601, sets up flow rules and actions on the switches and gateways within the EPC for differential routing. These flow rules direct GTP flows to particular loca-tions. In this example, the operator in this case peers its EPC with two other fixed operators. Routing through each peering point is handled by the respective P-GW-Dl and P-GW-D2 1603A, B. The dashed lines and arrows 1605 show traffic from a UE 1607 that needs to be routed to another peering operator. The flow rules and actions to distinguish which peering point the traffic should traverse are installed in the OpenFlow switches 1609 and gateways 1603A, B by the OpenFlow controller 1611. The OpenFlow controller 1611 calculates these flow rules and actions based on the routing tables it maintains for outside traffic, and the source and destination of the packets, as well as by any specialized for-warding treatment required for DSCP marked packets.")<br><br>Kempf at [0080] ("The long dash and dotted lines and arrows 1615 shows a example of a UE 1617 that is obtaining content from an external source. The content is originally not formulated for the UE's 1617 screen, so the OpenFlow controller 1611 has installed flow rules and actions on the P-GW-Dl 1603B, S-GW-D 1619 and the OpenFlow switches 1609 to route the flow through a transcoding application 1621 in the cloud computing facility. The transcoding application 1621 refor-mats the content so that it will fit on the UE's 1617 screen. A PCRF requests the specialized treatment at the time the UE sets up its session with the external content source via the IP Multimedia Subsystem (IMS) or another signaling protocol.")<br><br>Kempf at [0081] ("In one embodiment, OpenFlow is modified to pro-vide rules for GTP TEID Routing. FIG. 17 is a diagram of one embodiment of the OpenFlow flow table modification for GTP TEID routing. An OpenFlow switch that supports TEID routing matches on the 2 byte (16 bit) collection of header fields and the 4 byte (32 bit) GTP TEID, in addition to other OpenFlow header fields, in at least one flow table ( e.g., the first flow table). The GTP TEID flag can be wildcarded (i.e. matches are "don't care"). In one embodiment, the EPC pro-tocols do not assign any meaning to TEIDs other than as an endpoint identifier for tunnels, like ports in standard UDP/ TCP transport protocols. In other embodiments, the TEIDs can have a correlated meaning or semantics. The GTP header flags field can also be wildcarded, this can be partially matched by combining the following |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 8 of 1100

| No. | '111 Patent Claim 1 | Kempf |
|---|---|---|
| | | bitmasks: 0xFF00- Match the Message Type field; 0xe0-Match the Version field; 0xl0-Match the PT field; 0x04-Match the E field; 0x02- Match the S field; and 0x0l-Match the PN field.")<br><br>Kempf at [0085] ("The OpenFlow controller instantiates a virtual port for each physical port that may transmit or receive packets routed through a GTP tunnel, prior to installing any rules in the switch for GTP TEID routing.)<br><br>Kempf at [0089] ("n one embodiment, the system implements a GTP fast path encapsulation virtual port. When requested by the S-GW-C and P-GW-C control plane software running in the cloud computing system, the OpenFlow controller programs the gateway switch to install rules, actions, and TEID hash table entries for routing packets into GTP tunnels via a fast path GTP encapsulation virtual port. The rules match the packet filter for the input side of GTP tunnel's bearer. Typi-cally this will be a 4 tuple of: IP source address; IP destination address; UDP/TCP/SCTP source port; and UDP/TCP/SCTP destination port. The IP source address and destination address are typically the addresses for user data plane traffic, i.e. a UE or Internet service with which a UE is transacting, and similarly with the port numbers. For a rule matching the GTP-U tunnel input side, the associated instructions and are the following:<br><br>Write-Metadata ( GTP-TEID, OxFFFFFFFF)<br>Apply-Actions (Set-Output-Port GTP-Encap-VP)")<br><br>Kempf at [0092] ("In one embodiment, the system implements a GTP fast path decapsulation virtual port. When requested by the S-GW and P-GW control plane software running in the cloud computing system, the gateway switch installs rules and actions for routing GTP encapsulated packets out of GTP tunnels. The rules match the GTP header flags and the GTP TEID for the packet, in the modified OpenFlow flow table shown in FIG. 17 as follows: the IP destination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); the UDP destination port is the GTP-U destination port (2152); and the header fields and message type field is wildcarded with the flag 0XFFF0 and the upper two bytes of the field match the G-PDU message type (255) while the lower two bytes match 0x30, i.e. the packet is a GTP packet not a GTP' packet and the version number is 1.") |


~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 9 of 1100

| No. | '111 Patent Claim 1 | Kempf |
|---|---|---|
| | | Kempf at [0094] ("In one embodiment, the system implements han-dling of GTP-U control packets. The OpenFlow controller programs the gateway switch flow tables with 5 rules for each gateway switch IP address used for GTP traffic. These rules contain specified values for the following fields: the IP des-tination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); the UDP destination port is the GTP-U destination port (2152); the GTP header flags and message type field is wildcarded with 0xFFF0; the value of the header flags field is 0x30, i.e. the version number is 1 and the PT field is 1; and the value of the message type field is one of 1 (Echo Request), 2 (Echo Response), 26 (Error Indication), 31 (Support for Extension Headers Notification), or 254 (End Marker).")<br><br>Kempf at [0097] ("In one embodiment, the system implements han-dling of G-PDU packets with extension headers, sequence numbers, and N-PDU numbers. G-PDU packets with exten-sion headers, sequence numbers, and N-PDU numbers need to be forwarded to the local switch software control plane for processing. The OpenFlow controller programs 3 rules for this purpose. They have the following common header fields: the IP destination address is an IP address on which the gateway is expecting GTP traffic; and the IP protocol type is UDP (17); the UDP destination port is the GTP-U destination port (2152).")<br><br>Kempf at [0099] ("The instruction for these rules is the following:<br><br>Apply-Actions (Set-Output-Port LOCAL_GTP _U_DECAP)")<br><br>Kempf at [0104] ("In one embodiment, the system implements han-dling of GTP-C and GTP' control packets. Any GTP-C and GTP' control packets that are directed to IP addresses on a gateway switch are in error. These packets need to be handled by the S-GW-C, P-GW-C, and GTP' protocol entities in the cloud computing system, not the S-GW-D and P-GW-D enti-ties in the switches. To catch such packets, the OpenFlow controller must program the switch with the following two rules: the IP destination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); for one rule, the UDP destination port is the GTP-U destination port (2152), for the other, the UDP destination port is the GTP-C destination port (2123); the GTP header flags and message type fields are wildcarded.") |


~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 10 of 1100

| No. | '111 Patent Claim 1 | Kempf |
|---|---|---|
| | | Kempf at [0108] ("A GTP-extended Openflow switch contains at least one flow table that handles rules matching the GTP header fields as in FIG. 17. The Openflow controller programs the GTP header field rules in addition to the other fields to per-form GTP routing and adds appropriate actions if the rule is matched. For example, the following rule matches a GTP-C control packet directed to a control plane entity (MME, S-GW-C, P-GW-C) in the cloud computing system, which is not in the control plane VLAN: the VLAN tag is not set to the control plane VLAN, the destination IP address field is set to the IP address of the targeted control plane entity, the IP protocol type is UDP (17), the UDP destination port is the GTP-C destination port (2123), the GTP header flags and message type is wildcarded with 0xF0 and the matched ver-sion and protocol type fields are 2 and 1, indicating that the packet is a GTPv2 control plane packet and not GTP'.") |
| 1[b] | receiving, by the network node from the controller, the instruction and the criterion; | Kempf discloses receiving, by the network node from the controller, the instruction and the criterion.<br><br>*See supra at* 1[a]. |
| 1[c] | receiving, by the network node from the first entity over the packet network, a packet addressed to the second entity; | Kempf discloses receiving, by the network node from the first entity over the packet network, a packet addressed to the second entity.<br><br>For example, Kempf discloses communication between electronic devices in which data packets are sent from one electronic device to another destination device.<br><br>Kempf at [0003] ("The general packet radios system (GPRS) is a sys-tem that is used for transmitting Internet Protocol packets between user devices such as cellular phones and the Internet. The GPRS system includes the GPRS core network, which is an integrated part of the global system for mobile communi-cation (GSM). These systems are widely utilized by cellular phone network providers to enable cellular phone services over large areas.")<br><br>Kempf at [0004] ("The GPRS tunneling protocol (GTP) is an important communication protocol utilized within the GPRS core net-work. GTP enables end user devices ( e.g., cellular phones) in a GSM network to move from place to place while continuing to connect to the Internet. The end user devices are connected to other devices through a gateway |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 11 of 1100

| No. | '111 Patent Claim 1 | Kempf |
|---|---|---|
| | | GPRS support node (GGSN). The GGSN tracks the end user device's data from the end user device's serving GPRS support node (GGSN) that is handling the session originating from the end user device.")<br><br>Kempf at [0032] ("The techniques shown in the figures can be imple-mented using code and data stored and executed on one or more electronic devices ( e.g., an end station, a network ele-ment, etc.). Such electronic devices store and communicate (internally and/or with other electronic devices over a net-work) code and data using non-transitory machine-readable or computer-readable media, such as non-transitory machine-readable or computer-readable storage media ( e.g., magnetic disks; optical disks; random access memory; read only memory; flash memory devices; and phase-change memory). In addition, such electronic devices typically include a set of one or more processors coupled to one or more other compo-nents, such as one or more storage devices, user input/output devices (e.g., a keyboard, a touch screen, and/or a display), and network connections. The coupling of the set of proces-sors and other components is typically through one or more busses and bridges (also termed as bus controllers). The stor-age devices represent one or more non-transitory machine-readable or computer-readable storage media and non-tran-sitory machine-readable or computer-readable communication media. Thus, the storage device of a given electronic device typically stores code and/or data for execu-tion on the set of one or more processors of that electronic device. Of course, one or more parts of an embodiment of the invention may be implemented using different combinations of software, firmware, and/or hardware.")<br><br>Kempf at [0033] ("As used herein, a network element (e.g., a router, switch, bridge, etc.) is a piece of networking equipment, including hardware and software, that communicatively interconnects other equipment on the network (e.g., other network elements, end stations, etc.). Some network elements are "multiple services network elements" that provide sup-port for multiple networking functions (e.g., routing, bridg-ing, switching, Layer 2 aggregation, session border control, multicasting, and/or subscriber management), and/or provide support for multiple application services (e.g., data, voice, and video). Subscriber end stations ( e.g., servers, worksta-tions, laptops, palm tops, mobile phones, smart phones, mul-timedia phones, Voice Over Internet Protocol (VOIP) phones, portable media players, GPS units, gaming systems, set-top boxes (STBs), etc.) access content/services provided over the Internet and/or content/services provided on virtual private networks (VPNs) |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 12 of 1100

| No. | '111 Patent Claim 1 | Kempf |
|---|---|---|
| | | overlaid on the Internet. The content and/or services are typically provided by one or more end stations ( e.g., server end stations) belonging to a service or content provider or end stations participating in a peer to peer service, and may include public web pages (free content, store fronts, search services, etc.), private web pages ( e.g., username/pass-word accessed web pages providing email services, etc.), corporate networks over VPNs, IPTV, etc. Typically, sub-scriber end stations are coupled ( e.g., through customer premise equipment coupled to an access network (wired or wirelessly)) to edge network elements, which are coupled (e.g., through one or more core network elements to other edge network elements) to other end stations (e.g., server end stations).")<br><br>Kempf at [0040] ("The standard EPC architecture assumes a standard routed IP network for transport on top of which the mobile network entities and protocols are implemented. The enhanced EPC architecture described herein is instead at the level ofIP routing and media access control (MAC) switch-ing. Instead of using L2 routing and L3 internal gateway protocols to distribute IP routing and managing Ethernet and IP routing as a collection of distributed control entities, L2 and L3 routing management is centralized in a cloud facility and the routing is controlled from the cloud facility using the OpenFlow protocol. As used herein, the "OpenFlow proto-col" refers to the OpenFlow network protocol and switching specification defined in the OpenFlow Switch Specification at www.openflowswitch.org a web site hosted by Stanford Uni-versity. As used herein, an "OpenFlow switch" refers to a network element implementing the OpenFlow protocol.")<br><br>Kempf at [0079] ("FIG. 16 is a diagram of one embodiment of a process for EPC peering and differential routing for specialized ser-vice treatment. The OpenFlow signaling, indicated by the solid lines and arrows 1601, sets up flow rules and actions on the switches and gateways within the EPC for differential routing. These flow rules direct GTP flows to particular loca-tions. In this example, the operator in this case peers its EPC with two other fixed operators. Routing through each peering point is handled by the respective P-GW-Dl and P-GW-D2 1603A, B. The dashed lines and arrows 1605 show traffic from a UE 1607 that needs to be routed to another peering operator. The flow rules and actions to distinguish which peering point the traffic should traverse are installed in the OpenFlow switches 1609 and gateways 1603A, B by the OpenFlow controller 1611. The OpenFlow controller 1611 calculates these flow rules and actions based on the routing tables it maintains for outside |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 13 of 1100

| No. | '111 Patent Claim 1 | Kempf |
|---|---|---|
| | | traffic, and the source and destination of the packets, as well as by any specialized for-warding treatment required for DSCP marked packets.") |
| 1[d] | checking, by the network node, if the packet satisfies the criterion; | Kempf discloses checking, by the network node, if the packet satisfies the criterion.<br><br>For example, Kempf discloses determining by the network element if the packet header field matches an associated action in the flow table.<br><br>Kempf at [0044] ("FIG. 1 is a diagram of one embodiment of an example network with an OpenFlow switch, conforming to the OpenFlow 1.0 specification. The OpenFlow 1.0 protocol enables a controller 101 to connect to an OpenFlow 1.0 enabled switch 109 using a secure channel 103 and control a single forwarding table 107 in the switch 109. The controller 101 is an external software component executed by a remote computing device that enables a user to configure the Open-Flow 1.0 switch 109. The secure channel 103 can be provided by any type of network including a local area network (LAN) or a wide area network (WAN), such as the Internet.")<br><br>Kempf at [0045] ("FIG. 2 is a diagram illustrating one embodiment of the contents of a flow table entry. The forwarding table 107 is populated with entries consisting of a rule 201 defining matches for fields in packet headers; an action 203 associated to the flow match; and a collection of statistics 205 on the flow. When an incoming packet is received a lookup for a matching rule is made in the flow table 107. If the incoming packet matches a particular rule, the associated action defined in that flow table entry is performed on the packet.")<br><br>Kempf at [0046] ("A rule 201 contains key fields from several headers in the protocol stack, for example source and destination Ethernet MAC addresses, source and destination IP addresses, IP protocol type number, incoming and outgoing TCP or UDP port numbers. To define a flow, all the available matching fields may be used. But it is also possible to restrict the matching rule to a subset of the available fields by using wildcards for the unwanted fields.")<br><br>Kempf at [0047] ("The actions that are defined by the specification of OpenFlow 1.0 are Drop, which drops the matching packets; Forward, which forwards the packet to one or all |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 14 of 1100

| No. | '111 Patent Claim 1 | Kempf |
|---|---|---|
| | | outgoing ports, the incoming physical port itself, the controller via the secure channel, or the local networking stack (if it exists). OpenFlow 1.0 protocol data units (PDU s) are defined with a set of structures specified using the C programming language. Some of the more commonly used messages are: report switch configuration message; modify state messages (in-cluding a modify flow entry message and port modification message); read state messages, where while the system is running, the datapath may be queried about its current state using this message; and send packet message, which is used when the controller wishes to send a packet out through the datapath.")<br><br>Kempf at [0050] ("FIG. 4 illustrates one embodiment of the processing of packets through an OpenFlow 1.1 switched packet pro-cessing pipeline. A received packet is compared against each of the flow tables 401. After each flow table match, the actions are accumulated into an action set. If processing requires matching against another flow table, the actions in the matched rule include an action directing processing to the next table in the pipeline. Absent the inclusion of an action in the set to execute all accumulated actions immediately, the actions are executed at the end 403 of the packet processing pipeline. An action allows the writing of data to a metadata register, which is carried along in the packet processing pipe-line like the packet header.")<br><br>Kempf at [0051] ("FIG. 5 is a flowchart of one embodiment of the OpenFlow 1.1 rule matching process. OpenFlow 1.1 contains support for packet tagging. OpenFlow 1.1 allows matching based on header fields and multi-protocol label switching (MPLS) labels. One virtual LAN (VLAN) label and one MPLS label can be matched per table. The rule matching process is initiated with the arrival of a packet to be processed (Block 501 ). Starting at the first table 0 a lookup is performed to determine a match with the received packet (Block 503). If there is no match in this table, then one of a set of default actions is taken (i.e., send packet to controller, drop the packet or continue to next table) (Block 509). If there is a match, then an update to the action set is made along with counters, packet or match set fields and meta data (Block 505). A check is made to determine the next table to process, which can be the next table sequentially or one specified by an action of a matching rule (Block 507). Once all of the tables have been processed, then the resulting action set is executed (Block 511). FIG. 6 is a diagram of the fields, which a matching process can utilize for identifying rules to apply to a packet.") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 15 of 1100

| No. | '111 Patent Claim 1 | Kempf |
|---|---|---|
| | | |
| 1[e] | responsive to the packet not satisfying the criterion, sending, by the network node over the packet network, the packet to the second entity; and | Kempf discloses responsive to the packet not satisfying the criterion, sending, by the network node over the packet network, the packet to the second entity.<br><br>For example, Kempf discloses sending the packet from the network element to the destination device in response to the packet not matching the action in the flow table.<br><br>Kempf at [0044] ("FIG. 1 is a diagram of one embodiment of an example network with an OpenFlow switch, conforming to the OpenFlow 1.0 specification. The OpenFlow 1.0 protocol enables a controller 101 to connect to an OpenFlow 1.0 enabled switch 109 using a secure channel 103 and control a single forwarding table 107 in the switch 109. The controller 101 is an external software component executed by a remote computing device that enables a user to configure the Open-Flow 1.0 switch 109. The secure channel 103 can be provided by any type of network including a local area network (LAN) or a wide area network (WAN), such as the Internet.")<br><br>Kempf at [0045] ("FIG. 2 is a diagram illustrating one embodiment of the contents of a flow table entry. The forwarding table 107 is populated with entries consisting of a rule 201 defining matches for fields in packet headers; an action 203 associated to the flow match; and a collection of statistics 205 on the flow. When an incoming packet is received a lookup for a matching rule is made in the flow table 107. If the incoming packet matches a particular rule, the associated action defined in that flow table entry is performed on the packet.")<br><br>Kempf at [0046] ("A rule 201 contains key fields from several headers in the protocol stack, for example source and destination Ethernet MAC addresses, source and destination IP addresses, IP protocol type number, incoming and outgoing TCP or UDP port numbers. To define a flow, all the available matching fields may be used. But it is also possible to restrict the matching rule to a subset of the available fields by using wildcards for the unwanted fields.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 16 of 1100

| No. | '111 Patent Claim 1 | Kempf |
|---|---|---|
| | | Kempf at [0047] ("The actions that are defined by the specification of OpenFlow 1.0 are Drop, which drops the matching packets; Forward, which forwards the packet to one or all outgoing ports, the incoming physical port itself, the controller via the secure channel, or the local networking stack (if it exists). OpenFlow 1.0 protocol data units (PDU s) are defined with a set of structures specified using the C programming language. Some of the more commonly used messages are: report switch configuration message; modify state messages (in-cluding a modify flow entry message and port modification message); read state messages, where while the system is running, the datapath may be queried about its current state using this message; and send packet message, which is used when the controller wishes to send a packet out through the datapath.")<br><br>Kempf at [0050] ("FIG. 4 illustrates one embodiment of the processing of packets through an OpenFlow 1.1 switched packet pro-cessing pipeline. A received packet is compared against each of the flow tables 401. After each flow table match, the actions are accumulated into an action set. If processing requires matching against another flow table, the actions in the matched rule include an action directing processing to the next table in the pipeline. Absent the inclusion of an action in the set to execute all accumulated actions immediately, the actions are executed at the end 403 of the packet processing pipeline. An action allows the writing of data to a metadata register, which is carried along in the packet processing pipe-line like the packet header.")<br><br>Kempf at [0051] ("FIG. 5 is a flowchart of one embodiment of the OpenFlow 1.1 rule matching process. OpenFlow 1.1 contains support for packet tagging. OpenFlow 1.1 allows matching based on header fields and multi-protocol label switching (MPLS) labels. One virtual LAN (VLAN) label and one MPLS label can be matched per table. The rule matching process is initiated with the arrival of a packet to be processed (Block 501). Starting at the first table 0 a lookup is performed to determine a match with the received packet (Block 503). If there is no match in this table, then one of a set of default actions is taken (i.e., send packet to controller, drop the packet or continue to next table) (Block 509). If there is a match, then an update to the action set is made along with counters, packet or match set fields and meta data (Block 505). A check is made to determine the next table to process, which can be the next table sequentially or one specified by an action of a matching rule (Block 507). Once all of the tables have been processed, then the resulting action set is |

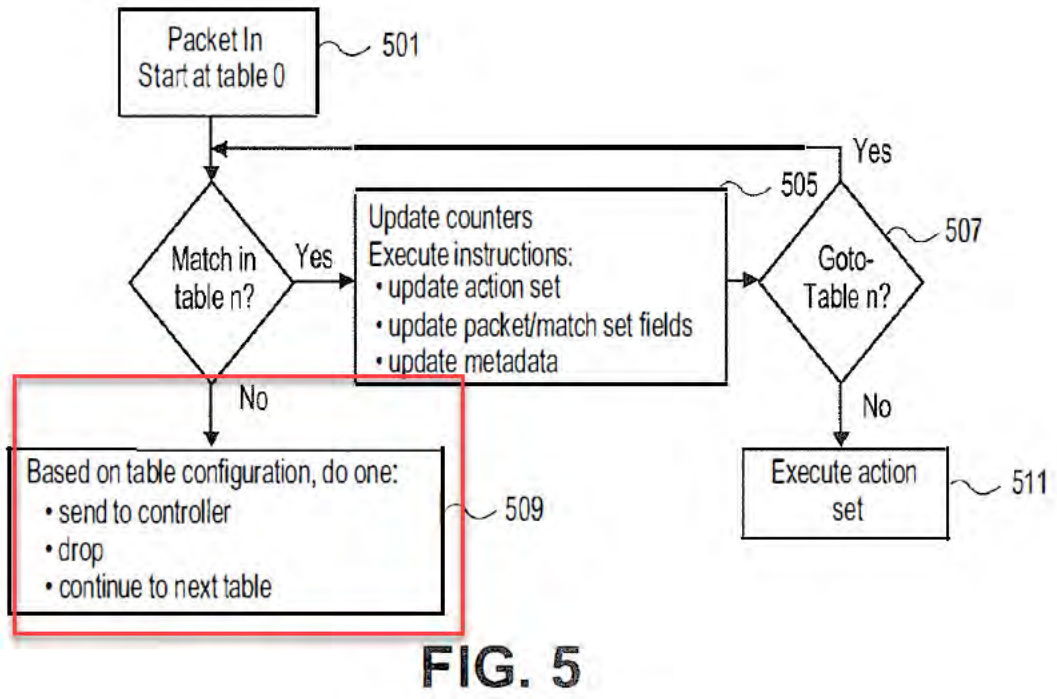


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 17 of 1100

| No. | '111 Patent Claim 1 | Kempf |
|---|---|---|
| | | executed (Block 511). FIG. 6 is a diagram of the fields, which a matching process can utilize for identifying rules to apply to a packet.")<br><br>Kempf at [0053] ("In one embodiment, a group table can be supported in conjunction with the OpenFlow 1.1 protocol. Group tables enable a method for allowing a single flow match to trigger forwarding on multiple ports. Group table entries consist of four fields: a group identifier, which is a 32 bit unsigned integer identifying the group; a group type that determines the group's semantics; counters that maintain statistics on the group; and an action bucket list, which is an ordered list of action buckets, where each bucket contains a set of actions to execute together with their parameters.")<br><br>Kempf at [0091] ("When a packet header matches a rule associated with the virtual port, the GTP TEID is written into the lower 32 bits of the metadata and the packet is directed to the virtual port. The virtual port calculates the hash of the TEID and looks up the tunnel header information in the tunnel header table. If no such tunnel information is present, the packet is forwarded to the controller with an error indication. Other-wise, the virtual port constructs a GTP tunnel header and encapsulates the packet. Any DSCP bits or VLAN priority bits are additionally set in the IP or MAC tunnel headers, and any VLAN tags or MPLS labels are pushed onto the packet. The encapsulated packet is forwarded out the physical port to which the virtual port is bound.")<br><br>Kempf at [0092] ("In one embodiment, the system implements a GTP fast path decapsulation virtual port. When requested by the S-GW and P-GW control plane software running in the cloud computing system, the gateway switch installs rules and actions for routing GTP encapsulated packets out of GTP tunnels. The rules match the GTP header flags and the GTP TEID for the packet, in the modified OpenFlow flow table shown in FIG. 17 as follows: the IP destination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); the UDP destination port is the GTP-U destination port (2152); and the header fields and message type field is wildcarded with the flag 0XFFF0 and the upper two bytes of the field match the G-PDU message type (255) while the lower two bytes match 0x30, i.e. the packet is a GTP packet not a GTP' packet and the version number is 1.") |

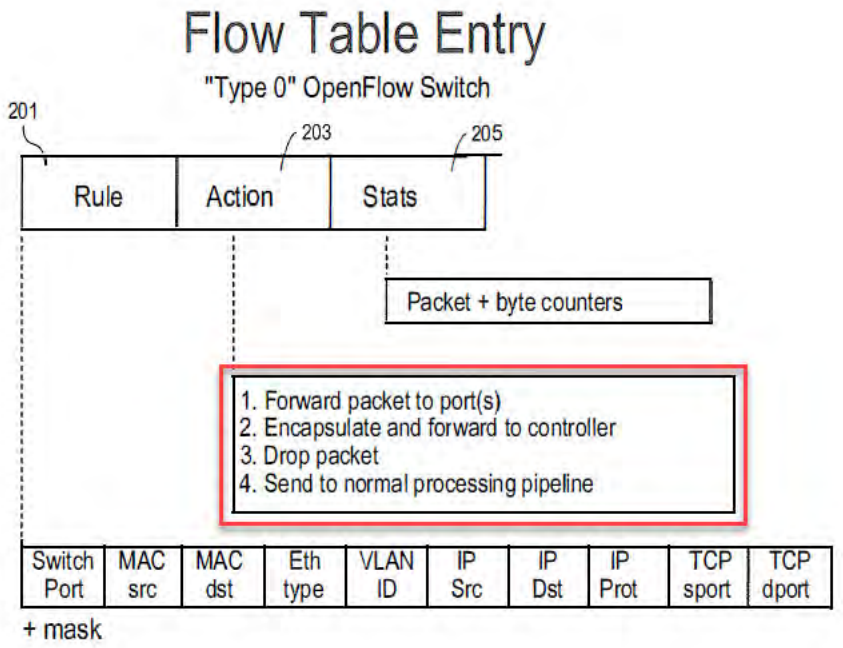


~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 18 of 1100

| No. | '111 Patent Claim 1 | Kempf |
|---|---|---|
| | | Kempf at Figure 5 (annotation added)<br>  <br><br>FIG. 5<br>Kempf at Figure 2 (annotation added) |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 19 of 1100

| No. | '111 Patent Claim 1 | Kempf |
|---|---|---|
| | |  <br>FIG. 2 |
| 1[f] | responsive to the packet satisfying the criterion, sending the packet, by the network node over the packet network, to an entity that is included in the instruction and is other than the second entity. | Kempf discloses responsive to the packet satisfying the criterion, sending the packet, by the network node over the packet network, to an entity that is included in the instruction and is other than the second entity.<br><br>For example, Kempf discloses sending the packet from the network element to the controller or another table, in response to the packet matching the corresponding action in the flow table. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 20 of 1100

| No. | '111 Patent Claim 1 | Kempf |
|---|---|---|
| | | Kempf at [0044] ("FIG. 1 is a diagram of one embodiment of an example network with an OpenFlow switch, conforming to the OpenFlow 1.0 specification. The OpenFlow 1.0 protocol enables a controller 101 to connect to an OpenFlow 1.0 enabled switch 109 using a secure channel 103 and control a single forwarding table 107 in the switch 109. The controller 101 is an external software component executed by a remote computing device that enables a user to configure the Open-Flow 1.0 switch 109. The secure channel 103 can be provided by any type of network including a local area network (LAN) or a wide area network (WAN), such as the Internet.")<br><br>Kempf at [0045] ("FIG. 2 is a diagram illustrating one embodiment of the contents of a flow table entry. The forwarding table 107 is populated with entries consisting of a rule 201 defining matches for fields in packet headers; an action 203 associated to the flow match; and a collection of statistics 205 on the flow. When an incoming packet is received a lookup for a matching rule is made in the flow table 107. If the incoming packet matches a particular rule, the associated action defined in that flow table entry is performed on the packet.")<br><br>Kempf at [0046] ("A rule 201 contains key fields from several headers in the protocol stack, for example source and destination Ethernet MAC addresses, source and destination IP addresses, IP protocol type number, incoming and outgoing TCP or UDP port numbers. To define a flow, all the available matching fields may be used. But it is also possible to restrict the matching rule to a subset of the available fields by using wildcards for the unwanted fields.")<br><br>Kempf at [0047] ("The actions that are defined by the specification of OpenFlow 1.0 are Drop, which drops the matching packets; Forward, which forwards the packet to one or all outgoing ports, the incoming physical port itself, the controller via the secure channel, or the local networking stack (if it exists). OpenFlow 1.0 protocol data units (PDU s) are defined with a set of structures specified using the C programming language. Some of the more commonly used messages are: report switch configuration message; modify state messages (in-cluding a modify flow entry message and port modification message); read state messages, where while the system is running, the datapath may be queried about its current state using this message; and send packet message, which is used when the controller wishes to send a packet out through the datapath.") |

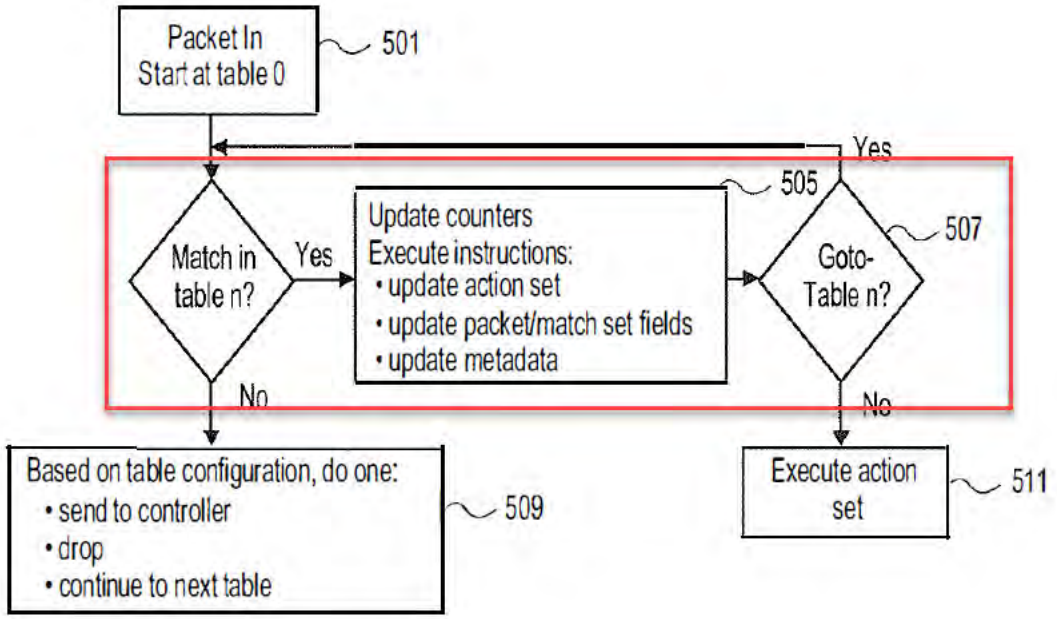


~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 21 of 1100

| No. | '111 Patent Claim 1 | Kempf |
|---|---|---|
| | | Kempf at [0050] ("FIG. 4 illustrates one embodiment of the processing of packets through an OpenFlow 1.1 switched packet pro-cessing pipeline. A received packet is compared against each of the flow tables 401. After each flow table match, the actions are accumulated into an action set. If processing requires matching against another flow table, the actions in the matched rule include an action directing processing to the next table in the pipeline. Absent the inclusion of an action in the set to execute all accumulated actions immediately, the actions are executed at the end 403 of the packet processing pipeline. An action allows the writing of data to a metadata register, which is carried along in the packet processing pipe-line like the packet header.")<br><br>Kempf at [0091] ("When a packet header matches a rule associated with the virtual port, the GTP TEID is written into the lower 32 bits of the metadata and the packet is directed to the virtual port. The virtual port calculates the hash of the TEID and looks up the tunnel header information in the tunnel header table. If no such tunnel information is present, the packet is forwarded to the controller with an error indication. Other-wise, the virtual port constructs a GTP tunnel header and encapsulates the packet. Any DSCP bits or VLAN priority bits are additionally set in the IP or MAC tunnel headers, and any VLAN tags or MPLS labels are pushed onto the packet. The encapsulated packet is forwarded out the physical port to which the virtual port is bound.")<br><br>Kempf at [0106] ("This encapsulates the packet and sends it to the OpenFlow controller.")<br><br>Kempf at Figure 5 (annotation added) |

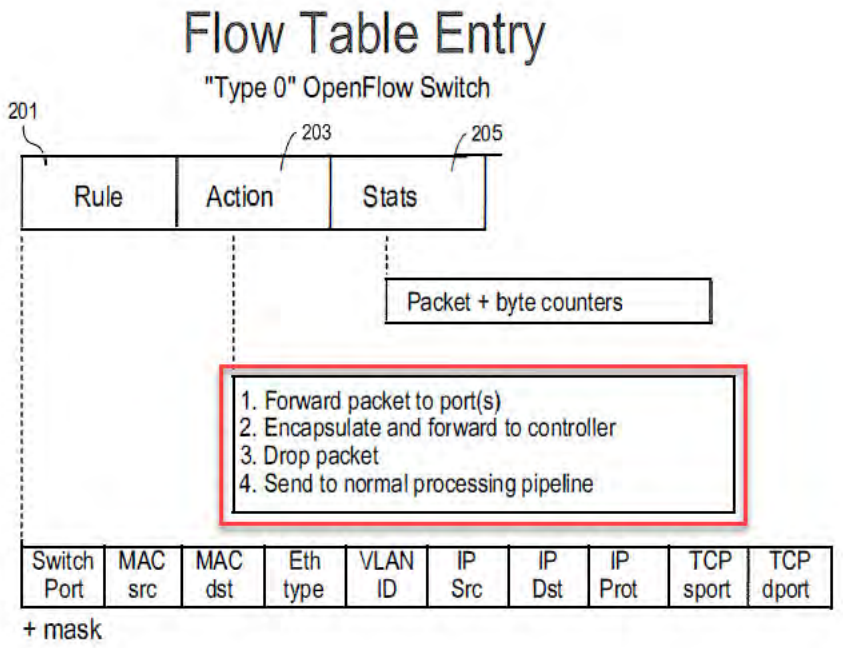


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 22 of 1100

| No. | '111 Patent Claim 1 | Kempf |
|---|---|---|
| | |  <br>FIG. 5<br>Kempf at Figure 2 (annotation added) |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 23 of 1100

| No. | '111 Patent Claim 1 | Kempf |
|---|---|---|
| | |  <br>FIG. 2 |

| No. | '111 Patent Claim 2 | Kempf |
|---|---|---|
| 2[a] | The method according to claim 1, wherein the instruction is 'probe', 'mirror', or 'terminate' instruction, and | Kempf discloses the method according to claim 1, wherein the instruction is 'probe', 'mirror', or 'terminate' instruction.<br><br>For example, Kempf discloses actions associated with a flow match that may be require further processing, duplication, or dropping. A person of ordinary skill in the art would understand that the actions associated with a flow match may be any action, including |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 24 of 1100

| No. | '111 Patent Claim 2 | Kempf |
|---|---|---|
| | | probe, mirror, or terminate. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met.<br><br>Kempf at [0045] ("FIG. 2 is a diagram illustrating one embodiment of the contents of a flow table entry. The forwarding table 107 is populated with entries consisting of a rule 201 defining matches for fields in packet headers; an action 203 associated to the flow match; and a collection of statistics 205 on the flow. When an incoming packet is received a lookup for a matching rule is made in the flow table 107. If the incoming packet matches a particular rule, the associated action defined in that flow table entry is performed on the packet.")<br><br>Kempf at [0047] ("The actions that are defined by the specification of OpenFlow 1.0 are Drop, which drops the matching packets; Forward, which forwards the packet to one or all outgoing ports, the incoming physical port itself, the controller via the secure channel, or the local networking stack (if it exists). OpenFlow 1.0 protocol data units (PDU s) are defined with a set of structures specified using the C programming language. Some of the more commonly used messages are: report switch configuration message; modify state messages (in-cluding a modify flow entry message and port modification message); read state messages, where while the system is running, the datapath may be queried about its current state using this message; and send packet message, which is used when the controller wishes to send a packet out through the datapath.")<br><br>Kempf at [0051] ("FIG. 5 is a flowchart of one embodiment of the OpenFlow 1.1 rule matching process. OpenFlow 1.1 contains support for packet tagging. OpenFlow 1.1 allows matching based on header fields and multi-protocol label switching (MPLS) labels. One virtual LAN (VLAN) label and one MPLS label can be matched per table. The rule matching process is initiated with the arrival of a packet to be processed (Block 501 ). Starting at the first table 0 a lookup is performed to determine a match with the received packet (Block 503). If there is no match in this table, then one of a set of default actions is taken (i.e., send packet to controller, drop the packet or continue to next table) (Block 509). If there is a match, then an update to the action set is made along with counters, packet or match set fields and meta data (Block 505). A check is made to determine the next table to process, which can be the next table sequentially or one specified by an action of a matching rule (Block 507). Once all of the tables have been processed, then the resulting action set is |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 25 of 1100

| No. | '111 Patent Claim 2 | Kempf |
|---|---|---|
| | | executed (Block 511). FIG. 6 is a diagram of the fields, which a matching process can utilize for identifying rules to apply to a packet.")<br><br>Kempf at [0052] ("Actions allow manipulating of tag stacks by pushing and popping labels. Combined with multiple tables, VLAN or MPLS label stacks can be processed by matching one label per table. FIG. 7 is a flow chart of one embodiment of a header parsing process. The parsing process matches a packet header by initializing a set of match fields (Block 701) and checking for the presence of a set of different header types. The process checks for a VLAN tag (Block 703). If the VLAN tag is present, then there are a series of processing steps for the VLAN tag (Blocks 705-707). If the switch supports MPLS (Block 709), then there are a series of steps for detecting and processing the MPLS header information (Blocks 711-715). If the switch supports address resolution protocol (ARP), then there are a series of steps for processing the ARP header (Blocks 719 and 721). If the packet has an IP header (Block 723), then there are a series of steps for processing the IP header (Blocks 725-733). This process is performed for each received packet.")<br><br>Kempf at [0055] ("OpenFlow 1.1 can be utilized to support virtual ports. A virtual port, as used herein, is an "action block" that performs some kind of processing action other than simply forwarding the packet out to a network connection like physi-cal ports do. Examples of a few built-in virtual ports include: ALL, which forwards the port out all ports except for the ingress port and any ports that are marked "Do Not Forward;" CONTROLLER, which encapsulates the packet and sends it to the controller; TABLE, which inserts the packet into the packet processing pipeline by submitting it to the first flow table, this action is only valid in the action set of a packet-out message; and IN_PORT, which sends the packet out the ingress port. In other embodiments, there can also be switched-defined virtual ports.") |

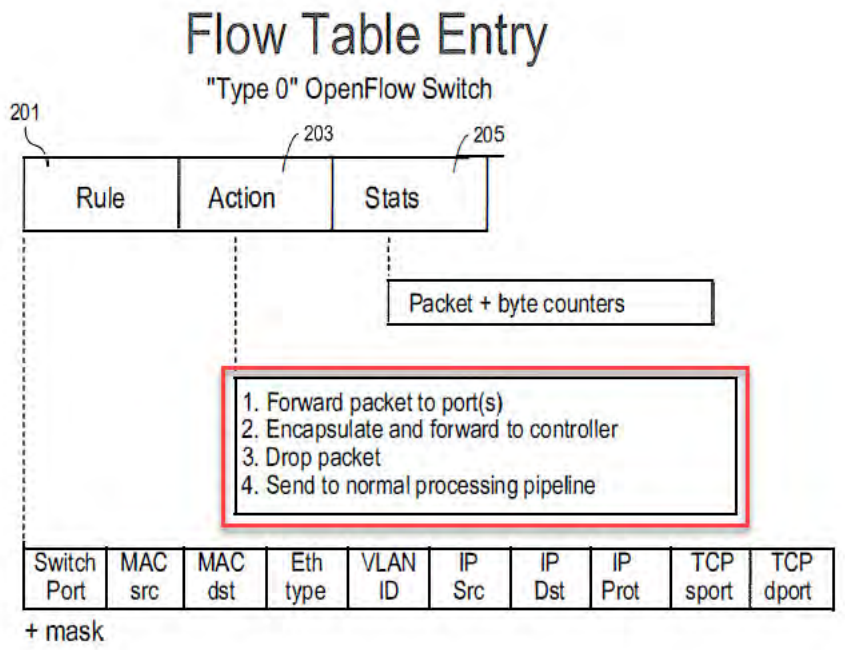


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 26 of 1100

| No. | '111 Patent Claim 2 | Kempf |
|---|---|---|
| 2[b] | upon receiving by the network node the 'terminate' instruction, the method further comprising blocking, by the network node, the packet from being sent to the second entity and to the controller. | Kempf discloses upon receiving by the network node the 'terminate' instruction, the method further comprising blocking, by the network node, the packet from being sent to the second entity and to the controller.<br><br>For example, Kempf discloses actions associated with a flow match including dropping packets from being sent any further.<br><br>Kempf at [0045] ("FIG. 2 is a diagram illustrating one embodiment of the contents of a flow table entry. The forwarding table 107 is populated with entries consisting of a rule 201 defining matches for fields in packet headers; an action 203 associated to the flow match; and a collection of statistics 205 on the flow. When an incoming packet is received a lookup for a matching rule is made in the flow table 107. If the incoming packet matches a particular rule, the associated action defined in that flow table entry is performed on the packet.")<br><br>Kempf at [0047] ("The actions that are defined by the specification of OpenFlow 1.0 are Drop, which drops the matching packets; Forward, which forwards the packet to one or all outgoing ports, the incoming physical port itself, the controller via the secure channel, or the local networking stack (if it exists). OpenFlow 1.0 protocol data units (PDU s) are defined with a set of structures specified using the C programming language. Some of the more commonly used messages are: report switch configuration message; modify state messages (in-cluding a modify flow entry message and port modification message); read state messages, where while the system is running, the datapath may be queried about its current state using this message; and send packet message, which is used when the controller wishes to send a packet out through the datapath.")<br><br>Kempf at [0051] ("FIG. 5 is a flowchart of one embodiment of the OpenFlow 1.1 rule matching process. OpenFlow 1.1 contains support for packet tagging. OpenFlow 1.1 allows matching based on header fields and multi-protocol label switching (MPLS) labels. One virtual LAN (VLAN) label and one MPLS label can be matched per table. The rule matching process is initiated with the arrival of a packet to be processed (Block 501 ). Starting at the first table 0 a lookup is performed to determine a match with the received packet (Block 503). If there is no match in this table, then one of a set of default actions is taken (i.e., send packet to controller, drop the packet or continue to next table) (Block 509). |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 27 of 1100

| No. | '111 Patent Claim 2 | Kempf |
|---|---|---|
| | | If there is a match, then an update to the action set is made along with counters, packet or match set fields and meta data (Block 505). A check is made to determine the next table to process, which can be the next table sequentially or one specified by an action of a matching rule (Block 507). Once all of the tables have been processed, then the resulting action set is executed (Block 511). FIG. 6 is a diagram of the fields, which a matching process can utilize for identifying rules to apply to a packet.")<br><br>Kempf at Figure 2 (annotation added)<br><br>  <br>FIG. 2 |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 28 of 1100

| No. | '111 Patent Claim 3 | Kempf |
|---|---|---|
| 3[a] | The method according to claim 1, wherein the instruction is a 'probe', a 'mirror', or a 'terminate' instruction, and | Kempf discloses the method according to claim 1, wherein the instruction is a 'probe', a 'mirror', or a 'terminate' instruction.<br><br>*See supra* at 2(a). |
| 3[b] | upon receiving by the network node the 'mirror' instruction and responsive to the packet satisfying the criterion, the method further comprising sending the packet, by the network node, to the second entity and to the controller. | Kempf discloses upon receiving by the network node the 'mirror' instruction and responsive to the packet satisfying the criterion, the method further comprising sending the packet, by the network node, to the second entity and to the controller.<br><br>For example, Kempf discloses actions associated with a flow match including an action to forward the packet to the destination device and to the controller. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Kempf is found to not meet this limitation, cou upon receiving by the network node the 'mirror' instruction and responsive to the packet satisfying the criterion, method further comprising sending the packet, by the network node, to the second entity and to the controller would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>Kempf at [0045] ("FIG. 2 is a diagram illustrating one embodiment of the contents of a flow table entry. The forwarding table 107 is populated with entries consisting of a rule 201 defining matches for fields in packet headers; an action 203 associated to the flow match; and a collection of statistics 205 on the flow. When an incoming packet is received a lookup for a matching rule is made in the flow table 107. If the incoming packet matches a particular rule, the associated action defined in that flow table entry is performed on the packet.")<br><br>Kempf at [0046] ("A rule 201 contains key fields from several headers in the protocol stack, for example source and destination Ethernet MAC addresses, source and destination IP addresses, IP protocol type number, incoming and outgoing TCP or UDP port numbers. To define a flow, all the available matching fields may be used. But it is also possible to restrict the matching rule to a subset of the available fields by using wildcards for the unwanted fields.") |

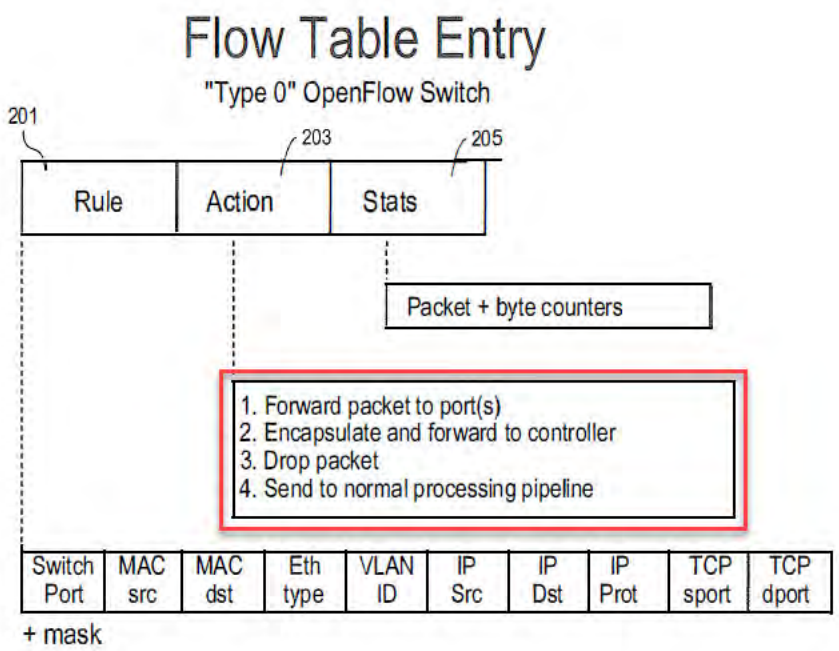


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 29 of 1100

| No. | '111 Patent Claim 3 | Kempf |
|---|---|---|
| | | Kempf at [0047] ("The actions that are defined by the specification of OpenFlow 1.0 are Drop, which drops the matching packets; Forward, which forwards the packet to one or all outgoing ports, the incoming physical port itself, the controller via the secure channel, or the local networking stack (if it exists). OpenFlow 1.0 protocol data units (PDU s) are defined with a set of structures specified using the C programming language. Some of the more commonly used messages are: report switch configuration message; modify state messages (in-cluding a modify flow entry message and port modification message); read state messages, where while the system is running, the datapath may be queried about its current state using this message; and send packet message, which is used when the controller wishes to send a packet out through the datapath.")<br><br>Kempf at [0106] ("This encapsulates the packet and sends it to the OpenFlow controller.")<br><br>Kempf at Figure 2 (annotation added) |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 30 of 1100

| No. | '111 Patent Claim 3 | Kempf |
|---|---|---|
| | |  <br>FIG. 2<br><br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Kempf in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 3(b) of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references.<br><br>For example, Chua discloses a mirror program in response to an indication based on the packet header in which the network devices mirror copies of the packets of the packet flow |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 31 of 1100

| No. | '111 Patent Claim 3 | Kempf |
|---|---|---|
| | | to a second service device while forwarding the packets of the packet flow to the destination of the packet flow.<br><br>Chua at 7:28-54 ("SDN controller 112 may receive data as input from service devices 116. For example, SDN controller 112 may be con-figured to receive data from an intrusion detection system (IDS) device, a Denial of Service (DoS) device, a Distributed Denial of Service (DDoS) device, an intrusion prevention system (IPS) device, or the like. Based on this information, SDN controller 112 may make network enforcement decisions for specific traffic flows. That is, SDN controller 112 may program network devices of SDN 106 to perform pro-grammed actions on packets of a packet flow based on this data. Such programmed actions may include:<br><br>Allow-explicitly allow a certain network flow to proceed to its destination<br>Block-explicitly block a certain flow from traversing SDN 106<br>Mirror-allow the traffic, but send a copy of the traffic for deeper inspection or recording to, e.g., one of service devices 116<br>Redirect-redirect the traffic to another network (such as a honeypot device or other device of service devices 116) for either inspection or to keep a potential hacker 'busy' to determine if there is a real security threat.<br>Transform-modify or translate values of headers of packets in the network flow<br>Encapsulate-encapsulate packets in the network flow with a particular header")<br><br>Chua at 16:23-44 ("More particularly, control unit 130 may configure any of service devices 116 to send data representative of a particular event to SDN controller 112, and control unit 130 may auto-matically reprogram one or more network devices of SDN 106 in response to such data. For example, security monitor-ing applications of service devices 116 may determine that a specific source port, destination port, source IP address, des-tination IP address, or the like should be acted upon. Alter-natively, security monitoring applications may determine that, due to content or deep packet inspection, a specific type of traffic is malicious and should be blocked. In either case, the corresponding one of service devices 116 may send a message to SDN controller 112 representative of these deter-minations. As yet another example, a network performance device may monitor various performance metrics, such as latency, jitter, packet loss, or the like, and provide feedback data to SDN controller 112 based on these metrics. SDN controller 112 may respond by programming |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 32 of 1100

| No. | '111 Patent Claim 3 | Kempf |
|---|---|---|
| | | network devices of SDN 106 to perform a programmed action, such as allowing corresponding traffic, blocking corresponding traf-fic, mirroring corresponding traffic, redirecting correspond-ing traffic.")<br><br>Chua at 28:7-32 ("In addition, SDN controller 112 may configure the service device to send service-related data to one or more network devices (334). The service-related data may cause the net-work devices to change a path along which the packet is forwarded. For example, when the service device is a security device (e.g., a firewall or an IDS), if the security device determines that one or more packets of a packet flow are malicious, the security device may send service data indicat-ing that the packet flow includes malicious data. SDN con-troller 112 may program the network devices of the SDN to perform a programmed action based on the service-related data (336). For example, SDN controller 112 may program network devices to, in response to an indication that packets of a packet flow include malicious data, forward packets of the packet flow to a destination of the packet flow, forward packets of malicious packet flows to a collection device for further analysis, cause network devices to drop packets of the malicious packet flows, send a close session message to devices from which packets of the malicious packet flows were received, block the packets of the packet flow, mirror copies of the packets of the packet flow to a second service device while forwarding the packets of the packet flow to the destination of the packet flow, redirect the packets of the packet flow to a third service device, transform one or more values of headers of the packets, and/or encapsulate the pack-ets with a particular header, or other such actions.")<br><br>As another example, Swenson discloses a counting mode instructed by the network controller to the steering device for monitoring and optimizing, in which the steering device forwards the packet flow to the user device/origin server and at the same time, sending the packet flow to the network controller.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 33 of 1100

| No. | '111 Patent Claim 3 | Kempf |
|---|---|---|
| | | interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0028] ("In contrast to conventional inline TCP throughput monitoring devices that monitor every single data packets transmitted and received, the network controller 140 is an "out-of-band" computer server that interfaces with the steer-ing device 130 to selectively inspect user flows of interest. The network controller 140 may further identify user flows (e.g., among the flows of interest) for optimization. In one embodiment, the network controller 140 may be imple-mented at the steering device 130 to monitor traffic. In other embodiments, the network controller 140 is coupled to and communicates with the steering device 130 for traffic moni-toring and optimization. When queried by the steering device 130, the network controller 140 determines if a given network flow should be ignored, monitored further or optimized. Opti-mization of a flow is often decided at the beginning of the flow because it is rarely possible to switch to optimized content mid-stream once non-optimized content delivery has begun. However, the network controller 140 may determine that existing flows associated with a particular subscriber or other entity should be optimized. In turn, new flows ( e.g., resulting from seek requests in media, new media requests, resume after pause, etc.) determined to be associated with the entity may be optimized. The network controller 140 uses the net-work state as well as historical traffic data in its decision for monitoring and optimization. Knowledge on the current net-work state, such as congestion, deems critical when it comes to data optimization.")<br><br>Swenson at [0059] ("In one embodiment, as the steering device 130 monitors network responses, it is looking for flows that match one or more signatures for video and images. When a match-ing flow is detected, the steering device 130 forwards the HTTP request and a portion of the HTTP response to the network controller 140 over the ICAP client interface 404. After receiving the request and the portion of response at the ICAP server interface 406, the flow analyzer 312 of the net-work controller 140 performs a deep flow inspection to deter-mine if the flow is worth bandwidth monitoring and/or user detection. For example, the flow inspection performed by the flow analyzer 312 may determine if the flow indeed contains large or medium object ( e.g., larger than 50 kB), and/or if the source IP address of the flow is from a user or a group of users that are required to be monitored by policies. The |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 34 of 1100

| No. | '111 Patent Claim 3 | Kempf |
|---|---|---|
| | | flow ana-lyzer 312 may also determine if the flow needs to be opti-mized based on historical flow statistical data.")<br><br>Swenson at [0064] ("Similar to the "continue" mode, after receiving the initial HTTP messages of a flow and determining to monitor the flow, the network controller 140 notify the steering device 130 to work in a "counting" mode for bandwidth monitoring. In contrast to the "continue" mode, when a matching flow is detected for "counting" mode, the steering device 130 for-wards the HTTP response directly to the user device 110. While at the same time, the steering device 130 send a cus-tomized ICAP message to the network controller 140 over the network link 425. In one embodiment, the customized ICAP message contains the HTTP request and response headers, as well as a count of payload size of the current flow. After updating the flow statistics, the network controller 140 may acknowledge the gateway over the network line 426. In the "counting" mode, the network controller 140 does not join the network response path as an inline network element, but simply listens to the counting of flow size. The benefit of the "counting" mode is to off-load the network controller 140 from ingesting and forwarding the network flow on the net-work response path, while still enabling the detection of con-gestions and estimation of bandwidth associated with the flows of interest.")<br><br>Swenson at [0071] ("After receiving the request, the video optimizer 150 forwards the video HTTP GET requests 622 to the origin server 160 and in return, receives a video file 624 from the origin server 160. The video optimizer 150 transcodes the video file to a format usable by the client device 110 based on network bandwidth available to the user device 110. The optimized video 626 is then transmitted from the video opti-mizer 150 to the steering device 130. In one embodiment, the steering device 130 intercepts the optimized video 626. The steering device 130 will then send an ICAP request to the network controller 140 for inspection. The network controller 140 deems this flow to be monitored and sends ICAP response 630. The steering device 130 then allows the flow to go through to the user device 110. The steering device 130 next sends periodic ICAP "counting" updates 632 to the network controller 140 until the flow completes. As such, the client receives the optimized video 626 for substantially real-time playback on an application executing on the user device 110.")<br>Swenson at [0072] ("In one embodiment, if the video optimizer 150 failed to retrieve user requested video file from the origin server 160, the video optimizer 150 appends a "do not |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 35 of 1100

| No. | '111 Patent Claim 3 | Kempf |
|---|---|---|
| | | transcode" flag to the HTTP redirect request and returned to the user device 110, which re-sends the request out over the network to the origin server 160. The origin server 160 responds appropriately to the request by sending back video 624, which is intercepted by the steering device 130 only. The steering device 130 forwards the video to the user device 110 and at the same time reports the flow size to the network controller 140 for monitoring purpose.") |

| No. | '111 Patent Claim 4 | Kempf |
|---|---|---|
| 4[a] | The method according to claim 1, wherein the instruction is 'probe', 'mirror', or 'terminate' instruction, and | Kempf discloses the method according to claim 1, wherein the instruction is 'probe', 'mirror', or 'terminate' instruction.<br><br>*See supra* at 2(a). |
| 4[b] | upon receiving by the network node the 'probe' instruction and responsive to the packet satisfying the criterion, the method further comprising: sending the packet, by the network node, to the controller; | Kempf discloses upon receiving by the network node the 'probe' instruction and responsive to the packet satisfying the criterion, the method further comprising: sending the packet, by the network node, to the controller.<br><br>For example, Kempf discloses actions associated with a flow match including an action for further processing and sending the packet to the controller.<br><br>Kempf at [0045] ("FIG. 2 is a diagram illustrating one embodiment of the contents of a flow table entry. The forwarding table 107 is populated with entries consisting of a rule 201 defining matches for fields in packet headers; an action 203 associated to the flow match; and a collection of statistics 205 on the flow. When an incoming packet is received a lookup for a matching rule is made in the flow table 107. If the incoming packet matches a particular rule, the associated action defined in that flow table entry is performed on the packet.")<br><br>Kempf at [0046] ("A rule 201 contains key fields from several headers in the protocol stack, for example source and destination Ethernet MAC addresses, source and destination IP addresses, IP protocol type number, incoming and outgoing TCP or UDP port numbers. To |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 36 of 1100

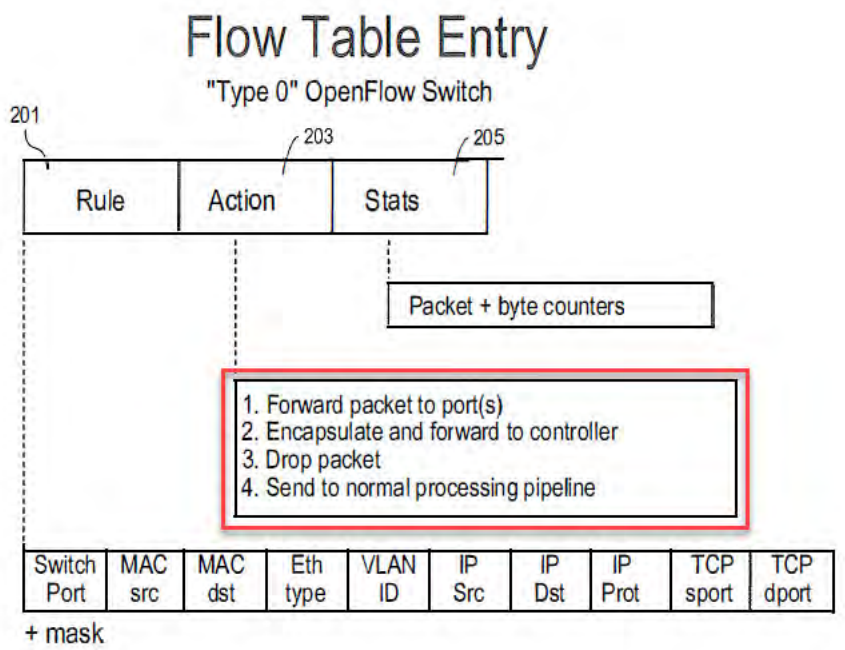
| No. | '111 Patent Claim 4 | Kempf |
|---|---|---|
| | | define a flow, all the available matching fields may be used. But it is also possible to restrict the matching rule to a subset of the available fields by using wildcards for the unwanted fields.")<br><br>Kempf at [0047] ("The actions that are defined by the specification of OpenFlow 1.0 are Drop, which drops the matching packets; Forward, which forwards the packet to one or all outgoing ports, the incoming physical port itself, the controller via the secure channel, or the local networking stack (if it exists). OpenFlow 1.0 protocol data units (PDU s) are defined with a set of structures specified using the C programming language. Some of the more commonly used messages are: report switch configuration message; modify state messages (in-cluding a modify flow entry message and port modification message); read state messages, where while the system is running, the datapath may be queried about its current state using this message; and send packet message, which is used when the controller wishes to send a packet out through the datapath.")<br><br>Kempf at [0050] ("FIG. 4 illustrates one embodiment of the processing of packets through an OpenFlow 1.1 switched packet pro-cessing pipeline. A received packet is compared against each of the flow tables 401. After each flow table match, the actions are accumulated into an action set. If processing requires matching against another flow table, the actions in the matched rule include an action directing processing to the next table in the pipeline. Absent the inclusion of an action in the set to execute all accumulated actions immediately, the actions are executed at the end 403 of the packet processing pipeline. An action allows the writing of data to a metadata register, which is carried along in the packet processing pipe-line like the packet header.")<br><br>Kempf at [0051] ("FIG. 5 is a flowchart of one embodiment of the OpenFlow 1.1 rule matching process. OpenFlow 1.1 contains support for packet tagging. OpenFlow 1.1 allows matching based on header fields and multi-protocol label switching (MPLS) labels. One virtual LAN (VLAN) label and one MPLS label can be matched per table. The rule matching process is initiated with the arrival of a packet to be processed (Block 501 ). Starting at the first table 0 a lookup is performed to determine a match with the received packet (Block 503). If there is no match in this table, then one of a set of default actions is taken (i.e., send packet to controller, drop the packet or continue to next table) (Block 509). If there is a match, then an update to the action set is made along with counters, packet or |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 37 of 1100

| No. | '111 Patent Claim 4 | Kempf |
|---|---|---|
| | | match set fields and meta data (Block 505). A check is made to determine the next table to process, which can be the next table sequentially or one specified by an action of a matching rule (Block 507). Once all of the tables have been processed, then the resulting action set is executed (Block 511). FIG. 6 is a diagram of the fields, which a matching process can utilize for identifying rules to apply to a packet.")<br><br>Kempf at [0055] ("OpenFlow 1.1 can be utilized to support virtual ports. A virtual port, as used herein, is an "action block" that performs some kind of processing action other than simply forwarding the packet out to a network connection like physi-cal ports do. Examples of a few built-in virtual ports include: ALL, which forwards the port out all ports except for the ingress port and any ports that are marked "Do Not Forward;" CONTROLLER, which encapsulates the packet and sends it to the controller; TABLE, which inserts the packet into the packet processing pipeline by submitting it to the first flow table, this action is only valid in the action set of a packet-out message; and IN_PORT, which sends the packet out the ingress port. In other embodiments, there can also be switched-defined virtual ports.")<br><br>Kempf at [0083] ("If a packet either needs encapsulation or arrives encapsulated with nonzero header flags, header extensions, and/or the GTP-U packet is not a G-PDU packet (i.e. it is a GTP-U control packet), the processing must proceed via the gateway's slow path (software) control plane. GTP-C and GTP' packets directed to the gateway's IP address are a result of mis-configuration and are in error. They must be sent to the OpenFlow controller, since these packets are handled by the S-GW-C and P-GW-C control plane entities in the cloud computing system or to the billing entity handling GTP' and not the S-GW-D and P-GW-D data plane switches.")<br><br>Kempf at [0091] ("When a packet header matches a rule associated with the virtual port, the GTP TEID is written into the lower 32 bits of the metadata and the packet is directed to the virtual port. The virtual port calculates the hash of the TEID and looks up the tunnel header information in the tunnel header table. If no such tunnel information is present, the packet is forwarded to the controller with an error indication. Other-wise, the virtual port constructs a GTP tunnel header and encapsulates the packet. Any DSCP bits or VLAN priority bits are additionally set in the IP or MAC tunnel headers, and any VLAN tags or MPLS labels are pushed onto the packet. The encapsulated packet is forwarded out the physical port to which the virtual port is bound.") |


~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 38 of 1100

| No. | '111 Patent Claim 4 | Kempf |
|---|---|---|
| | | Kempf at [0106] ("This encapsulates the packet and sends it to the OpenFlow controller.")<br><br>Kempf at Figure 2 (annotation added)<br><br>  <br>FIG. 2 |
| 4[c] | responsive to receiving the packet, analyzing the packet, by the controller; | Kempf discloses responsive to receiving the packet, analyzing the packet, by the controller.<br><br>For example, Kempf discloses further processing by the controller in response to a packet flow match indicating that a packet needs encapsulation or arrives encapsulated with |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 39 of 1100

| No. | '111 Patent Claim 4 | Kempf |
|---|---|---|
| | | nonzero header flags, header extensions, and/or the GTP-U packet is not a G-PDU packet (i.e. it is a GTP-U control packet).<br><br>Kempf at [0037] ("The EPC architecture also contains little flexibility for specialized treatment of user flows. Though the architec-ture does provide support for establishing quality of service (QoS), other sorts of data management are not available. For example services involving middle boxes, such as specialized deep packet inspection or interaction with local data caching and processing resources that might be utilized for transcod-ing or augmented reality applications, is difficult to support with the current EPC architecture. Almost all such applica-tions require the packet flows to exit through the PDN Gate-way, thereby being de-tunnelled from GTP, and to be pro-cessed within the wired network.")<br><br>Kempf at [0074] ("The operation of the EPC cloud computer system as follows. The UE 1317, E-NodeB 1317, S-GW-C 1307, and P-GW-C signal 1307 to the MME, PCRF, and HSS 1307 using the standard EPC protocols, to establish, modify, and delete bearers and GTP tunnels. This signaling triggers pro-cedure calls with the OpenFlow controller to modify the routing in the EPC as requested. The OpenFlow controller configures the standard OpenFlow switches, the Openflow S-GW-D 1315, and P-GW-D 1311 with flow rules and actions to enable the routing requested by the control plane entities. Details of this configuration are described in further detail herein below.")<br><br>Kempf at [0083] ("If a packet either needs encapsulation or arrives encapsulated with nonzero header flags, header extensions, and/or the GTP-U packet is not a G-PDU packet (i.e. it is a GTP-U control packet), the processing must proceed via the gateway's slow path (software) control plane. GTP-C and GTP' packets directed to the gateway's IP address are a result of mis-configuration and are in error. They must be sent to the OpenFlow controller, since these packets are handled by the S-GW-C and P-GW-C control plane entities in the cloud computing system or to the billing entity handling GTP' and not the S-GW-D and P-GW-D data plane switches.")<br><br>Kempf at [0084] ("GTP virtual ports are configured from the Open-Flow controller using a configuration protocol. The details of the configuration protocol are switch-dependent. The con-figuration protocol must support messages that perform following functions: allow the controller to query for and return an indication whether the switch supports GTP fast path |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 40 of 1100

| No. | '111 Patent Claim 4 | Kempf |
|---|---|---|
| | | virtual ports and what virtual port numbers are used for fast path and slow path GTP-U processing; and allow the controller to instantiate a GTP-U fast path virtual port within a switch datapath for use in the OpenFlow table set-output-port action. The configuration command must be run in a transaction so that, when the results of the action are reported back to the controller, either a GTP-U fast path virtual port for the requested datapath has been instantiated or an error has returned indicating why the request could not be honored. The command also allows the OpenFlow controller to bind a GTP-U virtual port to a physical port. For decapsulation virtual ports, the physical port is an input port. For encapsu-lation virtual ports, the physical port is an output port.")<br><br>Kempf at [0091] ("When a packet header matches a rule associated with the virtual port, the GTP TEID is written into the lower 32 bits of the metadata and the packet is directed to the virtual port. The virtual port calculates the hash of the TEID and looks up the tunnel header information in the tunnel header table. If no such tunnel information is present, the packet is forwarded to the controller with an error indication. Other-wise, the virtual port constructs a GTP tunnel header and encapsulates the packet. Any DSCP bits or VLAN priority bits are additionally set in the IP or MAC tunnel headers, and any VLAN tags or MPLS labels are pushed onto the packet. The encapsulated packet is forwarded out the physical port to which the virtual port is bound.")<br><br>Kempf at [0104] ("In one embodiment, the system implements han-dling of GTP-C and GTP' control packets. Any GTP-C and GTP' control packets that are directed to IP addresses on a gateway switch are in error. These packets need to be handled by the S-GW-C, P-GW-C, and GTP' protocol entities in the cloud computing system, not the S-GW-D and P-GW-D enti-ties in the switches. To catch such packets, the OpenFlow controller must program the switch with the following two rules: the IP destination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); for one rule, the UDP destination port is the GTP-U destination port (2152), for the other, the UDP destination port is the GTP-C destination port (2123); the GTP header flags and message type fields are wildcarded.")<br><br>Kempf at [0106] ("This encapsulates the packet and sends it to the OpenFlow controller.") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 41 of 1100

| No. | '111 Patent Claim 4 | Kempf |
|---|---|---|
| 4[d] | sending the packet, by the controller, to the network node; and | Kempf discloses sending the packet, by the controller, to the network node.<br><br>For example, Kempf discloses the controller sending the packet back through the intended datapath via the network element. A person of ordinary skill would understand that the packet, once sent to the controller for further processing, is then sent back to the network element to be forwarded to its originally intended destination. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Kempf is found to not meet this limitation, sending the packet, by the controller, to the network node would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>Kempf at [0047] ("The actions that are defined by the specification of OpenFlow 1.0 are Drop, which drops the matching packets; Forward, which forwards the packet to one or all outgoing ports, the incoming physical port itself, the controller via the secure channel, or the local networking stack (if it exists). OpenFlow 1.0 protocol data units (PDU s) are defined with a set of structures specified using the C programming language. Some of the more commonly used messages are: report switch configuration message; modify state messages (in-cluding a modify flow entry message and port modification message); read state messages, where while the system is running, the datapath may be queried about its current state using this message; and send packet message, which is used when the controller wishes to send a packet out through the datapath.")<br><br>Kempf at [0055] ("OpenFlow 1.1 can be utilized to support virtual ports. A virtual port, as used herein, is an "action block" that performs some kind of processing action other than simply forwarding the packet out to a network connection like physi-cal ports do. Examples of a few built-in virtual ports include: ALL, which forwards the port out all ports except for the ingress port and any ports that are marked "Do Not Forward;" CONTROLLER, which encapsulates the packet and sends it to the controller; TABLE, which inserts the packet into the packet processing pipeline by submitting it to the first flow table, this action is only valid in the action set of a packet-out message; and IN_PORT, which sends the packet out the ingress port. In other embodiments, there can also be switched-defined virtual ports.")<br><br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Kempf in combination with (1) the knowledge of a person of ordinary skill in the art, alone |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 42 of 1100

| No. | '111 Patent Claim 4 | Kempf |
|---|---|---|
| | | or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 4(d) of Exhibit E-4 renders the claim, including the present limitation, obvious. Below is an example.<br><br>For example, Swenson discloses sending the packet, for example a video or image, back to the steering device after the network controller analyzes the packet and updates flow statistics.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0028] ("In contrast to conventional inline TCP throughput monitoring devices that monitor every single data packets transmitted and received, the network controller 140 is an "out-of-band" computer server that interfaces with the steer-ing device 130 to selectively inspect user flows of interest. The network controller 140 may further identify user flows (e.g., among the flows of interest) for optimization. In one embodiment, the network controller 140 may be imple-mented at the steering device 130 to monitor traffic. In other embodiments, the network controller 140 is coupled to and communicates with the steering device 130 for traffic moni-toring and optimization. When queried by the steering device 130, the network controller 140 determines if a given network flow should be ignored, monitored further or optimized. Opti-mization of a flow is often decided at the beginning of the flow because it is rarely possible to switch to optimized content mid-stream once non-optimized content delivery has begun. However, the network controller 140 may determine that existing flows associated with a particular subscriber or other entity should be optimized. In turn, new flows ( e.g., resulting from seek requests in media, new media |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 43 of 1100

| No. | '111 Patent Claim 4 | Kempf |
|---|---|---|
| | | requests, resume after pause, etc.) determined to be associated with the entity may be optimized. The network controller 140 uses the net-work state as well as historical traffic data in its decision for monitoring and optimization. Knowledge on the current net-work state, such as congestion, deems critical when it comes to data optimization.")<br><br>Swenson at [0029] ("As a flow is sent to the network controller 140 for inspection, historical network traffic data stored at the net-work controller 140 may be searched. The historical network traffic data includes information such as subscriber informa-tion, the cell towers to which the user devices attached, rout-ers through which the traffic is passing, geography regions, the backhaul segments, and time-of-day of the flows. For example, in a mobile network, the cell tower to which a user device is attached can be most useful, since it is the location where most congestion occurs due to limited bandwidth and high cost of the radio access network infrastructure. The network controller 140 looks into the historical traffic data for the average of the bandwidth per user at the particular cell tower. The network controller 140 can then estimate the amount ofbandwidth or degree of congestion for the new flow based on the historical record.")<br><br>Swenson at [0057] ("The Internet content adaption protocol is a light-weight protocol aimed at executing a simple remote proce-dure call on HTTP messages. ICAP leverages edge-based devices to help deliver value-added services using transparent HTTP proxy caches. Content adaptation refers to performing the particular value added service, such as content manipula-tion or other processing, for the associated HTTP client request/response. ICAP clients pass HTTP messages to ICAP servers for transformation or other processing. In tum, the ICAP server executes its transformation service on the HTTP messages and sends back responses to the ICAP client. At the core of this process is a cache that can proxy all client trans-actions and process them through ICAP servers, which may focus on specific functions, such as ad insertion, virus scan-ning, content translation, language translation, or content fil-tering. ICAP servers, such as those utilized by the network controller 140, handle these tasks to off-load value-added services from network devices including an ICAP client, such as the steering device 130. By offloading value added services from the steering device 130, processing infrastructure (e.g., optimization services and network controllers) may be scaled independent from the steering devices handling raw HTTP throughput.") |

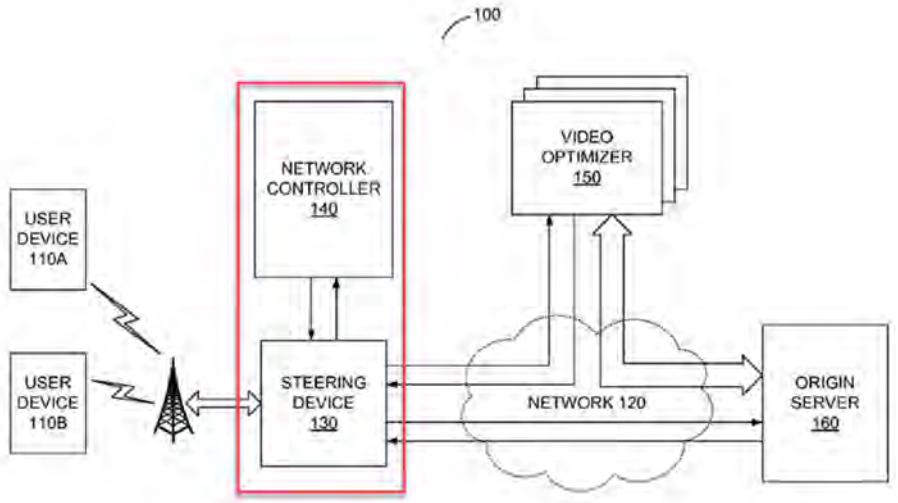


~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 44 of 1100

| No. | '111 Patent Claim 4 | Kempf |
|---|---|---|
| | | Swenson at [0059] ("In one embodiment, as the steering device 130 monitors network responses, it is looking for flows that match one or more signatures for video and images. When a match-ing flow is detected, the steering device 130 forwards the HTTP request and a portion of the HTTP response to the network controller 140 over the ICAP client interface 404. After receiving the request and the portion of response at the ICAP server interface 406, the flow analyzer 312 of the net-work controller 140 performs a deep flow inspection to deter-mine if the flow is worth bandwidth monitoring and/or user detection. For example, the flow inspection performed by the flow analyzer 312 may determine if the flow indeed contains large or medium object ( e.g., larger than 50 kB), and/or if the source IP address of the flow is from a user or a group of users that are required to be monitored by policies. The flow ana-lyzer 312 may also determine if the flow needs to be opti-mized based on historical flow statistical data.")<br><br>Swenson at [0060] ("If the flow is deemed of interest, the steering device 130 is notified to steer the flow through the network controller 140. This is known as the "continue" working mode for bandwidth monitoring. In the "continue" mode, the network controller 140 interfaces with the steering device 130 to func-tion, on-demand, as a traditional inline network element for flows deemed of interest. Thus, the network controller 140 ingests the network flow for inspection and subsequently forwards the network flow on the network response path. For example, for this particular flow, the origin server 160 responds to the user request by sending video or images over the network link 413 to the steering device 130, which for-wards the video or images to the network controller 140 over a network link 414. After the network controller 140 updates the flow statistics, the video or images are returned to the steering device 130 over a network link 415, which transmits the video or images to the user device 110 over the network link 416.")<br><br>Swenson at [0071] ("After receiving the request, the video optimizer 150 forwards the video HTTP GET requests 622 to the origin server 160 and in return, receives a video file 624 from the origin server 160. The video optimizer 150 transcodes the video file to a format usable by the client device 110 based on network bandwidth available to the user device 110. The optimized video 626 is then transmitted from the video opti-mizer 150 to the steering device 130. In one embodiment, the steering device 130 intercepts the optimized video 626. The steering device 130 will then send an ICAP request to the network controller 140 for inspection. The network controller 140 deems this flow to be monitored and sends |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 45 of 1100

| No. | '111 Patent Claim 4 | Kempf |
|---|---|---|
| | | ICAP response 630. The steering device 130 then allows the flow to go through to the user device 110. The steering device 130 next sends periodic ICAP "counting" updates 632 to the network controller 140 until the flow completes. As such, the client receives the optimized video 626 for substantially real-time playback on an application executing on the user device 110.")<br><br>Swenson at Figure 1 (annotation added)<br><br>  <br>FIG. 1 |
| 4[e] | responsive to receiving the packet, sending the packet, by the network node, to the second entity. | Kempf discloses responsive to receiving the packet, sending the packet, by the network node, to the second entity.<br><br>For example, Kempf discloses sending the packet back through the intended datapath via the network element. A person of ordinary skill would understand that the packet, once sent to the controller for further processing, is then sent back to the network element to be forwarded to its originally intended destination. Thus, at least under the apparent claim |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 46 of 1100

| No. | '111 Patent Claim 4 | Kempf |
|---|---|---|
| | | scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Kempf is found to not meet this limitation, responsive to receiving the packet, sending the packet, by the network node, to the second entit would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>Kempf at [0047] ("The actions that are defined by the specification of OpenFlow 1.0 are Drop, which drops the matching packets; Forward, which forwards the packet to one or all outgoing ports, the incoming physical port itself, the controller via the secure channel, or the local networking stack (if it exists). OpenFlow 1.0 protocol data units (PDU s) are defined with a set of structures specified using the C programming language. Some of the more commonly used messages are: report switch configuration message; modify state messages (in-cluding a modify flow entry message and port modification message); read state messages, where while the system is running, the datapath may be queried about its current state using this message; and send packet message, which is used when the controller wishes to send a packet out through the datapath.")<br><br>Kempf at [0055] ("OpenFlow 1.1 can be utilized to support virtual ports. A virtual port, as used herein, is an "action block" that performs some kind of processing action other than simply forwarding the packet out to a network connection like physi-cal ports do. Examples of a few built-in virtual ports include: ALL, which forwards the port out all ports except for the ingress port and any ports that are marked "Do Not Forward;" CONTROLLER, which encapsulates the packet and sends it to the controller; TABLE, which inserts the packet into the packet processing pipeline by submitting it to the first flow table, this action is only valid in the action set of a packet-out message; and IN_PORT, which sends the packet out the ingress port. In other embodiments, there can also be switched-defined virtual ports.")<br><br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, the VMware NSX System in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 4(e) of Exhibit E-4 renders the claim, including the present limitation, obvious. Below is an example. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 47 of 1100

| No. | '111 Patent Claim 4 | Kempf |
|---|---|---|
| | | For example, Swenson discloses sending the packet, for example a video or image, back to the steering device after the network controller analyzes the packet and updates flow statistics, i.e., sending the packet, by the controller, to the network node. Swenson further discloses the steering device, upon having the packet returned to it, i.e., responsive to receiving the packet, transmitting the packet to the destination entity, for example, the user device or origin server, i.e., sending the packet, by the network node, to the second entity.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0028] ("In contrast to conventional inline TCP throughput monitoring devices that monitor every single data packets transmitted and received, the network controller 140 is an "out-of-band" computer server that interfaces with the steer-ing device 130 to selectively inspect user flows of interest. The network controller 140 may further identify user flows (e.g., among the flows of interest) for optimization. In one embodiment, the network controller 140 may be imple-mented at the steering device 130 to monitor traffic. In other embodiments, the network controller 140 is coupled to and communicates with the steering device 130 for traffic moni-toring and optimization. When queried by the steering device 130, the network controller 140 determines if a given network flow should be ignored, monitored further or optimized. Opti-mization of a flow is often decided at the beginning of the flow because it is rarely possible to switch to optimized content mid-stream once non-optimized content delivery has begun. However, the network controller 140 may determine that existing flows associated with a particular subscriber or other entity should be optimized. In turn, new flows ( e.g., resulting from seek requests in media, new media requests, resume after pause, etc.) determined to be associated with the entity may be optimized. The network controller 140 uses the net-work state as well as historical traffic |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 48 of 1100

| No. | '111 Patent Claim 4 | Kempf |
|---|---|---|
| | | data in its decision for monitoring and optimization. Knowledge on the current net-work state, such as congestion, deems critical when it comes to data optimization.")<br><br>Swenson at [0029] ("As a flow is sent to the network controller 140 for inspection, historical network traffic data stored at the net-work controller 140 may be searched. The historical network traffic data includes information such as subscriber informa-tion, the cell towers to which the user devices attached, rout-ers through which the traffic is passing, geography regions, the backhaul segments, and time-of-day of the flows. For example, in a mobile network, the cell tower to which a user device is attached can be most useful, since it is the location where most congestion occurs due to limited bandwidth and high cost of the radio access network infrastructure. The network controller 140 looks into the historical traffic data for the average of the bandwidth per user at the particular cell tower. The network controller 140 can then estimate the amount of bandwidth or degree of congestion for the new flow based on the historical record.")<br><br>Swenson at [0057] ("The Internet content adaption protocol is a light-weight protocol aimed at executing a simple remote proce-dure call on HTTP messages. ICAP leverages edge-based devices to help deliver value-added services using transparent HTTP proxy caches. Content adaptation refers to performing the particular value added service, such as content manipula-tion or other processing, for the associated HTTP client request/response. ICAP clients pass HTTP messages to ICAP servers for transformation or other processing. In tum, the ICAP server executes its transformation service on the HTTP messages and sends back responses to the ICAP client. At the core of this process is a cache that can proxy all client trans-actions and process them through ICAP servers, which may focus on specific functions, such as ad insertion, virus scan-ning, content translation, language translation, or content fil-tering. ICAP servers, such as those utilized by the network controller 140, handle these tasks to off-load value-added services from network devices including an ICAP client, such as the steering device 130. By offloading value added services from the steering device 130, processing infrastructure (e.g., optimization services and network controllers) may be scaled independent from the steering devices handling raw HTTP throughput.")<br><br>Swenson at [0059] ("In one embodiment, as the steering device 130 monitors network responses, it is looking for flows that match one or more signatures for video and images. When a match-ing flow is detected, the steering device 130 forwards the HTTP request and |

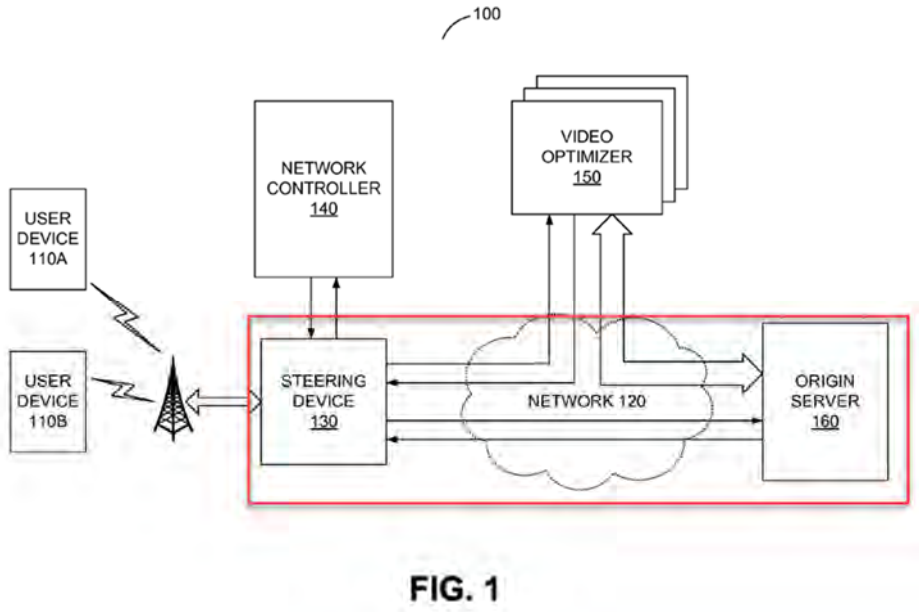


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 49 of 1100

| No. | '111 Patent Claim 4 | Kempf |
|---|---|---|
| | | a portion of the HTTP response to the network controller 140 over the ICAP client interface 404. After receiving the request and the portion of response at the ICAP server interface 406, the flow analyzer 312 of the net-work controller 140 performs a deep flow inspection to deter-mine if the flow is worth bandwidth monitoring and/or user detection. For example, the flow inspection performed by the flow analyzer 312 may determine if the flow indeed contains large or medium object ( e.g., larger than 50 kB), and/or if the source IP address of the flow is from a user or a group of users that are required to be monitored by policies. The flow ana-lyzer 312 may also determine if the flow needs to be opti-mized based on historical flow statistical data.")<br><br>Swenson at [0060] ("If the flow is deemed of interest, the steering device 130 is notified to steer the flow through the network controller 140. This is known as the "continue" working mode for bandwidth monitoring. In the "continue" mode, the network controller 140 interfaces with the steering device 130 to func-tion, on-demand, as a traditional inline network element for flows deemed of interest. Thus, the network controller 140 ingests the network flow for inspection and subsequently forwards the network flow on the network response path. For example, for this particular flow, the origin server 160 responds to the user request by sending video or images over the network link 413 to the steering device 130, which for-wards the video or images to the network controller 140 over a network link 414. After the network controller 140 updates the flow statistics, the video or images are returned to the steering device 130 over a network link 415, which transmits the video or images to the user device 110 over the network link 416.")<br><br>Swenson at [0071] ("After receiving the request, the video optimizer 150 forwards the video HTTP GET requests 622 to the origin server 160 and in return, receives a video file 624 from the origin server 160. The video optimizer 150 transcodes the video file to a format usable by the client device 110 based on network bandwidth available to the user device 110. The optimized video 626 is then transmitted from the video opti-mizer 150 to the steering device 130. In one embodiment, the steering device 130 intercepts the optimized video 626. The steering device 130 will then send an ICAP request to the network controller 140 for inspection. The network controller 140 deems this flow to be monitored and sends ICAP response 630. The steering device 130 then allows the flow to go through to the user device 110. The steering device 130 next sends periodic ICAP "counting" updates 632 to the network controller 140 until the flow completes. As such, the client receives the optimized |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 50 of 1100

| No. | '111 Patent Claim 4 | Kempf |
|---|---|---|
| | | video 626 for substantially real-time playback on an application executing on the user device 110.")<br><br>Swenson at Figure 1 (annotation added)<br><br>  <br>FIG. 1 |

| No. | '111 Patent Claim 5 | Kempf |
|---|---|---|
| 5 | The method according to claim 1, further comprising responsive to the packet satisfying the criterion and to the instruction, sending the packet or a portion thereof, by the | Kempf discloses the method according to claim 1, further comprising responsive to the packet satisfying the criterion and to the instruction, sending the packet or a portion thereof, by the network node, to the controller.<br><br>For example, Kempf discloses sending in response to a flow table match, the packet or packet field to the controller by the network element. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Kempf is found to not meet this limitation, further comprising responsive to |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 51 of 1100

| No. | '111 Patent Claim 5 | Kempf |
|---|---|---|
| | network node, to the controller. | the packet satisfying the criterion and to the instruction, sending the packet or a portion thereof, by the network node, to the controller would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>*See supra* at Claim 1.<br><br>Kempf at [0045] ("FIG. 2 is a diagram illustrating one embodiment of the contents of a flow table entry. The forwarding table 107 is populated with entries consisting of a rule 201 defining matches for fields in packet headers; an action 203 associated to the flow match; and a collection of statistics 205 on the flow. When an incoming packet is received a lookup for a matching rule is made in the flow table 107. If the incoming packet matches a particular rule, the associated action defined in that flow table entry is performed on the packet.")<br><br>Kempf at [0046] ("A rule 201 contains key fields from several headers in the protocol stack, for example source and destination Ethernet MAC addresses, source and destination IP addresses, IP protocol type number, incoming and outgoing TCP or UDP port numbers. To define a flow, all the available matching fields may be used. But it is also possible to restrict the matching rule to a subset of the available fields by using wildcards for the unwanted fields.")<br><br>Kempf at [0047] ("The actions that are defined by the specification of OpenFlow 1.0 are Drop, which drops the matching packets; Forward, which forwards the packet to one or all outgoing ports, the incoming physical port itself, the controller via the secure channel, or the local networking stack (if it exists). OpenFlow 1.0 protocol data units (PDU s) are defined with a set of structures specified using the C programming language. Some of the more commonly used messages are: report switch configuration message; modify state messages (in-cluding a modify flow entry message and port modification message); read state messages, where while the system is running, the datapath may be queried about its current state using this message; and send packet message, which is used when the controller wishes to send a packet out through the datapath.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 52 of 1100

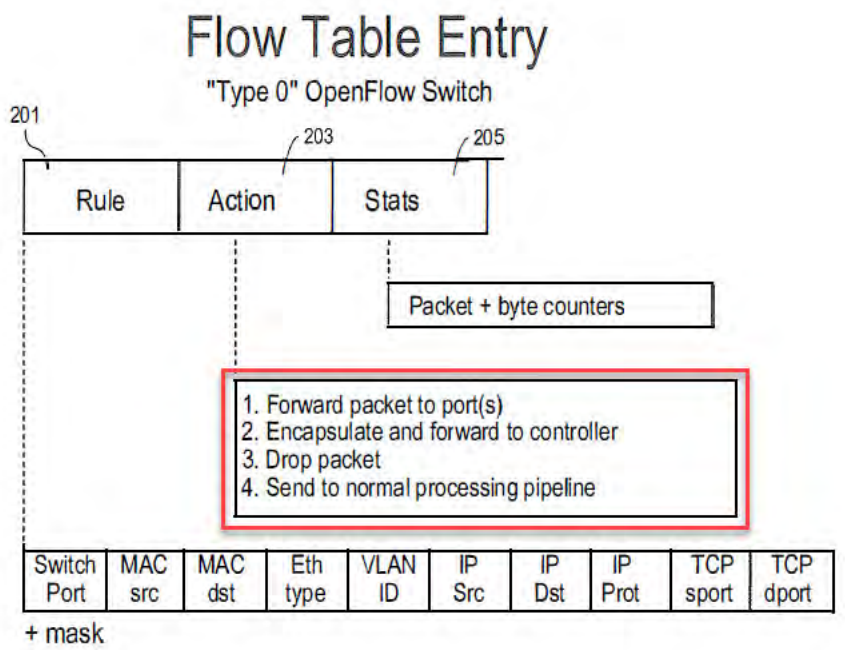
| No. | '111 Patent Claim 5 | Kempf |
|---|---|---|
| | | Kempf at [0050] ("FIG. 4 illustrates one embodiment of the processing of packets through an OpenFlow 1.1 switched packet pro-cessing pipeline. A received packet is compared against each of the flow tables 401. After each flow table match, the actions are accumulated into an action set. If processing requires matching against another flow table, the actions in the matched rule include an action directing processing to the next table in the pipeline. Absent the inclusion of an action in the set to execute all accumulated actions immediately, the actions are executed at the end 403 of the packet processing pipeline. An action allows the writing of data to a metadata register, which is carried along in the packet processing pipe-line like the packet header.")<br><br>Kempf at [0051] ("FIG. 5 is a flowchart of one embodiment of the OpenFlow 1.1 rule matching process. OpenFlow 1.1 contains support for packet tagging. OpenFlow 1.1 allows matching based on header fields and multi-protocol label switching (MPLS) labels. One virtual LAN (VLAN) label and one MPLS label can be matched per table. The rule matching process is initiated with the arrival of a packet to be processed (Block 501 ). Starting at the first table 0 a lookup is performed to determine a match with the received packet (Block 503). If there is no match in this table, then one of a set of default actions is taken (i.e., send packet to controller, drop the packet or continue to next table) (Block 509). If there is a match, then an update to the action set is made along with counters, packet or match set fields and meta data (Block 505). A check is made to determine the next table to process, which can be the next table sequentially or one specified by an action of a matching rule (Block 507). Once all of the tables have been processed, then the resulting action set is executed (Block 511). FIG. 6 is a diagram of the fields, which a matching process can utilize for identifying rules to apply to a packet.")<br><br>Kempf at [0055] ("OpenFlow 1.1 can be utilized to support virtual ports. A virtual port, as used herein, is an "action block" that performs some kind of processing action other than simply forwarding the packet out to a network connection like physi-cal ports do. Examples of a few built-in virtual ports include: ALL, which forwards the port out all ports except for the ingress port and any ports that are marked "Do Not Forward;" CONTROLLER, which encapsulates the packet and sends it to the controller; TABLE, which inserts the packet into the packet processing pipeline by submitting it to the first flow table, this action is only valid in the action set of a packet-out message; and IN_PORT, which sends the packet out the ingress port. In other embodiments, there can also be switched-defined virtual ports.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 53 of 1100

| No. | '111 Patent Claim 5 | Kempf |
|---|---|---|
| | | Kempf at [0083] ("If a packet either needs encapsulation or arrives encapsulated with nonzero header flags, header extensions, and/or the GTP-U packet is not a G-PDU packet (i.e. it is a GTP-U control packet), the processing must proceed via the gateway's slow path (software) control plane. GTP-C and GTP' packets directed to the gateway's IP address are a result of mis-configuration and are in error. They must be sent to the OpenFlow controller, since these packets are handled by the S-GW-C and P-GW-C control plane entities in the cloud computing system or to the billing entity handling GTP' and not the S-GW-D and P-GW-D data plane switches.")<br><br>Kempf at [0091] ("When a packet header matches a rule associated with the virtual port, the GTP TEID is written into the lower 32 bits of the metadata and the packet is directed to the virtual port. The virtual port calculates the hash of the TEID and looks up the tunnel header information in the tunnel header table. If no such tunnel information is present, the packet is forwarded to the controller with an error indication. Other-wise, the virtual port constructs a GTP tunnel header and encapsulates the packet. Any DSCP bits or VLAN priority bits are additionally set in the IP or MAC tunnel headers, and any VLAN tags or MPLS labels are pushed onto the packet. The encapsulated packet is forwarded out the physical port to which the virtual port is bound.)<br><br>Kempf at [0106] ("This encapsulates the packet and sends it to the OpenFlow controller.")<br><br>Kempf at [0133] ("Before returning a result to the PGW-C from the GTP routing update RPC, the OpenFlow controller issues a sequence of OpenFlow messages to the appropriate data plane gateway entity. In the example embodiment, the sequence begins with an OFP _BARRIER_REQUEST to ensure that there are no pending messages that might influ-ence processing of the following messages. Then an OFPT_ FLOW _MOD message is issued, including the of_match structure with GTP extension as the match field and OFPFC_ ADD as the command field. The message specifies actions and instructions, as described above, to establish a flow route for the GTP tunnel that encapsulates and decapsulates the packets through the appropriate virtual port. In addition, immediately following the OFPT_FLOW _MOD message, the OpenFlow controller issues an GTP _ADD_TEID_ TABLE_ENTRY message to the gateways containing the TEID hash table entries for the encapsulation virtual port. As described above, the two OpenFlow messages are followed by an |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 54 of 1100

| No. | '111 Patent Claim 5 | Kempf |
|---|---|---|
| | | OFPT_BARRIER_REQUEST message to force the gateways to process the flow route and TEID hash table update before proceeding.")<br><br>Kempf at Figure 2 (annotation added)<br><br>  <br><br>FIG. 2<br><br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Kempf in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 5 of Exhibit E-4 renders the claim, including the present limitation, obvious. Below is an example. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 55 of 1100

| No. | '111 Patent Claim 5 | Kempf |
|---|---|---|
| | | For example, Copeland discloses sending packets and sampled packet headers to the intrusion detection engine on the monitoring appliance based on matching predetermined values associated with a concern index.<br><br>Copeland at [0067] ("The host servers 130 are directly or indirectly coupled to one or more network devices 135 such as routers or switches that support providing a sampled data stream such as that provided by sFlow. In a typical preferred configuration for the present invention, a monitoring appli-ance 150 operating a flow-based intrusion detection engine 155 is receiving sampled packet headers from one or more network devices 135. The monitoring appliance 150 moni-tors the communications between the host server 130 and other hosts 120, 110 in the attempt to detect intrusion activity.")<br><br>Copeland [0079] ("Large packets tend to be fragmented by networks that cannot handle a large packet size. A 16-bit packet identification is used to reassemble fragmented packets. Three one-bit set of fragmentation flags control whether a packet is or may be fragmented. The 13-bit fragment offset is a sequence number for the 4-byte words in the packet when reassembled. In a series of fragments, the first offset will be zero.")<br><br>Copeland at [0097] ("The described TCP session 300 of FIG. 3 is a generic TCP session in which a network might engage. In accordance with the invention, flow data is collected about the session to help determine if the communication is abnormal. In the preferred embodiment, information such as the total number of packets sent, the total amount of data sent, the session start time and duration, and the TCP flags set in all of the packets, are collected, stored in the database 160, and analyzed to determine if the communication was suspicious. If a communication is deemed suspicious, i.e. it meets predetermined criteria, a predetermined concern index value associated with a determined category of suspicious activity is added to the cumulated CI value associated with the host that made the communication.")<br><br>Copeland at [0120] ("The sampled packet headers sent from the sFlow agent are captured and processed by the sample packet collector 505 in order to create a "Packet Data" data struc-ture that includes the sFlow agent source of the packets, the header of the sampled |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 56 of 1100

| No. | ’111 Patent Claim 5 | Kempf |
|---|---|---|
| | | packets, and other information avail-able from the sFlow data stream that may be important. For<br>example, one data field that is optionally available pr vides the username of the user using the computer at the time of the communications. This information is extremely useful in some environments subject to regulatory requirements and monitoring of the communications on the network. In this case the username will be stored as "supplementary infor-mation" for auditing purposes in the flow data. Other infor-mation, including the sampling device and the physical port on which the communications was detected may also be retained for other uses such as mitigation, where a host may be removed from the network.”)<br><br>Copeland at [0126]-[0129] (“If a particular packet 101 being processed by the packet classifier 510 matches a particular entry or record in the flow data structure 162, data from that particular packet 101 is used to update the statistics in the corresponding flow data structure record. A packet 101 is considered to match to a flow data structure record if both IP numbers match and the source of the sampled packet matches and:<br><br>(1) both port numbers match and no port is marked as the "server" port, or<br>(2) the port number previously marked as the "server" port matches, or<br>(3) one of the port numbers matches, but the other does not, and the neither port number has been marked as the server port (in this case the matching port number is marked as the "server" port).”)<br><br>Copeland at [0144] (“Concern index (CI) values calculated from packet anomalies also add to a host's accumulated concern index value. Table II of FIG. 7 shows one scheme for assigning concern index values due to other events revealed by the flow analysis. For example, there are many combinations of TCP flag bits that are rarely or never seen in valid TCP connections. When the packet classifier thread 510 recog-nizes one of these combinations, it directly adds a predeter-mined value to the sending host's accumulated concern index value. When the packet classifier thread 510 searches along the flow linked-list (i.e. flow data 162) for a match to the current packet 101, it keeps count of the number of flows active with matching IP addresses but no matching port number. If this number exceeds a predetermined threshold value (e.g., 4) and is greater than the previous number |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 57 of 1100

| No. | '111 Patent Claim 5 | Kempf |
|---|---|---|
| | | noticed, CI is added for an amount corresponding to a "port scan." A bit in the host record is set to indicate that the host has received CI for "port scanning."")<br><br>Copeland at [0150] ("A preferred hardware configuration 800 of an embodiment that executes the functions of the above-described flow-based engine is described in reference to FIG. 8. FIG. 8 illustrates a typically hardware configuration 800 for a network intrusion detection system. A monitoring appliance 150 serves as a pass-by filter of network traffic. A network device 135, such as a router or switch supporting sFlow provides the location for connecting the monitoring appliance 150 to the network 899 for monitoring the network traffic.")<br><br>Copeland at [0159]-[0162] ("A packet 101 is considered to match to a flow data structure record if both IP numbers match and the source of the sampled data matches and:<br><br>(a). both port numbers match and no port is marked as the "server" port, or<br>(b). the port number previously marked as the "server" port matches, or<br>(c). one of the port numbers matches, but the other does not, and the neither port number has been marked as the server port (in this case the matching port number is marked as the "server" port).") |

| No. | '111 Patent Claim 6 | Kempf |
|---|---|---|
| 6 | The method according to claim 5, further comprising storing the received packet or a portion thereof, by the controller, in a memory. | Kempf discloses the method according to claim 5, further comprising storing the received packet or a portion thereof, by the controller, in a memory.<br><br>For example, Kempf discloses storing the packets and packet fields in the controller, and further updating rules based on that stored information. A person of ordinary skill in the art would understand that updating the rules sent to network elements by the controller is based on storing the packet information informing the rule update.<br><br>*See supra* at Claim 5. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 58 of 1100

| No. | '111 Patent Claim 6 | Kempf |
|---|---|---|
| | | Kempf at [0051] ("FIG. 5 is a flowchart of one embodiment of the OpenFlow 1.1 rule matching process. OpenFlow 1.1 contains support for packet tagging. OpenFlow 1.1 allows matching based on header fields and multi-protocol label switching (MPLS) labels. One virtual LAN (VLAN) label and one MPLS label can be matched per table. The rule matching process is initiated with the arrival of a packet to be processed (Block 501 ). Starting at the first table 0 a lookup is performed to determine a match with the received packet (Block 503). If there is no match in this table, then one of a set of default actions is taken (i.e., send packet to controller, drop the packet or continue to next table) (Block 509). If there is a match, then an update to the action set is made along with counters, packet or match set fields and meta data (Block 505). A check is made to determine the next table to process, which can be the next table sequentially or one specified by an action of a matching rule (Block 507). Once all of the tables have been processed, then the resulting action set is executed (Block 511). FIG. 6 is a diagram of the fields, which a matching process can utilize for identifying rules to apply to a packet.")<br><br>Kempf at [0065] ("A cloud computing system can be composed of any number of computing devices having any range of capabili-ties (e.g., processing power or storage capacity). The cloud computing system can be a private or public system. The computing devices can be in communication with one another across any communication system or network. A cloud com-puting system can support a single cloud or service or any number of discrete clouds or services. Services, applications and similar programs can be virtualized or executed as stan-dard code. In one embodiment, cloud computing systems can support web services applications. Web services applications consist of a load balancing front end that dispatches requests to a pool of Web servers. The requests originate from appli-cations on remote machines on the Internet and therefore the security and privacy requirements are much looser than for applications in a private corporate network.")<br><br>Kempf at [0134] ("Prior to returning from the GTP routing update RPC, the OpenFlow controller also issues GTP flow routing updates to any GTP extended OpenFlow Switches (GxOFSs) that need to be involved in customized GTP flow routing. The messages in these updates consist of an OFP _BARRIER_ REQUEST followed by an OFPT_FLOW _MOD message containing the ofp_match structure with GTP extension for the new GTP flow as the match field and OFPFC_ADD as the command field, and the actions and instructions described above for customized GTP flow routing. A final OFP _BAR-RIER_REQUEST |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 59 of 1100

| No. | '111 Patent Claim 6 | Kempf |
|---|---|---|
| | | forces the switch to process the change before responding. The flow routes on any GxOFSs are installed after installing the GTP tunnel endpoint route on the SGW-D and prior to installing the GTP tunnel endpoint route on the PGW-D, as illustrated in FIG. 19. The OpenFlow controller does not respond to the PGW-C RPC until all flow routing updates have been accomplished.") |

| No. | '111 Patent Claim 7 | Kempf |
|---|---|---|
| 7 | The method according to claim 5, further comprising responsive to the packet satisfying the criterion and to the instruction, sending a portion of the packet, by the network node, to the controller. | Kempf discloses the method according to claim 5, further comprising responsive to the packet satisfying the criterion and to the instruction, sending a portion of the packet, by the network node, to the controller.<br><br>For example, Kempf discloses sending in response to a flow table match a packet field to the controller by the network element. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Kempf is found to not meet this limitation, further comprising responsive to the packet satisfying the criterion and to instruction, sending a portion of the packet, by the network node, to the controller would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>*See supra* at Claim 5.<br><br>Kempf at [0050] ("FIG. 4 illustrates one embodiment of the processing of packets through an OpenFlow 1.1 switched packet pro-cessing pipeline. A received packet is compared against each of the flow tables 401. After each flow table match, the actions are accumulated into an action set. If processing requires matching against another flow table, the actions in the matched rule include an action directing processing to the next table in the pipeline. Absent the inclusion of an action in the set to execute all accumulated actions immediately, the actions are executed at the end 403 of the packet processing pipeline. An action allows the writing of data to a metadata register, which is carried along in the packet processing pipe-line like the packet header.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 60 of 1100

| No. | '111 Patent Claim 7 | Kempf |
|---|---|---|
| | | Kempf at [0055] ("OpenFlow 1.1 can be utilized to support virtual ports. A virtual port, as used herein, is an "action block" that performs some kind of processing action other than simply forwarding the packet out to a network connection like physi-cal ports do. Examples of a few built-in virtual ports include: ALL, which forwards the port out all ports except for the ingress port and any ports that are marked "Do Not Forward;" CONTROLLER, which encapsulates the packet and sends it to the controller; TABLE, which inserts the packet into the packet processing pipeline by submitting it to the first flow table, this action is only valid in the action set of a packet-out message; and IN_PORT, which sends the packet out the ingress port. In other embodiments, there can also be switched-defined virtual ports.")<br><br>Kempf at [0083] ("If a packet either needs encapsulation or arrives encapsulated with nonzero header flags, header extensions, and/or the GTP-U packet is not a G-PDU packet (i.e. it is a GTP-U control packet), the processing must proceed via the gateway's slow path (software) control plane. GTP-C and GTP' packets directed to the gateway's IP address are a result of mis-configuration and are in error. They must be sent to the OpenFlow controller, since these packets are handled by the S-GW-C and P-GW-C control plane entities in the cloud computing system or to the billing entity handling GTP' and not the S-GW-D and P-GW-D data plane switches.")<br><br>Kempf at [0087] ("In one embodiment, slow path support for GTP is implemented with an OpenFlow gateway switch. An Open-Flow mobile gateway switch also contains support on the software control plane for slow path packet processing. This path is taken by G-PDU (message type 255) packets with nonzero header fields or extension headers, and user data plane packets requiring encapsulation with such fields or addition of extension headers, and by G TP-U control packets. For this purpose, the switch supports three local ports in the software control plane: LOCAL_GTP _CONTROL-the switch fast path forwards GTP encapsulated packets directed to the gateway IP address that contain GTP-U control mes-sages and the local switch software control plane initiates local control plane actions depending on the GTP-U control message; LOCAL_GTP _U_DECAP-the switch fast path forwards G-PDU packets to this port that have nonzero header fields or extension headers (i.e. E!=0, S!=0, or PN!=0). These packets require specialized handling. The local switch software slow path processes the packets and performs the specialized handling; and LOCAL_GTP _U_ENCAP-the switch fast path forwards user data plane packets to this port that require encapsulation in a GTP tunnel with nonzero header fields or extension headers |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 61 of 1100

| No. | '111 Patent Claim 7 | Kempf |
|---|---|---|
| | | (i.e. E!=0, S!=0, or PN!=0). These packets require specialized handling. The local switch software slow path encapsulates the packets and performs the specialized handling. In addition to forwarding the packet, the switch fast path makes the OpenFlow metadata field avail-able to the slow path software.")<br><br>Kempf at [0091] ("When a packet header matches a rule associated with the virtual port, the GTP TEID is written into the lower 32 bits of the metadata and the packet is directed to the virtual port. The virtual port calculates the hash of the TEID and looks up the tunnel header information in the tunnel header table. If no such tunnel information is present, the packet is forwarded to the controller with an error indication. Other-wise, the virtual port constructs a GTP tunnel header and encapsulates the packet. Any DSCP bits or VLAN priority bits are additionally set in the IP or MAC tunnel headers, and any VLAN tags or MPLS labels are pushed onto the packet. The encapsulated packet is forwarded out the physical port to which the virtual port is bound.")<br><br>Kempf at [0092] ("In one embodiment, the system implements a GTP fast path decapsulation virtual port. When requested by the S-GW and P-GW control plane software running in the cloud computing system, the gateway switch installs rules and actions for routing GTP encapsulated packets out of GTP tunnels. The rules match the GTP header flags and the GTP TEID for the packet, in the modified OpenFlow flow table shown in FIG. 17 as follows: the IP destination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); the UDP destination port is the GTP-U destination port (2152); and the header fields and message type field is wildcarded with the flag 0XFFF0 and the upper two bytes of the field match the G-PDU message type (255) while the lower two bytes match 0x30, i.e. the packet is a GTP packet not a GTP' packet and the version number is 1.")<br><br>Kempf at [0098] ("The header flags and message type fields for the three rules are wildcarded with the following bitmasks and match as follows: bitmask 0xFFF4 and the upper two bytes match the G-PDU message type (255) while the lower two bytes are Ox34, indicating that the version number is 1, the packet is a GTP packet, and there is an extension header present; bitmask 0xFFFF2 and the upper two bytes match the G-PDU message type (255) while the lower two bytes are 0x32, indicating that the version number is 1, the packet is a GTP packet, and there is a sequence number bitmask 0xFF0l and the upper two bytes |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 62 of 1100

| No. | '111 Patent Claim 7 | Kempf |
|---|---|---|
| | | match the G-PDU message type (255) while the lower two bytes are 0x31, indicating that the version number is 1, the packet is a GTP packet, and a N-PDU is present.")<br><br>Kempf at [0101] ("In one embodiment, the system implements han-dling of user data plane packets requiring GTP-U encapsula-tion with extension headers, sequence numbers, and N-PDU numbers. User data plane packets that require extension head-ers, sequence numbers, or N-PDU numbers during GTP encapsulation require special handling by the software slow path. For these packets, the OpenFlow controller programs a rule matching the 4 tuple: IP source address; IP destination address; UDP/TCP/SCTP source port; and UDP/TCP/SCTP destination port. The instructions for matching packets are:<br><br>Write-Metadata ( GTP-TEID, 0x FFFFFFFF)<br>Apply-Actions (Set-Output-Port LOCAL_GTP _U_ENCAP )")<br><br>Kempf at [0106] ("This encapsulates the packet and sends it to the OpenFlow controller.")<br><br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Kempf in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 7 of Exhibit E-4 renders the claim, including the present limitation, obvious. Below is an example.<br><br>For example, Copeland discloses sending packets and sampled packet headers to the intrusion detection engine on the monitoring appliance based on matching predetermined values associated with a concern index.<br><br>Copeland at [0067] ("The host servers 130 are directly or indirectly coupled to one or more network devices 135 such as routers or switches that support providing a sampled data stream such as that provided by sFlow. In a typical preferred configuration for the present invention, a monitoring appli-ance 150 operating a flow-based intrusion detection engine 155 is receiving sampled packet headers from one or more network devices 135. The monitoring appliance 150 moni-tors the communications between the host server 130 and other hosts 120, 110 in the attempt to detect intrusion activity.") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 63 of 1100

| No. | '111 Patent Claim 7 | Kempf |
|---|---|---|
| | | Copeland [0079] ("Large packets tend to be fragmented by networks that cannot handle a large packet size. A 16-bit packet identification is used to reassemble fragmented packets. Three one-bit set of fragmentation flags control whether a packet is or may be fragmented. The 13-bit fragment offset is a sequence number for the 4-byte words in the packet when reassembled. In a series of fragments, the first offset will be zero.")<br><br>Copeland at [0097] ("The described TCP session 300 of FIG. 3 is a generic TCP session in which a network might engage. In accordance with the invention, flow data is collected about the session to help determine if the communication is abnormal. In the preferred embodiment, information such as the total number of packets sent, the total amount of data sent, the session start time and duration, and the TCP flags set in all of the packets, are collected, stored in the database 160, and analyzed to determine if the communication was suspicious. If a communication is deemed suspicious, i.e. it meets predetermined criteria, a predetermined concern index value associated with a determined category of suspicious activity is added to the cumulated CI value associated with the host that made the communication.")<br><br>Copeland at [0120] ("The sampled packet headers sent from the sFlow agent are captured and processed by the sample packet collector 505 in order to create a "Packet Data" data struc-ture that includes the sFlow agent source of the packets, the header of the sampled packets, and other information avail-able from the sFlow data stream that may be important. For<br>example, one data field that is optionally available pr vides the username of the user using the computer at the time of the communications. This information is extremely useful in some environments subject to regulatory requirements and monitoring of the communications on the network. In this case the username will be stored as "supplementary infor-mation" for auditing purposes in the flow data. Other infor-mation, including the sampling device and the physical port on which the communications was detected may also be retained for other uses such as mitigation, where a host may be removed from the network.")<br><br>Copeland at [0126]-[0129] ("If a particular packet 101 being processed by the packet classifier 510 matches a particular entry or record in the flow data structure 162, data from |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 64 of 1100

| No. | '111 Patent Claim 7 | Kempf |
|---|---|---|
| | | that particular packet 101 is used to update the statistics in the corresponding flow data structure record. A packet 101 is considered to match to a flow data structure record if both IP numbers match and the source of the sampled packet matches and:<br><br>(1) both port numbers match and no port is marked as the "server" port, or<br>(2) the port number previously marked as the "server" port matches, or<br>(3) one of the port numbers matches, but the other does not, and the neither port number has been marked as the server port (in this case the matching port number is marked as the "server" port).")<br><br>Copeland at [0144] ("Concern index (CI) values calculated from packet anomalies also add to a host's accumulated concern index value. Table II of FIG. 7 shows one scheme for assigning concern index values due to other events revealed by the flow analysis. For example, there are many combinations of TCP flag bits that are rarely or never seen in valid TCP connections. When the packet classifier thread 510 recog-nizes one of these combinations, it directly adds a predeter-mined value to the sending host's accumulated concern index value. When the packet classifier thread 510 searches along the flow linked-list (i.e. flow data 162) for a match to the current packet 101, it keeps count of the number of flows active with matching IP addresses but no matching port number. If this number exceeds a predetermined threshold value (e.g., 4) and is greater than the previous number noticed, CI is added for an amount corresponding to a "port scan." A bit in the host record is set to indicate that the host has received CI for "port scanning."")<br><br>Copeland at [0150] ("A preferred hardware configuration 800 of an embodiment that executes the functions of the above-described flow-based engine is described in reference to FIG. 8. FIG. 8 illustrates a typically hardware configuration 800 for a network intrusion detection system. A monitoring appliance 150 serves as a pass-by filter of network traffic. A network device 135, such as a router or switch supporting sFlow provides the location for connecting the monitoring appliance 150 to the network 899 for monitoring the network traffic.")<br><br>Copeland at [0159]-[0162] ("A packet 101 is considered to match to a flow data structure record if both IP numbers match and the source of the sampled data matches and: |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 65 of 1100

| No. | '111 Patent Claim 7 | Kempf |
|---|---|---|
| | | (a). both port numbers match and no port is marked as the "server" port, or<br>(b). the port number previously marked as the "server" port matches, or<br>(c). one of the port numbers matches, but the other does not, and the neither port number has been marked as the server port (in this case the matching port number is marked as the "server" port).") |

| No. | '111 Patent Claim 8 | Kempf |
|---|---|---|
| 8[a] | The method according to claim 7, wherein the portion of the packet consists of multiple consecutive bytes, and | Kempf discloses the method according to claim 7, wherein the portion of the packet consists of multiple consecutive bytes.<br><br>For example, Kempf discloses consecutive bytes of a packet header field.<br><br>*See supra* at Claim 7.<br><br>Kempf at [0081] ("In one embodiment, OpenFlow is modified to pro-vide rules for GTP TEID Routing. FIG. 17 is a diagram of one embodiment of the OpenFlow flow table modification for GTP TEID routing. An OpenFlow switch that supports TEID routing matches on the 2 byte (16 bit) collection of header fields and the 4 byte (32 bit) GTP TEID, in addition to other OpenFlow header fields, in at least one flow table ( e.g., the first flow table). The GTP TEID flag can be wildcarded (i.e. matches are "don't care"). In one embodiment, the EPC pro-tocols do not assign any meaning to TEIDs other than as an endpoint identifier for tunnels, like ports in standard UDP/ TCP transport protocols. In other embodiments, the TEIDs can have a correlated meaning or semantics. The GTP header flags field can also be wildcarded, this can be partially matched by combining the following bitmasks: 0xFF00- Match the Message Type field; 0xe0-Match the Version field; 0xl0-Match the PT field; 0x04-Match the E field; 0x02- Match the S field; and 0x0l-Match the PN field.")<br><br>Kempf at [0083] ("If a packet either needs encapsulation or arrives encapsulated with nonzero header flags, header extensions, and/or the GTP-U packet is not a G-PDU packet (i.e. it is a GTP-U control packet), the processing must proceed via the gateway's slow path (software) control plane. GTP-C and GTP' packets directed to the gateway's IP address are a |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 66 of 1100

| No. | '111 Patent Claim 8 | Kempf |
|---|---|---|
| | | result of mis-configuration and are in error. They must be sent to the OpenFlow controller, since these packets are handled by the S-GW-C and P-GW-C control plane entities in the cloud computing system or to the billing entity handling GTP' and not the S-GW-D and P-GW-D data plane switches.")<br><br>Kempf at [0087] ("In one embodiment, slow path support for GTP is implemented with an OpenFlow gateway switch. An Open-Flow mobile gateway switch also contains support on the software control plane for slow path packet processing. This path is taken by G-PDU (message type 255) packets with nonzero header fields or extension headers, and user data plane packets requiring encapsulation with such fields or addition of extension headers, and by G TP-U control packets. For this purpose, the switch supports three local ports in the software control plane: LOCAL_GTP _CONTROL-the switch fast path forwards GTP encapsulated packets directed to the gateway IP address that contain GTP-U control mes-sages and the local switch software control plane initiates local control plane actions depending on the GTP-U control message; LOCAL_GTP _U_DECAP-the switch fast path forwards G-PDU packets to this port that have nonzero header fields or extension headers (i.e. E!=0, S!=0, or PN!=0). These packets require specialized handling. The local switch software slow path processes the packets and performs the specialized handling; and LOCAL_GTP _U_ENCAP-the switch fast path forwards user data plane packets to this port that require encapsulation in a GTP tunnel with nonzero header fields or extension headers (i.e. E!=0, S!=0, or PN!=0). These packets require specialized handling. The local switch software slow path encapsulates the packets and performs the specialized handling. In addition to forwarding the packet, the switch fast path makes the OpenFlow metadata field avail-able to the slow path software.")<br><br>Kempf at [0091] ("When a packet header matches a rule associated with the virtual port, the GTP TEID is written into the lower 32 bits of the metadata and the packet is directed to the virtual port. The virtual port calculates the hash of the TEID and looks up the tunnel header information in the tunnel header table. If no such tunnel information is present, the packet is forwarded to the controller with an error indication. Other-wise, the virtual port constructs a GTP tunnel header and encapsulates the packet. Any DSCP bits or VLAN priority bits are additionally set in the IP or MAC tunnel headers, and any VLAN tags or MPLS labels are pushed onto the packet. The encapsulated packet is forwarded out the physical port to which the virtual port is bound.") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 67 of 1100

| No. | '111 Patent Claim 8 | Kempf |
|---|---|---|
| | | Kempf at [0092] ("In one embodiment, the system implements a GTP fast path decapsulation virtual port. When requested by the S-GW and P-GW control plane software running in the cloud computing system, the gateway switch installs rules and actions for routing GTP encapsulated packets out of GTP tunnels. The rules match the GTP header flags and the GTP TEID for the packet, in the modified OpenFlow flow table shown in FIG. 17 as follows: the IP destination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); the UDP destination port is the GTP-U destination port (2152); and the header fields and message type field is wildcarded with the flag 0XFFF0 and the upper two bytes of the field match the G-PDU message type (255) while the lower two bytes match 0x30, i.e. the packet is a GTP packet not a GTP' packet and the version number is 1.")<br><br>Kempf at [0098] ("The header flags and message type fields for the three rules are wildcarded with the following bitmasks and match as follows: bitmask 0xFFF4 and the upper two bytes match the G-PDU message type (255) while the lower two bytes are Ox34, indicating that the version number is 1, the packet is a GTP packet, and there is an extension header present; bitmask 0xFFFF2 and the upper two bytes match the G-PDU message type (255) while the lower two bytes are 0x32, indicating that the version number is 1, the packet is a GTP packet, and there is a sequence number bitmask 0xFF0l and the upper two bytes match the G-PDU message type (255) while the lower two bytes are 0x31, indicating that the version number is 1, the packet is a GTP packet, and a N-PDU is present.")<br><br>Kempf at [0101] ("In one embodiment, the system implements han-dling of user data plane packets requiring GTP-U encapsula-tion with extension headers, sequence numbers, and N-PDU numbers. User data plane packets that require extension head-ers, sequence numbers, or N-PDU numbers during GTP encapsulation require special handling by the software slow path. For these packets, the OpenFlow controller programs a rule matching the 4 tuple: IP source address; IP destination address; UDP/TCP/SCTP source port; and UDP/TCP/SCTP destination port. The instructions for matching packets are:<br><br>Write-Metadata ( GTP-TEID, 0x FFFFFFFF)<br>Apply-Actions (Set-Output-Port LOCAL_GTP _U_ENCAP )") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 68 of 1100

| No. | '111 Patent Claim 8 | Kempf |
|---|---|---|
| 8[b] | wherein the instruction comprises identification of the consecutive bytes in the packet. | Kempf discloses wherein the instruction comprises identification of the consecutive bytes in the packet.<br><br>For example, Kempf discloses rules in which the flow table includes matching to the consecutive bytes of a packet header.<br><br>Kempf at [0081] ("In one embodiment, OpenFlow is modified to pro-vide rules for GTP TEID Routing. FIG. 17 is a diagram of one embodiment of the OpenFlow flow table modification for GTP TEID routing. An OpenFlow switch that supports TEID routing matches on the 2 byte (16 bit) collection of header fields and the 4 byte (32 bit) GTP TEID, in addition to other OpenFlow header fields, in at least one flow table ( e.g., the first flow table). The GTP TEID flag can be wildcarded (i.e. matches are "don't care"). In one embodiment, the EPC pro-tocols do not assign any meaning to TEIDs other than as an endpoint identifier for tunnels, like ports in standard UDP/ TCP transport protocols. In other embodiments, the TEIDs can have a correlated meaning or semantics. The GTP header flags field can also be wildcarded, this can be partially matched by combining the following bitmasks: 0xFF00- Match the Message Type field; 0xe0-Match the Version field; 0xl0-Match the PT field; 0x04-Match the E field; 0x02- Match the S field; and 0x0l-Match the PN field.")<br><br>Kempf at [0083] ("If a packet either needs encapsulation or arrives encapsulated with nonzero header flags, header extensions, and/or the GTP-U packet is not a G-PDU packet (i.e. it is a GTP-U control packet), the processing must proceed via the gateway's slow path (software) control plane. GTP-C and GTP' packets directed to the gateway's IP address are a result of mis-configuration and are in error. They must be sent to the OpenFlow controller, since these packets are handled by the S-GW-C and P-GW-C control plane entities in the cloud computing system or to the billing entity handling GTP' and not the S-GW-D and P-GW-D data plane switches.")<br><br>Kempf at [0087] ("In one embodiment, slow path support for GTP is implemented with an OpenFlow gateway switch. An Open-Flow mobile gateway switch also contains support on the software control plane for slow path packet processing. This path is taken by G-PDU (message type 255) packets with nonzero header fields or extension headers, and user data plane packets requiring encapsulation with such fields or addition of extension headers, and |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 69 of 1100

| No. | '111 Patent Claim 8 | Kempf |
|---|---|---|
| | | by G TP-U control packets. For this purpose, the switch supports three local ports in the software control plane: LOCAL_GTP _CONTROL-the switch fast path forwards GTP encapsulated packets directed to the gateway IP address that contain GTP-U control mes-sages and the local switch software control plane initiates local control plane actions depending on the GTP-U control message; LOCAL_GTP _U_DECAP-the switch fast path forwards G-PDU packets to this port that have nonzero header fields or extension headers (i.e. E!=0, S!=0, or PN!=0). These packets require specialized handling. The local switch software slow path processes the packets and performs the specialized handling; and LOCAL_GTP _U_ENCAP-the switch fast path forwards user data plane packets to this port that require encapsulation in a GTP tunnel with nonzero header fields or extension headers (i.e. E!=0, S!=0, or PN!=0). These packets require specialized handling. The local switch software slow path encapsulates the packets and performs the specialized handling. In addition to forwarding the packet, the switch fast path makes the OpenFlow metadata field avail-able to the slow path software.")<br><br>Kempf at [0091] ("When a packet header matches a rule associated with the virtual port, the GTP TEID is written into the lower 32 bits of the metadata and the packet is directed to the virtual port. The virtual port calculates the hash of the TEID and looks up the tunnel header information in the tunnel header table. If no such tunnel information is present, the packet is forwarded to the controller with an error indication. Other-wise, the virtual port constructs a GTP tunnel header and encapsulates the packet. Any DSCP bits or VLAN priority bits are additionally set in the IP or MAC tunnel headers, and any VLAN tags or MPLS labels are pushed onto the packet. The encapsulated packet is forwarded out the physical port to which the virtual port is bound.")<br><br>Kempf at [0092] ("In one embodiment, the system implements a GTP fast path decapsulation virtual port. When requested by the S-GW and P-GW control plane software running in the cloud computing system, the gateway switch installs rules and actions for routing GTP encapsulated packets out of GTP tunnels. The rules match the GTP header flags and the GTP TEID for the packet, in the modified OpenFlow flow table shown in FIG. 17 as follows: the IP destination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); the UDP destination port is the GTP-U destination port (2152); and the header fields and message type field is wildcarded with the flag 0XFFF0 and the upper two bytes of the field match the G-PDU message type (255) |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 70 of 1100

| No. | '111 Patent Claim 8 | Kempf |
|---|---|---|
| | | while the lower two bytes match 0x30, i.e. the packet is a GTP packet not a GTP' packet and the version number is 1.")<br><br>Kempf at [0098] ("The header flags and message type fields for the three rules are wildcarded with the following bitmasks and match as follows: bitmask 0xFFF4 and the upper two bytes match the G-PDU message type (255) while the lower two bytes are Ox34, indicating that the version number is 1, the packet is a GTP packet, and there is an extension header present; bitmask 0xFFFF2 and the upper two bytes match the G-PDU message type (255) while the lower two bytes are 0x32, indicating that the version number is 1, the packet is a GTP packet, and there is a sequence number bitmask 0xFF0l and the upper two bytes match the G-PDU message type (255) while the lower two bytes are 0x31, indicating that the version number is 1, the packet is a GTP packet, and a N-PDU is present.")<br><br>Kempf at [0101] ("In one embodiment, the system implements han-dling of user data plane packets requiring GTP-U encapsula-tion with extension headers, sequence numbers, and N-PDU numbers. User data plane packets that require extension head-ers, sequence numbers, or N-PDU numbers during GTP encapsulation require special handling by the software slow path. For these packets, the OpenFlow controller programs a rule matching the 4 tuple: IP source address; IP destination address; UDP/TCP/SCTP source port; and UDP/TCP/SCTP destination port. The instructions for matching packets are:<br><br>Write-Metadata ( GTP-TEID, 0x FFFFFFFF)<br>Apply-Actions (Set-Output-Port LOCAL_GTP _U_ENCAP )") |

| No. | '111 Patent Claim 9 | Kempf |
|---|---|---|
| 9 | The method according to claim 5, further comprising responsive to receiving the packet, analyzing the packet, by the controller. | Kempf discloses the method according to claim 5, further comprising responsive to receiving the packet, analyzing the packet, by the controller.<br><br>For example, Kempf discloses further processing by the controller in response a packet flow match indicating a packet either needs encapsulation or arrives encapsulated with nonzero header flags, header extensions, and/or the GTP-U packet is not a G-PDU packet (i.e. it is a GTP-U control packet). |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 71 of 1100

| No. | '111 Patent Claim 9 | Kempf |
|---|---|---|
| | | *See supra* at Claim 5.<br><br>Kempf at [0037] ("The EPC architecture also contains little flexibility for specialized treatment of user flows. Though the architec-ture does provide support for establishing quality of service (QoS), other sorts of data management are not available. For example services involving middle boxes, such as specialized deep packet inspection or interaction with local data caching and processing resources that might be utilized for transcod-ing or augmented reality applications, is difficult to support with the current EPC architecture. Almost all such applica-tions require the packet flows to exit through the PDN Gate-way, thereby being de-tunnelled from GTP, and to be pro-cessed within the wired network.")<br><br>Kempf at [0074] ("The operation of the EPC cloud computer system as follows. The UE 1317, E-NodeB 1317, S-GW-C 1307, and P-GW-C signal 1307 to the MME, PCRF, and HSS 1307 using the standard EPC protocols, to establish, modify, and delete bearers and GTP tunnels. This signaling triggers pro-cedure calls with the OpenFlow controller to modify the routing in the EPC as requested. The OpenFlow controller configures the standard OpenFlow switches, the Openflow S-GW-D 1315, and P-GW-D 1311 with flow rules and actions to enable the routing requested by the control plane entities. Details of this configuration are described in further detail herein below.")<br><br>Kempf at [0083] ("If a packet either needs encapsulation or arrives encapsulated with nonzero header flags, header extensions, and/or the GTP-U packet is not a G-PDU packet (i.e. it is a GTP-U control packet), the processing must proceed via the gateway's slow path (software) control plane. GTP-C and GTP' packets directed to the gateway's IP address are a result of mis-configuration and are in error. They must be sent to the OpenFlow controller, since these packets are handled by the S-GW-C and P-GW-C control plane entities in the cloud computing system or to the billing entity handling GTP' and not the S-GW-D and P-GW-D data plane switches.")<br><br>Kempf at [0084] ("GTP virtual ports are configured from the Open-Flow controller using a configuration protocol. The details of the configuration protocol are switch-dependent. The con-figuration protocol must support messages that perform following functions: allow the controller to query for and return an indication whether the switch supports GTP fast path virtual ports and what virtual port numbers are used for fast path and slow path GTP-U |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 72 of 1100

| No. | '111 Patent Claim 9 | Kempf |
|---|---|---|
| | | processing; and allow the controller to instantiate a GTP-U fast path virtual port within a switch datapath for use in the OpenFlow table set-output-port action. The configuration command must be run in a transaction so that, when the results of the action are reported back to the controller, either a GTP-U fast path virtual port for the requested datapath has been instantiated or an error has returned indicating why the request could not be honored. The command also allows the OpenFlow controller to bind a GTP-U virtual port to a physical port. For decapsulation virtual ports, the physical port is an input port. For encapsu-lation virtual ports, the physical port is an output port.")<br><br>Kempf at [0091] ("When a packet header matches a rule associated with the virtual port, the GTP TEID is written into the lower 32 bits of the metadata and the packet is directed to the virtual port. The virtual port calculates the hash of the TEID and looks up the tunnel header information in the tunnel header table. If no such tunnel information is present, the packet is forwarded to the controller with an error indication. Other-wise, the virtual port constructs a GTP tunnel header and encapsulates the packet. Any DSCP bits or VLAN priority bits are additionally set in the IP or MAC tunnel headers, and any VLAN tags or MPLS labels are pushed onto the packet. The encapsulated packet is forwarded out the physical port to which the virtual port is bound.")<br><br>Kempf at [0104] ("In one embodiment, the system implements han-dling of GTP-C and GTP' control packets. Any GTP-C and GTP' control packets that are directed to IP addresses on a gateway switch are in error. These packets need to be handled by the S-GW-C, P-GW-C, and GTP' protocol entities in the cloud computing system, not the S-GW-D and P-GW-D enti-ties in the switches. To catch such packets, the OpenFlow controller must program the switch with the following two rules: the IP destination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); for one rule, the UDP destination port is the GTP-U destination port (2152), for the other, the UDP destination port is the GTP-C destination port (2123); the GTP header flags and message type fields are wildcarded.")<br><br>Kempf at [0106] ("This encapsulates the packet and sends it to the OpenFlow controller.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 73 of 1100

| No. | '111 Patent Claim 12 | Kempf |
|---|---|---|
| 12 | The method according to claim 9, wherein the analyzing comprises applying security or data analytic application. | Kempf discloses the method according to claim 9, wherein the analyzing comprises applying security or data analytic application.<br><br>For example, Kempf discloses packet processing for security purposes.<br><br>*See supra* at Claim 9.<br><br>Kempf at [0037] ("The EPC architecture also contains little flexibility for specialized treatment of user flows. Though the architec-ture does provide support for establishing quality of service (QoS), other sorts of data management are not available. For example services involving middle boxes, such as specialized deep packet inspection or interaction with local data caching and processing resources that might be utilized for transcod-ing or augmented reality applications, is difficult to support with the current EPC architecture. Almost all such applica-tions require the packet flows to exit through the PDN Gate-way, thereby being de-tunnelled from GTP, and to be pro-cessed within the wired network.")<br><br>Kempf at [0065] ("A cloud computing system can be composed of any number of computing devices having any range of capabili-ties (e.g., processing power or storage capacity). The cloud computing system can be a private or public system. The computing devices can be in communication with one another across any communication system or network. A cloud com-puting system can support a single cloud or service or any number of discrete clouds or services. Services, applications and similar programs can be virtualized or executed as stan-dard code. In one embodiment, cloud computing systems can support web services applications. Web services applications consist of a load balancing front end that dispatches requests to a pool of Web servers. The requests originate from appli-cations on remote machines on the Internet and therefore the security and privacy requirements are much looser than for applications in a private corporate network.")<br><br>Kempf at [0066] ("Cloud computer systems can also support secure multi-tenancy, in which the cloud computer system provider offers virtual private network (VPN)-like connections between the client's distributed office networks outside the cloud and a VPN within the cloud computing system. This allows the client's applications within the cloud computing system to operate in a network environment that resembles a corporate WAN. For private data centers, in which services are only offered to customers within the corporation owning |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 74 of 1100

| No. | '111 Patent Claim 12 | Kempf |
|---|---|---|
| | | the data center, the security and privacy requirements for multi-tenancy are relaxed. But for public data centers, the cloud operator must ensure that the traffic from multiple tenants is isolated and there is no possibility for traffic from one client to reach another client network.")<br><br>Kempf at [0070] ("The cloud manager 1303 monitors the central pro-cessor unit (CPU) utilization of the EPC control plane entities 1307 and the control plane traffic between the EPC control plane entities 1307 within the cloud. It also monitors the control plane traffic between the end user devices (UEs) and E-NodeBs, which do not have control plane entities in the cloud computing system 1301, and the EPC control plane entities 1307. If the EPC control plane entities 1307 begin to exhibit signs of overloading, such as the utilization of too much CPU time, or the queueing up of too much traffic to be processed, the overloaded control plane entity 1307 requests that the cloud manager 1303 start up a new VM to handle the load. Additionally, the EPC control plane entities 1307 them-selves can issue event notifications to the cloud manager 1303 if they detect internally that they are beginning to experience overloading.")<br><br>Kempf at [0074] ("The operation of the EPC cloud computer system as follows. The UE 1317, E-NodeB 1317, S-GW-C 1307, and P-GW-C signal 1307 to the MME, PCRF, and HSS 1307 using the standard EPC protocols, to establish, modify, and delete bearers and GTP tunnels. This signaling triggers pro-cedure calls with the OpenFlow controller to modify the routing in the EPC as requested. The OpenFlow controller configures the standard OpenFlow switches, the Openflow S-GW-D 1315, and P-GW-D 1311 with flow rules and actions to enable the routing requested by the control plane entities. Details of this configuration are described in further detail herein below.")<br><br>Kempf at [0083] ("If a packet either needs encapsulation or arrives encapsulated with nonzero header flags, header extensions, and/or the GTP-U packet is not a G-PDU packet (i.e. it is a GTP-U control packet), the processing must proceed via the gateway's slow path (software) control plane. GTP-C and GTP' packets directed to the gateway's IP address are a result of mis-configuration and are in error. They must be sent to the OpenFlow controller, since these packets are handled by the S-GW-C and P-GW-C control plane entities in the cloud computing system or to the billing entity handling GTP' and not the S-GW-D and P-GW-D data plane switches.") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 75 of 1100

| No. | '111 Patent Claim 12 | Kempf |
|---|---|---|
| | | Kempf at [0084] ("GTP virtual ports are configured from the Open-Flow controller using a configuration protocol. The details of the configuration protocol are switch-dependent. The con-figuration protocol must support messages that perform following functions: allow the controller to query for and return an indication whether the switch supports GTP fast path virtual ports and what virtual port numbers are used for fast path and slow path GTP-U processing; and allow the controller to instantiate a GTP-U fast path virtual port within a switch datapath for use in the OpenFlow table set-output -port action. The configuration command must be run in a transaction so that, when the results of the action are reported back to the controller, either a GTP-U fast path virtual port for the requested datapath has been instantiated or an error has returned indicating why the request could not be honored. The command also allows the OpenFlow controller to bind a GTP-U virtual port to a physical port. For decapsulation virtual ports, the physical port is an input port. For encapsu-lation virtual ports, the physical port is an output port.")<br><br>Kempf at [0091] ("When a packet header matches a rule associated with the virtual port, the GTP TEID is written into the lower 32 bits of the metadata and the packet is directed to the virtual port. The virtual port calculates the hash of the TEID and looks up the tunnel header information in the tunnel header table. If no such tunnel information is present, the packet is forwarded to the controller with an error indication. Other-wise, the virtual port constructs a GTP tunnel header and encapsulates the packet. Any DSCP bits or VLAN priority bits are additionally set in the IP or MAC tunnel headers, and any VLAN tags or MPLS labels are pushed onto the packet. The encapsulated packet is forwarded out the physical port to which the virtual port is bound.")<br><br>Kempf at [0104] ("In one embodiment, the system implements han-dling of GTP-C and GTP' control packets. Any GTP-C and GTP' control packets that are directed to IP addresses on a gateway switch are in error. These packets need to be handled by the S-GW-C, P-GW-C, and GTP' protocol entities in the cloud computing system, not the S-GW-D and P-GW-D enti-ties in the switches. To catch such packets, the OpenFlow controller must program the switch with the following two rules: the IP destination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); for one rule, the UDP destination port is the GTP-U destination port (2152), for the other, the UDP destination |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 76 of 1100

| No. | '111 Patent Claim 12 | Kempf |
|---|---|---|
| | | port is the GTP-C destination port (2123); the GTP header flags and message type fields are wildcarded.")<br><br>Kempf at [0106] ("This encapsulates the packet and sends it to the OpenFlow controller.") |

| No. | '111 Patent Claim 13 | Kempf |
|---|---|---|
| 13 | The method according to claim 9, wherein the analyzing comprises applying security application that comprises firewall or intrusion detection functionality. | Kempf discloses the method according to claim 9, wherein the analyzing comprises applying security application that comprises firewall or intrusion detection functionality.<br><br>For example, Kempf discloses packet processing for security purposes, i.e., wherein the analyzing comprises applying security application that comprises firewall or intrusion detection functionality. A person of ordinary skill in the art would understand security monitoring of packets can comprise use of a firewall or intrusion detection functionality. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Kempf is found to not meet this limitation, wherein the analyzing comprises applying security application that comprises firewall or intrusion detection functionality would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>*See supra* at Claim 9.<br><br>Kempf at [0037] ("The EPC architecture also contains little flexibility for specialized treatment of user flows. Though the architec-ture does provide support for establishing quality of service (QoS), other sorts of data management are not available. For example services involving middle boxes, such as specialized deep packet inspection or interaction with local data caching and processing resources that might be utilized for transcod-ing or augmented reality applications, is difficult to support with the current EPC architecture. Almost all such applica-tions require the packet flows to exit through the PDN Gate-way, thereby being de-tunnelled from GTP, and to be pro-cessed within the wired network.") |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 77 of 1100

| No. | '111 Patent Claim 13 | Kempf |
|---|---|---|
| | | Kempf at [0065] ("A cloud computing system can be composed of any number of computing devices having any range of capabili-ties (e.g., processing power or storage capacity). The cloud computing system can be a private or public system. The computing devices can be in communication with one another across any communication system or network. A cloud com-puting system can support a single cloud or service or any number of discrete clouds or services. Services, applications and similar programs can be virtualized or executed as stan-dard code. In one embodiment, cloud computing systems can support web services applications. Web services applications consist of a load balancing front end that dispatches requests to a pool of Web servers. The requests originate from appli-cations on remote machines on the Internet and therefore the security and privacy requirements are much looser than for applications in a private corporate network.")<br><br>Kempf at [0066] ("Cloud computer systems can also support secure multi-tenancy, in which the cloud computer system provider offers virtual private network (VPN)-like connections between the client's distributed office networks outside the cloud and a VPN within the cloud computing system. This allows the client's applications within the cloud computing system to operate in a network environment that resembles a corporate WAN. For private data centers, in which services are only offered to customers within the corporation owning the data center, the security and privacy requirements for multi-tenancy are relaxed. But for public data centers, the cloud operator must ensure that the traffic from multiple tenants is isolated and there is no possibility for traffic from one client to reach another client network.")<br><br>Kempf at [0070] ("The cloud manager 1303 monitors the central pro-cessor unit (CPU) utilization of the EPC control plane entities 1307 and the control plane traffic between the EPC control plane entities 1307 within the cloud. It also monitors the control plane traffic between the end user devices (UEs) and E-NodeBs, which do not have control plane entities in the cloud computing system 1301, and the EPC control plane entities 1307. If the EPC control plane entities 1307 begin to exhibit signs of overloading, such as the utilization of too much CPU time, or the queueing up of too much traffic to be processed, the overloaded control plane entity 1307 requests that the cloud manager 1303 start up a new VM to handle the load. Additionally, the EPC control plane entities 1307 them-selves can issue event notifications to the cloud manager 1303 if they detect internally that they are beginning to experience overloading.") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 78 of 1100

| No. | '111 Patent Claim 13 | Kempf |
|---|---|---|
| | | Kempf at [0074] ("The operation of the EPC cloud computer system as follows. The UE 1317, E-NodeB 1317, S-GW-C 1307, and P-GW-C signal 1307 to the MME, PCRF, and HSS 1307 using the standard EPC protocols, to establish, modify, and delete bearers and GTP tunnels. This signaling triggers pro-cedure calls with the OpenFlow controller to modify the routing in the EPC as requested. The OpenFlow controller configures the standard OpenFlow switches, the Openflow S-GW-D 1315, and P-GW-D 1311 with flow rules and actions to enable the routing requested by the control plane entities. Details of this configuration are described in further detail herein below.")<br><br>Kempf at [0083] ("If a packet either needs encapsulation or arrives encapsulated with nonzero header flags, header extensions, and/or the GTP-U packet is not a G-PDU packet (i.e. it is a GTP-U control packet), the processing must proceed via the gateway's slow path (software) control plane. GTP-C and GTP' packets directed to the gateway's IP address are a result of mis-configuration and are in error. They must be sent to the OpenFlow controller, since these packets are handled by the S-GW-C and P-GW-C control plane entities in the cloud computing system or to the billing entity handling GTP' and not the S-GW-D and P-GW-D data plane switches.")<br><br>Kempf at [0084] ("GTP virtual ports are configured from the Open-Flow controller using a configuration protocol. The details of the configuration protocol are switch-dependent. The con-figuration protocol must support messages that perform following functions: allow the controller to query for and return an indication whether the switch supports GTP fast path virtual ports and what virtual port numbers are used for fast path and slow path GTP-U processing; and allow the controller to instantiate a GTP-U fast path virtual port within a switch datapath for use in the OpenFlow table set-output-port action. The configuration command must be run in a transaction so that, when the results of the action are reported back to the controller, either a GTP-U fast path virtual port for the requested datapath has been instantiated or an error has returned indicating why the request could not be honored. The command also allows the OpenFlow controller to bind a GTP-U virtual port to a physical port. For decapsulation virtual ports, the physical port is an input port. For encapsu-lation virtual ports, the physical port is an output port.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 79 of 1100

| No. | '111 Patent Claim 13 | Kempf |
|---|---|---|
| | | Kempf at [0091] ("When a packet header matches a rule associated with the virtual port, the GTP TEID is written into the lower 32 bits of the metadata and the packet is directed to the virtual port. The virtual port calculates the hash of the TEID and looks up the tunnel header information in the tunnel header table. If no such tunnel information is present, the packet is forwarded to the controller with an error indication. Other-wise, the virtual port constructs a GTP tunnel header and encapsulates the packet. Any DSCP bits or VLAN priority bits are additionally set in the IP or MAC tunnel headers, and any VLAN tags or MPLS labels are pushed onto the packet. The encapsulated packet is forwarded out the physical port to which the virtual port is bound.")<br><br>Kempf at [0104] ("In one embodiment, the system implements han-dling of GTP-C and GTP' control packets. Any GTP-C and GTP' control packets that are directed to IP addresses on a gateway switch are in error. These packets need to be handled by the S-GW-C, P-GW-C, and GTP' protocol entities in the cloud computing system, not the S-GW-D and P-GW-D enti-ties in the switches. To catch such packets, the OpenFlow controller must program the switch with the following two rules: the IP destination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); for one rule, the UDP destination port is the GTP-U destination port (2152), for the other, the UDP destination port is the GTP-C destination port (2123); the GTP header flags and message type fields are wildcarded.")<br><br>Kempf at [0106] ("This encapsulates the packet and sends it to the OpenFlow controller.")<br><br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Kempf in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 13 of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references.<br><br>For example, Copeland discloses analysis by the intrusion detection engine on the monitoring appliance to detect communication intruders and suspicious activity. The monitoring appliance may also work in coordination with a firewall. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 80 of 1100

| No. | '111 Patent Claim 13 | Kempf |
|---|---|---|
| | | Copeland at [0065] ("The intrusion detection engine 155 analyzes the flow data 160 to determine if the flow appears to be legitimate traffic or possible suspicious activity. Flows with suspicious activity are assigned a predetermined concern index (CI) value based upon a heuristically predetermined assessment of the significance of the threat of the particular traffic or flow or suspicious activity. The flow concern index values have been derived heuristically from extensive net-work traffic analysis. Concern index values are associated with particular hosts and stored in the host data structure 166 (FIG. 1). Exemplary concern index values for various exemplary flow-based events and other types of events are illustrated in connection with FIGS. 6 and 7.")<br><br>Copeland at [0067] ("The host servers 130 are directly or indirectly coupled to one or more network devices 135 such as routers or switches that support providing a sampled data stream such as that provided by sFlow. In a typical preferred configuration for the present invention, a monitoring appli-ance 150 operating a flow-based intrusion detection engine 155 is receiving sampled packet headers from one or more network devices 135. The monitoring appliance 150 moni-tors the communications between the host server 130 and other hosts 120, 110 in the attempt to detect intrusion activity.")<br><br>Copeland at [0068] ("Those skilled in the art understand that many networks utilize firewalls to limit unwanted network traffic. A monitoring appliance 150 can be connected before a firewall to detect intrusions directed at the network. Con-versely, the monitoring appliance 150 may be installed behind a firewall to detect intrusions that bypass the firewall. Some systems install two firewalls with web and e-mails servers in the so-called "demilitarized zone" or "DMZ" between firewalls. One common placement of the monitor-ing appliance 150 is in this demilitarized zone. Of course, those skilled in the art will appreciate that the flow-based intrusion detection system 155 or appliance 150 can operate without the existence of any firewalls.")<br><br>Copeland at [0069] ("It will now be appreciated that the disclosed meth-odology of intrusion detection is accomplished at least in part by analyzing communication flows to determine if such communications have the flow characteristics of probes or attacks. By analyzing communications for abnormal flow characteristics, attacks can be determined without the need for resource-intensive packet data analysis. A flow can be determined from the packets 101 that are transmitted between two hosts utilizing a single service. The |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 81 of 1100

| No. | '111 Patent Claim 13 | Kempf |
|---|---|---|
| | | addresses and port numbers of communications are easily discerned by analysis of the header information in a datagram.")<br><br>Copeland at [0112] ("FIG. 5 illustrates a logical software architecture of a flow-based intrusion detection engine 155 constructed in accordance with an embodiment of the present invention. As will be understood by those skilled in the art, the system is constructed utilizing Internet-enabled computer systems with computer programs designed to carry out the functions described herein. Preferably, the various computing func-tions are implemented as different but related processes known as "threads" which executed concurrently on modern day multi-threaded, multitasking computer systems.")<br><br>Copeland at Figure 5 |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 82 of 1100

| No. | '111 Patent Claim 13 | Kempf |
|---|---|---|

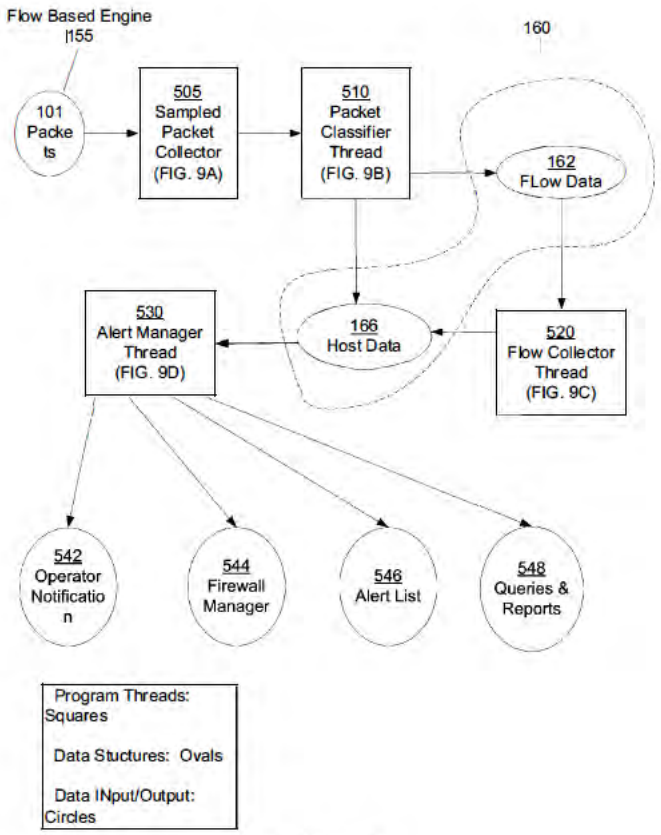



FIG. 5

Copeland at [0149] ("The alert manager 530 also looks for hosts whose CI or traffic (byte rate) exceeds preset alarm thresholds and which have not been handled on previous runs. The new alarm conditions can cause immediate operator notification by an operator notification process 542. These conditions can be highlighted on the user interface, and cause SNMP trap messages to be sent to a network monitor such as HP Openview, and/or email messages to the network adminis-trator that in turn may cause messages to be sent to beepers or cell phones. Messages can also be sent to cause automated devices such as a firewall manager 544 to drop packets going to or from an offending host. It will thus be



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 83 of 1100

| No. | '111 Patent Claim 13 | Kempf |
|---|---|---|
| | | appreciated that the present invention advantageously operates in conjunc-tion with firewalls and other network security devices and processes to provide additional protection for an entity's computer network and computer resources.")<br><br>Copeland at [0177] ("If an alarm threshold has been exceeded, the "Yes" branch of step 975 is followed to step 976. In step 976, the alert manager thread generates certain predetermined signals designed to drawn the attention of a system administrator or other interested person. The alert manager 530 looks for hosts whose CI or traffic (byte rate) exceeds preset alarm thresholds and have not been handled on previous runs. The new alarm conditions can cause immediate operator notifi-cation. These conditions can be highlighted on the user interface, and cause SNMP trap messages to be sent to a network monitor such as HP Openview, and/or email mes-sages to the network administrator that in turn may cause messages to be sent to beepers or cell phones. Messages can also be sent to cause automated devices such as a firewall manager to drop packets going to or from an offending host. Step 976 is followed by step 972, in which the thread 530 awaits the requisite amount of time.")<br><br>For example, Chua '877 discloses security determinations and analysis that involve the use of firewalls or intrusion detection services.<br><br>Chua '877 at 31:48-59 ("In some examples, SDN controller 112 further performs deep packet inspection (DPI) on packets from client device 102 ( 402). For example, SDN controller 112 may inspect one or more preliminary packets of packet flows originating from or directed to client device 102, and after determining that the packet flows are not malicious ( after a predetermined number of packets), stop inspecting the packet flows. Alternatively, SDN controller 112 may program network devices of SDN 106 to forward a predetermined number of packets of the packet flows originating from or destined for client device 102 through a deep packet inspection service device, which may correspond to one of service devices 116.")<br><br>Chua '877 at 25:32-52 ("In the example of FIG. 5, SDN controller 112 determines zones for packet flows through the network devices forming the SDN (304). The zones generally correspond to packet flows, that is, paths through the SDN followed by particular packets |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 84 of 1100

| No. | '111 Patent Claim 13 | Kempf |
|---|---|---|
| | | SDN controller 112 may store data defining the zones in the data model discussed above. The data defining the zones may specify entities (e.g., users, devices, or the like) that have access to each zone. Thus, SDN controller 112 may program network devices of the SDN such that entities that are not authorized to access a particular zone are prevented from accessing the zone. SDN controller 112 may specify a zone using packet header field values, such as a source port, a destination port, a source IP address, a destination IP address, a virtual local area network (VLAN) tag, multiprotocol label switching (MPLS) labels, a packet protocol, and/or an IP subnet. In some cases, SDN controller 112 may specify whether a corresponding packet flow for a zone is suspect or malicious and construct the zone such that packets of the packet flow are prevented from reaching an intended destination. As noted above, zones may be ordered based on priority values when overlap occurs.")<br><br>Chua '877 at 25:53-65 ("Furthermore, SDN controller 112 determines trusted packet flows (306). For example, SDN controller 112 may determine that certain packet flows can be trusted based on security controls, and that other packet flows cannot be trusted based on the security controls. That is, SDN controller 112 may determine whether a packet flow can be trusted based on values of packet headers for the packet flows, e.g., values of headers at various layers of the OSI model ( e.g., any or all of layers 2-7 of the OSI model). In some examples, SDN controller 112 may omit any or all of steps 302, 304, and 306, e.g., omitting any or all of determination of service devices, determination of zones, and/or determination of trusted packet flows.")<br><br>Chua '877 at 5:50-6:5 ("SDN 106 generally serves to interconnect various endpoint devices, such as client device 102 and server device 104. In addition, SDN 106 may provide services to network traffic flowing between client device 102 and server device 104. Alternatively, SDN 106 may provide services to client device 102, without further directing traffic to server device 106. For example, administrator 114 may use SDN controller 112 to program network devices of SDN 106 to direct network traffic for client device 102 to one or more of service devices 116. Service devices 116 may include, for example, intrusion detection service (IDS) devices, intrusion prevention system (IPS) devices, web proxies, web servers, web-application firewalls and the like. In other examples, service devices 116 may, additionally or alternatively, include devices for provid-ing services such as, for example, denial of service (DoS) protection, distributed denial of service (DDoS) protection, traffic |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 85 of 1100

| No. | '111 Patent Claim 13 | Kempf |
|---|---|---|
| | | filtering, wide area network (WAN) acceleration, or other such services. Service devices 116 may also addition-ally or alternatively include malware detection devices, net-work anti-virus devices, network packet capture and analysis devices, honeypot devices, reflector net devices, tar pit devices, domain name service (DNS) and global DNS server devices, mail proxies, and anti-spam devices.")<br><br>Chua '877 at 6:6-24 ("Service devices 116 may, additionally or alternatively, include devices in various device categories such as, for example, network and application security devices, application optimization devices, scaling devices, traffic shaping devices, and/or monitoring and analytics devices. Moreover, although shown as individual devices, it should be understood that service devices may be realized by physical devices, multi-tenant devices, or using virtual services (e.g., cloud-based services). Moreover, service devices 116 may represent multi-function devices. For purposes of example and ease of explanation, this disclosure primarily describes individual service devices. However, it should be understood that the techniques of this disclosure may be readily applied to virtual devices and cloud-based applications, in addition or in the alternative to physical devices. Likewise, where this disclosure refers to a switch or other network device, it should be understood that these techniques may apply to virtual switches or other virtual network devices.")<br><br>Chua '877 at 7:3-13 ("Devices that may be plugged into (that is, communicatively coupled to) SDN controller 112 ( also sometimes referred to as a "FlowDirector") generally include classes of devices found in most network-based DMZs, including firewalls, web proxies, mail proxies, AV (anti-virus) proxies, mail systems, IDS (intrusion detection systems), IPS (intrusion prevention systems), VPN (virtual private network) servers, web application firewalls, vulnerability scanners, network recording and analysis systems, and packet shapers. Most of these devices are either security devices, or traffic engineering or visibility devices, in some examples.")<br><br>Chua '877 at 14:32-51 ("One example use case for SDN controller 112 includes performing internal security zone partitioning. In today's enterprise environment, certain flows can be trusted, based on security controls placed on the end points, while others must be assumed to have some potential for risk. SDN controller 112 may create security zones based both on physical topol-ogy as well as threat assessments based on L2-L4 header information. Business-level security rules can be implemented directly on SDN controller 112 to direct |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 86 of 1100

| No. | '111 Patent Claim 13 | Kempf |
|---|---|---|
| | | only higher risk flows through specific L4-L 7 devices ( e.g., service devices 116) to monitor for or block malicious traffic. That is, when an SDN interconnects a set of enterprise network devices of a common enterprise network and also provides connections for the enterprise network devices outside of the enterprise network, SDN controller 112 may determine that connections between the enterprise network devices within the enterprise network can be trusted, whereas connections to network devices outside the enterprise network cannot be trusted and, therefore, should be monitored by a security device.")<br><br>Chua '877 at 14:52-63 ("Thus, SDN controller 112 may determine separate sets of packet flows based on security controls, e.g., a first set of packet flows that can be trusted and a second set of packet flows that are not trusted. Then, SDN controller 112 may determine a first set of one or more paths for the first set of packet flows that omit a security device for the first set of packet flows (that is, based on the determination that the first set of packet flows can be trusted), and a second set of one or more paths for the second set of packet flows that direct the second set of packet flows through the security device (based on the determination that the second set of packet flows are not trusted).")<br><br>Chua '877 at 14:64-15:3 ("The security controls may include various types of information. For example, the security controls may specify values for one or more packet headers at various layers of the Open Systems Interconnection (OSI) network model. The security controls may specify information for any or all of network layers two, three, four, five, six, and/or seven of the OSI model.")<br><br>Chua '877 at 16:23-44 ("More particularly, control unit 130 may configure any of service devices 116 to send data representative of a particular event to SDN controller 112, and control unit 130 may auto-matically reprogram one or more network devices of SDN 106 in response to such data. For example, security monitor-ing applications of service devices 116 may determine that a specific source port, destination port, source IP address, des-tination IP address, or the like should be acted upon. Alter-natively, security monitoring applications may determine that, due to content or deep packet inspection, a specific type of traffic is malicious and should be blocked. In either case, the corresponding one of service devices 116 may send a message to SDN controller 112 representative of these deter-minations. As yet another example, a network performance device may monitor various performance |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 87 of 1100

| No. | '111 Patent Claim 13 | Kempf |
|---|---|---|
| | | metrics, such as latency, jitter, packet loss, or the like, and provide feedback data to SDN controller 112 based on these metrics. SDN controller 112 may respond by programming network devices of SDN 106 to perform a programmed action, such as allowing corresponding traffic, blocking corresponding traf-fic, mirroring corresponding traffic, redirecting correspond-ing traffic.")<br><br>Chua '877 at 19:60-20:4 ("FIG. 3 is a conceptual diagram illustrating an example 60 system 200 including various devices that may be used in accordance with the techniques of this disclosure. In this example, system 200 includes various network devices, including firewall 206, router 208, switch 210, web proxy 212, intrusion detection system (IDS) 214, web server 216, 65 administrator ("admin") workstation 220, and software defined network (SDN) controller 218. Web clients 202 can access system 200 via a network, such as the Internet, e.g., Internet 204. Internet 204 may include additional network devices not explicitly shown in FIG. 3, such as routers, switches, hubs, gateways, security devices, or the like.") |

| No. | '111 Patent Claim 14 | Kempf |
|---|---|---|
| 14 | The method according to claim 9, wherein the analyzing comprises performing Deep Packet Inspection (DPI) or using a DPI engine on the packet. | Kempf discloses the method according to claim 9, wherein the analyzing comprises performing Deep Packet Inspection (DPI) or using a DPI engine on the packet.<br><br>For example, Kempf discloses packet processing for security purposes. A person of ordinary of skill in the art would understand the current architectures could be modified to provide deep packet inspection functionality. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Kempf is found to not meet this limitation, wherein the analyzing comprises performing Deep Packet Inspection (DPI) or using a DPI engine on the packet would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>*See supra* at Claim 9.<br><br>Kempf at [0037] ("The EPC architecture also contains little flexibility for specialized treatment of user flows. Though the architec-ture does provide support for establishing |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 88 of 1100

| No. | '111 Patent Claim 14 | Kempf |
|---|---|---|
| | | quality of service (QoS), other sorts of data management are not available. For example services involving middle boxes, such as specialized deep packet inspection or interaction with local data caching and processing resources that might be utilized for transcod-ing or augmented reality applications, is difficult to support with the current EPC architecture. Almost all such applica-tions require the packet flows to exit through the PDN Gate-way, thereby being de-tunnelled from GTP, and to be pro-cessed within the wired network.")<br><br>Kempf at [0065] ("A cloud computing system can be composed of any number of computing devices having any range of capabili-ties (e.g., processing power or storage capacity). The cloud computing system can be a private or public system. The computing devices can be in communication with one another across any communication system or network. A cloud com-puting system can support a single cloud or service or any number of discrete clouds or services. Services, applications and similar programs can be virtualized or executed as stan-dard code. In one embodiment, cloud computing systems can support web services applications. Web services applications consist of a load balancing front end that dispatches requests to a pool of Web servers. The requests originate from appli-cations on remote machines on the Internet and therefore the security and privacy requirements are much looser than for applications in a private corporate network.")<br><br>Kempf at [0066] ("Cloud computer systems can also support secure multi-tenancy, in which the cloud computer system provider offers virtual private network (VPN)-like connections between the client's distributed office networks outside the cloud and a VPN within the cloud computing system. This allows the client's applications within the cloud computing system to operate in a network environment that resembles a corporate WAN. For private data centers, in which services are only offered to customers within the corporation owning the data center, the security and privacy requirements for multi-tenancy are relaxed. But for public data centers, the cloud operator must ensure that the traffic from multiple tenants is isolated and there is no possibility for traffic from one client to reach another client network.")<br><br>Kempf at [0070] ("The cloud manager 1303 monitors the central pro-cessor unit (CPU) utilization of the EPC control plane entities 1307 and the control plane traffic between the EPC control plane entities 1307 within the cloud. It also monitors the control plane traffic between the end user devices (UEs) and E-NodeBs, which do not have control plane entities |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 89 of 1100

| No. | '111 Patent Claim 14 | Kempf |
|---|---|---|
| | | in the cloud computing system 1301, and the EPC control plane entities 1307. If the EPC control plane entities 1307 begin to exhibit signs of overloading, such as the utilization of too much CPU time, or the queueing up of too much traffic to be processed, the overloaded control plane entity 1307 requests that the cloud manager 1303 start up a new VM to handle the load. Additionally, the EPC control plane entities 1307 them-selves can issue event notifications to the cloud manager 1303 if they detect internally that they are beginning to experience overloading.")<br><br>Kempf at [0074] ("The operation of the EPC cloud computer system as follows. The UE 1317, E-NodeB 1317, S-GW-C 1307, and P-GW-C signal 1307 to the MME, PCRF, and HSS 1307 using the standard EPC protocols, to establish, modify, and delete bearers and GTP tunnels. This signaling triggers pro-cedure calls with the OpenFlow controller to modify the routing in the EPC as requested. The OpenFlow controller configures the standard OpenFlow switches, the Openflow S-GW-D 1315, and P-GW-D 1311 with flow rules and actions to enable the routing requested by the control plane entities. Details of this configuration are described in further detail herein below.")<br><br>Kempf at [0083] ("If a packet either needs encapsulation or arrives encapsulated with nonzero header flags, header extensions, and/or the GTP-U packet is not a G-PDU packet (i.e. it is a GTP-U control packet), the processing must proceed via the gateway's slow path (software) control plane. GTP-C and GTP' packets directed to the gateway's IP address are a result of mis-configuration and are in error. They must be sent to the OpenFlow controller, since these packets are handled by the S-GW-C and P-GW-C control plane entities in the cloud computing system or to the billing entity handling GTP' and not the S-GW-D and P-GW-D data plane switches.")<br><br>Kempf at [0084] ("GTP virtual ports are configured from the Open-Flow controller using a configuration protocol. The details of the configuration protocol are switch-dependent. The con-figuration protocol must support messages that perform following functions: allow the controller to query for and return an indication whether the switch supports GTP fast path virtual ports and what virtual port numbers are used for fast path and slow path GTP-U processing; and allow the controller to instantiate a GTP-U fast path virtual port within a switch datapath for use in the OpenFlow table set-output-port action. The configuration command must be run in a transaction so that, when the results of the action are reported |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 90 of 1100

| No. | '111 Patent Claim 14 | Kempf |
|---|---|---|
| | | back to the controller, either a GTP-U fast path virtual port for the requested datapath has been instantiated or an error has returned indicating why the request could not be honored. The command also allows the OpenFlow controller to bind a GTP-U virtual port to a physical port. For decapsulation virtual ports, the physical port is an input port. For encapsu-lation virtual ports, the physical port is an output port.")<br><br>Kempf at [0091] ("When a packet header matches a rule associated with the virtual port, the GTP TEID is written into the lower 32 bits of the metadata and the packet is directed to the virtual port. The virtual port calculates the hash of the TEID and looks up the tunnel header information in the tunnel header table. If no such tunnel information is present, the packet is forwarded to the controller with an error indication. Other-wise, the virtual port constructs a GTP tunnel header and encapsulates the packet. Any DSCP bits or VLAN priority bits are additionally set in the IP or MAC tunnel headers, and any VLAN tags or MPLS labels are pushed onto the packet. The encapsulated packet is forwarded out the physical port to which the virtual port is bound.")<br><br>Kempf at [0104] ("In one embodiment, the system implements han-dling of GTP-C and GTP' control packets. Any GTP-C and GTP' control packets that are directed to IP addresses on a gateway switch are in error. These packets need to be handled by the S-GW-C, P-GW-C, and GTP' protocol entities in the cloud computing system, not the S-GW-D and P-GW-D enti-ties in the switches. To catch such packets, the OpenFlow controller must program the switch with the following two rules: the IP destination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); for one rule, the UDP destination port is the GTP-U destination port (2152), for the other, the UDP destination port is the GTP-C destination port (2123); the GTP header flags and message type fields are wildcarded.")<br><br>Kempf at [0106] ("This encapsulates the packet and sends it to the OpenFlow controller.")<br><br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Kempf in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 14 of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references. |


~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 91 of 1100

| No. | '111 Patent Claim 14 | Kempf |
|---|---|---|
| | | For example, Chua '877 discloses analyzing, including deep packet inspection, performed by the controller on packets received.<br><br>Chua '877 at 31:48-59 ("In some examples, SDN controller 112 further performs deep packet inspection (DPI) on packets from client device 102 ( 402). For example, SDN controller 112 may inspect one or more preliminary packets of packet flows originating from or directed to client device 102, and after determining that the packet flows are not malicious ( after a predetermined number of packets), stop inspecting the packet flows. Alternatively, SDN controller 112 may program network devices of SDN 106 to forward a predetermined number of packets of the packet flows originating from or destined for client device 102 through a deep packet inspection service device, which may correspond to one of service devices 116.")<br><br>Chua '877 at 10:48-52 ("As another possible extension, the central control platform can also capture and inspect the first or a fixed number of packets to perform deep packet inspection for application classification to extend the policy enforcement to specific application types.")<br><br>Chua '877 at 16:23-44 ("More particularly, control unit 130 may configure any of service devices 116 to send data representative of a particular event to SDN controller 112, and control unit 130 may auto-matically reprogram one or more network devices of SDN 106 in response to such data. For example, security monitor-ing applications of service devices 116 may determine that a specific source port, destination port, source IP address, des-tination IP address, or the like should be acted upon. Alter-natively, security monitoring applications may determine that, due to content or deep packet inspection, a specific type of traffic is malicious and should be blocked. In either case, the corresponding one of service devices 116 may send a message to SDN controller 112 representative of these deter-minations. As yet another example, a network performance device may monitor various performance metrics, such as latency, jitter, packet loss, or the like, and provide feedback data to SDN controller 112 based on these metrics. SDN controller 112 may respond by programming network devices of SDN 106 to perform a programmed action, such as allowing corresponding traffic, blocking corresponding traf-fic, mirroring corresponding traffic, redirecting correspond-ing traffic.") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 92 of 1100

| No. | '111 Patent Claim 14 | Kempf |
|---|---|---|
| | | Chua '877 at 31:48-59 ("In some examples, SDN controller 112 further performs deep packet inspection (DPI) on packets from client device 102 ( 402). For example, SDN controller 112 may inspect one or more preliminary packets of packet flows originating from or directed to client device 102, and after determining that the packet flows are not malicious ( after a predetermined number of packets), stop inspecting the packet flows. Alternatively, SDN controller 112 may program network devices of SDN 106 to forward a predetermined number of packets of the packet flows originating from or destined for client device 102 through a deep packet inspection service device, which may correspond to one of service devices 116.")<br><br>For example, Chandrasekaran discloses the controller performing Deep Packet Inspection, which provides the ability to look into the packet past basic header information so that the contents of a particular packet can be determined, i.e., analyzing comprises performing Deep Packet Inspection (DPI) or using a DPI engine on the packet.<br><br>Chandrasekaran at [0014] ("The term 'wireless controller' or 'controller' as used herein may refer to a wireless LAN (local area network) controller, mobility controller, wireless control device, wire-less control system, or any other network device operable to perform control functions for a wireless network. The net-work site may also include a wireless control system or other platform for centralized wireless LAN planning, configura-tion, and management. The wireless controller 12 enables system wide functions for wireless applications and may support any number of access points 14. Each access point 14 may serve any number of mobile devices 16 in the wireless network. The wireless controller 12 may be, for example, a standalone device or a rack-mounted appliance. In the example shown in FIG. 1, the wireless controller 12 and access points 14 are separate devices and may be located remote from one another. The wireless controller 12 may also be integrated with the access point 14 ( e.g., autonomous AP) or located at a switch, router, switch/router, or other network device. Thus, the wireless controller 12 may be a physical device located at a standalone device, access point, switch, router, or other network device. The wireless controller 12 may also be a virtual device located in a network or cloud, for example.")<br><br>Chandrasekaran at [0016] ("The wireless controller 12 includes a stateful appli-cation classifier 18 and the AP 14 includes a stateless appli-cation classifier 22. After the stateful |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 93 of 1100

| No. | '111 Patent Claim 14 | Kempf |
|---|---|---|
| | | classifier 18 identifies the application, the controller 12 transmits ( e.g., pushes) clas-sification information 26 to the AP 14 so that the AP can perform stateless classification and apply policies ( e.g., QoS or other policies) to traffic received from the mobile device 16. The controller 12 may also provide the classification information 26 to another AP 14 if the client 16 roams to a new AP, as shown in FIG. 1. Implementation of the stateful classifier 18 at the controller 12 and stateless classifier 22 at the AP 14 allows for policies to be applied for downstream traffic (packet 25) at the wireless controller 12, and for upstream traffic (packet 28) at the access point 14.")<br><br>Chandrasekaran at [0020] ("In one embodiment, the stateful classifier 18 is a classification engine configured for NBAR (Network Based Application Recognition) or other technology used to classify applications. The classifier 18 is operable to recognize a wide variety of applications, including Web-based and client/ server applications. The applications may include, for example, Skype, YouTube, Netflix, WebEx, Google Voice, BitTorrent, Citrix, virtual desktop, PCoIP, or any other appli-cation. The classification engine may be configured, for example, to identify generic protocols and perform heuristic analysis for encrypted protocols. The classifiers 18, 22 are configured to perform deep packet inspection (DPI), which provides the ability to look into the packet past basic header information so that the contents of a particular packet can be determined.")<br><br>Chandrasekaran at [0021] ("Once the application is recognized, QoS or other policies associated with the application can be applied to traffic so that the network can invoke services for that par-ticular application. For example, the application may have certain requirements and expectations from the network infrastructure, which may be specified in terms of bandwidth, delay, jitter, throughput, packet loss, or other performance attributes.")<br><br>Chandrasekaran at [0023] ("In one embodiment, the classification information 26 transmitted from the controller 12 to the AP 14 includes tuple information for a flow ( e.g., source IP address, destina-tion IP address, source port, destination port, and protocol), application identifier (ID), and stateless DPI information. Stateless DPI information includes classification and sub-classification information ( e.g., fixed or variable offset with a pattern or regular expression) and rules for applying policies on the sub-classified packets. The policies may include, for example, drop packet, mark a DSCP (Differentiated Services Code Point) value in the packet, or rate limit the traffic.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 94 of 1100

| No. | '111 Patent Claim 14 | Kempf |
|---|---|---|
| | | Chandrasekaran at [0026] ("Memory 34 may be a volatile memory or non-vola-tile storage, which stores various applications, operating sys-tems, modules, and data for execution and use by the proces-sor 32. Memory 34 may include, for example, classification database 35. The classification database 35 may be any data structure configured for at least temporarily storing classifi-cation information including, for example, flow information, application ID, stateless DPI rules, and policies.")<br><br>Chandrasekaran at [0031] ("FIG. 3 is a flowchart illustrating an example of a process at the controller 12 for classification of traffic for application aware policies in a wireless network, in accor-dance with one embodiment. At step 40, the controller 12 receives packets belonging to a network flow. The controller 12 performs stateful classification to identify an application associated with the flow ( step 42). The controller 12 transmits classification information ( e.g., flow information, stateless DPI rule, and policy) to the AP 14 for use in stateless classi-fication at the AP (step 44). The controller 12 applies policies to downstream traffic (received at the controller and destined for the client 16) (step 46) and receives upstream traffic for which policies have been applied at the AP 14 (step 48). If the controller 12 determines ( e.g., receives an indication) that the client 16 has roamed, it transmits the classification informa-tion to the new AP 14 to which the client has roamed (steps 50 and 52).")<br><br>Chandrasekaran at [0033] ("The following describes an example of the above process for WebEx traffic that has different sub-classifications for voice and video traffic. Stateful classification is first performed by the controller 12 at the beginning of the flow. The controller 12 may need to process, for example, 10, 100, or any other number of packets to classify the flow as Web Ex traffic. Once the classification is performed, the controller 12 sends the stateless DPI rules and flow information to the AP 14 for stateless sub-classification to distinguish voice, video, or data within a WebEx flow. For example, after the controller 12 identifies the WebEx meeting traffic, it pushes the tuple, the stateless DPI rules (as shown below), and policies to the AP 14 for upstream traffic marking, dropping, or rate-limit-ing. If the client 16 roams, the controller 12 transmits the same classification information to the new AP to which the client has roamed.") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 95 of 1100

| No. | '111 Patent Claim 15 | Kempf |
|---|---|---|
| 15[a] | The method according to claim 9, wherein the packet comprises distinct header and payload fields, and | Kempf discloses the method according to claim 9, wherein the packet comprises distinct header and payload fields.<br><br>For example, Kempf discloses packets with packet fields and payload fields.<br><br>*See supra* at Claim 9.<br><br>Kempf at [0045] ("FIG. 2 is a diagram illustrating one embodiment of the contents of a flow table entry. The forwarding table 107 is populated with entries consisting of a rule 201 defining matches for fields in packet headers; an action 203 associated to the flow match; and a collection of statistics 205 on the flow. When an incoming packet is received a lookup for a matching rule is made in the flow table 107. If the incoming packet matches a particular rule, the associated action defined in that flow table entry is performed on the packet.")<br><br>Kempf at [0046] ("A rule 201 contains key fields from several headers in the protocol stack, for example source and destination Ethernet MAC addresses, source and destination IP addresses, IP protocol type number, incoming and outgoing TCP or UDP port numbers. To define a flow, all the available matching fields may be used. But it is also possible to restrict the matching rule to a subset of the available fields by using wildcards for the unwanted fields.")<br><br>Kempf at [0050] ("FIG. 4 illustrates one embodiment of the processing of packets through an OpenFlow 1.1 switched packet pro-cessing pipeline. A received packet is compared against each of the flow tables 401. After each flow table match, the actions are accumulated into an action set. If processing requires matching against another flow table, the actions in the matched rule include an action directing processing to the next table in the pipeline. Absent the inclusion of an action in the set to execute all accumulated actions immediately, the actions are executed at the end 403 of the packet processing pipeline. An action allows the writing of data to a metadata register, which is carried along in the packet processing pipe-line like the packet header.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 96 of 1100

| No. | '111 Patent Claim 15 | Kempf |
|---|---|---|
| | | Kempf at [0051] ("FIG. 5 is a flowchart of one embodiment of the OpenFlow 1.1 rule matching process. OpenFlow 1.1 contains support for packet tagging. OpenFlow 1.1 allows matching based on header fields and multi-protocol label switching (MPLS) labels. One virtual LAN (VLAN) label and one MPLS label can be matched per table. The rule matching process is initiated with the arrival of a packet to be processed (Block 501 ). Starting at the first table 0 a lookup is performed to determine a match with the received packet (Block 503). If there is no match in this table, then one of a set of default actions is taken (i.e., send packet to controller, drop the packet or continue to next table) (Block 509). If there is a match, then an update to the action set is made along with counters, packet or match set fields and meta data (Block 505). A check is made to determine the next table to process, which can be the next table sequentially or one specified by an action of a matching rule (Block 507). Once all of the tables have been processed, then the resulting action set is executed (Block 511). FIG. 6 is a diagram of the fields, which a matching process can utilize for identifying rules to apply to a packet.")<br><br>Kempf at [0052] ("Actions allow manipulating of tag stacks by pushing and popping labels. Combined with multiple tables, VLAN or MPLS label stacks can be processed by matching one label per table. FIG. 7 is a flow chart of one embodiment of a header parsing process. The parsing process matches a packet header by initializing a set of match fields (Block 701) and checking for the presence of a set of different header types. The process checks for a VLAN tag (Block 703). If the VLAN tag is present, then there are a series of processing steps for the VLAN tag (Blocks 705-707). If the switch supports MPLS (Block 709), then there are a series of steps for detecting and processing the MPLS header information (Blocks 711-715). If the switch supports address resolution protocol (ARP), then there are a series of steps for processing the ARP header (Blocks 719 and 721). If the packet has an IP header (Block 723), then there are a series of steps for processing the IP header (Blocks 725-733). This process is performed for each received packet.")<br><br>Kempf at [0081] ("In one embodiment, OpenFlow is modified to pro-vide rules for GTP TEID Routing. FIG. 17 is a diagram of one embodiment of the OpenFlow flow table modification for GTP TEID routing. An OpenFlow switch that supports TEID routing matches on the 2 byte (16 bit) collection of header fields and the 4 byte (32 bit) GTP TEID, in addition to other OpenFlow header fields, in at least one flow table ( e.g., the first flow table). The GTP TEID flag can be wildcarded (i.e. matches are "don't care"). In one |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 97 of 1100

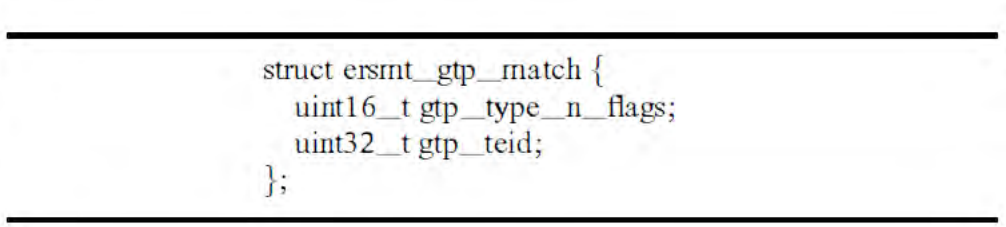
| No. | '111 Patent Claim 15 | Kempf |
|---|---|---|
| | | embodiment, the EPC pro-tocols do not assign any meaning to TEIDs other than as an endpoint identifier for tunnels, like ports in standard UDP/ TCP transport protocols. In other embodiments, the TEIDs can have a correlated meaning or semantics. The GTP header flags field can also be wildcarded, this can be partially matched by combining the following bitmasks: 0xFF00- Match the Message Type field; 0xe0-Match the Version field; 0xl0-Match the PT field; 0x04-Match the E field; 0x02- Match the S field; and 0x0l-Match the PN field.")<br><br>Kempf at [0083] ("If a packet either needs encapsulation or arrives encapsulated with nonzero header flags, header extensions, and/or the GTP-U packet is not a G-PDU packet (i.e. it is a GTP-U control packet), the processing must proceed via the gateway's slow path (software) control plane. GTP-C and GTP' packets directed to the gateway's IP address are a result of mis-configuration and are in error. They must be sent to the OpenFlow controller, since these packets are handled by the S-GW-C and P-GW-C control plane entities in the cloud computing system or to the billing entity handling GTP' and not the S-GW-D and P-GW-D data plane switches.")<br><br>Kempf at [0086] ("In one embodiment, an OpenFlow GTP gateway maintains a hash table mapping GTP TEIDs into the tunnel header fields for their bearers. FIG. 18 is a diagram of the structure of a flow table row. The TEID hash keys are calcu-lated using a suitable hash algorithm with low collision fre-quency, for example SHA-1. The gateway maintains one such flow table row for each GTP TEID/bearer. The TEID field contains the GTP TEID for the tunnel. The VLAN tags and MPLS labels fields contain an ordered list of VLAN tags and/or MPLS labels defining tunnels into which the packet needs to be routed. The VLAN priority bits and MPLS traffic class bits are included in the labels. Such tunnels may or may not be required. If they are not required, then these fields are empty. The tunnel origin source IP address contains the address on the encapsulating gateway to which any control traffic involving the tunnel should be directed (for example, error indications). The tunnel end destination IP address field contains the IP address of the gateway to which the tunneled packet should be routed, at which the packet will be decap-sulated and removed from the GTP tunnel. The QoS DSCP field contains the DiffServe Code Point, if any, for the bearer in the case of a dedicated bearer. This field may be empty if the bearer is a default bearer with best effort QoS, but will contain nonzero values if the bearer QoS is more than best effort.") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 98 of 1100

| No. | '111 Patent Claim 15 | Kempf |
|---|---|---|
| | | Kempf at [0092] ("In one embodiment, the system implements a GTP fast path decapsulation virtual port. When requested by the S-GW and P-GW control plane software running in the cloud computing system, the gateway switch installs rules and actions for routing GTP encapsulated packets out of GTP tunnels. The rules match the GTP header flags and the GTP TEID for the packet, in the modified OpenFlow flow table shown in FIG. 17 as follows: the IP destination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); the UDP destination port is the GTP-U destination port (2152); and the header fields and message type field is wildcarded with the flag 0XFFF0 and the upper two bytes of the field match the G-PDU message type (255) while the lower two bytes match 0x30, i.e. the packet is a GTP packet not a GTP' packet and the version number is 1.")<br><br>Kempf at [0094] ("In one embodiment, the system implements han-dling of GTP-U control packets. The OpenFlow controller programs the gateway switch flow tables with 5 rules for each gateway switch IP address used for GTP traffic. These rules contain specified values for the following fields: the IP des-tination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); the UDP destination port is the GTP-U destination port (2152); the GTP header flags and message type field is wildcarded with 0xFFF0; the value of the header flags field is 0x30, i.e. the version number is 1 and the PT field is 1; and the value of the message type field is one of 1 (Echo Request), 2 (Echo Response), 26 (Error Indication), 31 (Support for Extension Headers Notification), or 254 (End Marker).")<br><br>Kempf at [0115] ("In another embodiment, OpenF!ow 1.2 supports an extensible match structure, OXM, shown in FIG. 22, in which the flow match is encoded as a type-length-value. The oxm_ class field values 0x0000 to 0x7FFF are reserved for Open Network Foundation members, Ox8000 to 0xFFFE are reserved for future standardization, and 0xFFFF is designated for experimentation. The oxm_field identifies a subtype within the class, the HM field specifies whether the value contains a bitmask (yes=l, no=0), and oxm_length contains the length of the value payload.")<br><br>Kempf at [0116] ("For GTP TEID routing, we define a value payload by the ersmt_gtp_match structure: |


~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 99 of 1100

| No. | '111 Patent Claim 15 | Kempf |
|---|---|---|
| | |  |
| 15[b] | wherein the analyzing comprises checking part of, or whole of, the payload field. | Kempf discloses wherein the analyzing comprises checking part of, or whole of, the payload field.<br><br>For example, Kempf discloses processing the message type field of the packet.<br><br>Kempf at [0081] ("In one embodiment, OpenFlow is modified to pro-vide rules for GTP TEID Routing. FIG. 17 is a diagram of one embodiment of the OpenFlow flow table modification for GTP TEID routing. An OpenFlow switch that supports TEID routing matches on the 2 byte (16 bit) collection of header fields and the 4 byte (32 bit) GTP TEID, in addition to other OpenFlow header fields, in at least one flow table ( e.g., the first flow table). The GTP TEID flag can be wildcarded (i.e. matches are "don't care"). In one embodiment, the EPC pro-tocols do not assign any meaning to TEIDs other than as an endpoint identifier for tunnels, like ports in standard UDP/ TCP transport protocols. In other embodiments, the TEIDs can have a correlated meaning or semantics. The GTP header flags field can also be wildcarded, this can be partially matched by combining the following bitmasks: 0xFF00- Match the Message Type field; 0xe0-Match the Version field; 0xl0-Match the PT field; 0x04-Match the E field; 0x02- Match the S field; and 0x0l-Match the PN field.")<br><br>Kempf at [0092] ("In one embodiment, the system implements a GTP fast path decapsulation virtual port. When requested by the S-GW and P-GW control plane software running in the cloud computing system, the gateway switch installs rules and actions for routing GTP encapsulated packets out of GTP tunnels. The rules match the GTP header flags and the GTP TEID for the packet, in the modified OpenFlow flow table shown in FIG. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 100 of 1100

| No. | '111 Patent Claim 15 | Kempf |
|---|---|---|
| | | 17 as follows: the IP destination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); the UDP destination port is the GTP-U destination port (2152); and the header fields and message type field is wildcarded with the flag 0XFFF0 and the upper two bytes of the field match the G-PDU message type (255) while the lower two bytes match 0x30, i.e. the packet is a GTP packet not a GTP' packet and the version number is 1.")<br><br>Kempf at [0094] ("In one embodiment, the system implements han-dling of GTP-U control packets. The OpenFlow controller programs the gateway switch flow tables with 5 rules for each gateway switch IP address used for GTP traffic. These rules contain specified values for the following fields: the IP des-tination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); the UDP destination port is the GTP-U destination port (2152); the GTP header flags and message type field is wildcarded with 0xFFF0; the value of the header flags field is 0x30, i.e. the version number is 1 and the PT field is 1; and the value of the message type field is one of 1 (Echo Request), 2 (Echo Response), 26 (Error Indication), 31 (Support for Extension Headers Notification), or 254 (End Marker).")<br><br>Kempf at [0098] ("The header flags and message type fields for the three rules are wildcarded with the following bitmasks and match as follows: bitmask 0xFFF4 and the upper two bytes match the G-PDU message type (255) while the lower two bytes are Ox34, indicating that the version number is 1, the packet is a GTP packet, and there is an extension header present; bitmask 0xFFFF2 and the upper two bytes match the G-PDU message type (255) while the lower two bytes are 0x32, indicating that the version number is 1, the packet is a GTP packet, and there is a sequence number present; and bitmask 0xFF0l and the upper two bytes match the G-PDU message type (255) while the lower two bytes are 0x31, indicating that the version number is 1, the packet is a GTP packet, and a N-PDU is present.")<br><br>Kempf at [0115] ("In another embodiment, OpenF!ow 1.2 supports an extensible match structure, OXM, shown in FIG. 22, in which the flow match is encoded as a type-length-value. The oxm_ class field values 0x0000 to 0x7FFF are reserved for Open Network Foundation members, Ox8000 to 0xFFFE are reserved for future standardization, and 0xFFFF is designated for experimentation. The oxm_ field identifies a subtype within the |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 101 of 1100

| No. | '111 Patent Claim 15 | Kempf |
|---|---|---|
| | | class, the HM field specifies whether the value contains a bitmask (yes=l, no=0), and oxm_length contains the length of the value payload.")<br><br>Kempf at [0116] ("For GTP TEID routing, we define a value payload by the ersmt_gtp_match structure:<br><br>struct ersmt_gtp_match {<br>uint16_t gtp_type_n_flags;<br>uint32_t gtp_teid;<br>}; |

| No. | '111 Patent Claim 16 | Kempf |
|---|---|---|
| 16[a] | The method according to claim 1, wherein the packet comprises distinct header and payload fields, | Kempf discloses the method according to claim 1, wherein the packet comprises distinct header and payload fields.<br><br>*See supra* at Claim 1, 15[a]. |
| 16[b] | the header comprises one or more flag bits, and | Kempf discloses the header comprises one or more flag bits.<br><br>For example, Kempf discloses packet headers with flag bits.<br><br>Kempf at [0081] ("In one embodiment, OpenFlow is modified to pro-vide rules for GTP TEID Routing. FIG. 17 is a diagram of one embodiment of the OpenFlow flow table modification for GTP TEID routing. An OpenFlow switch that supports TEID routing matches on the 2 byte (16 bit) collection of header fields and the 4 byte (32 bit) GTP TEID, in addition to other OpenFlow header fields, in at least one flow table ( e.g., the first flow table). The GTP TEID flag can be wildcarded (i.e. matches are "don't care"). In one embodiment, the EPC pro-tocols do not assign any meaning to TEIDs other than as an endpoint identifier for tunnels, like ports in standard UDP/ TCP transport protocols. In other |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 102 of 1100

| No. | '111 Patent Claim 16 | Kempf |
|---|---|---|
| | | embodiments, the TEIDs can have a correlated meaning or semantics. The GTP header flags field can also be wildcarded, this can be partially matched by combining the following bitmasks: 0xFF00- Match the Message Type field; 0xe0-Match the Version field; 0xl0-Match the PT field; 0x04-Match the E field; 0x02- Match the S field; and 0x0l-Match the PN field.")<br><br>Kempf at [0082] ("In one embodiment, OpenFlow can be modified to support virtual ports for fast path GTP TEID encapsulation and decapsulation. An OpenFlow mobile gateway can be used to support GTP encapsulation and decapsulation with virtual ports. The GTP encapsulation and decapsulation virtual ports can be used for fast encapsulation and decapsulation of user data packets within GTP-U tunnels, and can be designed simply enough that they can be implemented in hardware or firmware. For this reason, GTP virtual ports may have the following restrictions on traffic they will handle: Protocol Type (PT) field= 1, where GTP encapsulation ports only sup-port GTP, not GTP' (PT field=0); Extension Header flag (E)=0, where no extension headers are supported, Sequence Number flag (S)=0, where no sequence numbers are sup-ported; N-PDU flag (PN)=0; and Message type=255, where Only G-PDU messages, i.e. tunneled user data, is supported in the fast path.")<br><br>Kempf at [0083] ("If a packet either needs encapsulation or arrives encapsulated with nonzero header flags, header extensions, and/or the GTP-U packet is not a G-PDU packet (i.e. it is a GTP-U control packet), the processing must proceed via the gateway's slow path (software) control plane. GTP-C and GTP' packets directed to the gateway's IP address are a result of mis-configuration and are in error. They must be sent to the OpenFlow controller, since these packets are handled by the S-GW-C and P-GW-C control plane entities in the cloud computing system or to the billing entity handling GTP' and not the S-GW-D and P-GW-D data plane switches.")<br><br>Kempf at [0088] ("To support slow path encapsulation, the software control plane on the switch maintains a hash table with keys calculated from the GTP-U TEID. The TEID hash keys are calculated using a suitable hash algorithm with low collision frequency, for example SHA-1. The flow table entries contain a record of how the packet header, including the GTP encap-sulation header, should be configured. This includes: the same header fields as for the hardware or firmware encapsu-lation table in FIG.18; values for the GTP header |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 103 of 1100

| No. | '111 Patent Claim 16 | Kempf |
|---|---|---|
| | | flags (PT, E, S, and PN); the sequence number and/or the N-PDU number if any; if the E flag is 1, then the flow table contains a list of the extension headers, including their types, which the slow path should insert into the GTP header.")<br><br>Kempf at [0092] ("In one embodiment, the system implements a GTP fast path decapsulation virtual port. When requested by the S-GW and P-GW control plane software running in the cloud computing system, the gateway switch installs rules and actions for routing GTP encapsulated packets out of GTP tunnels. The rules match the GTP header flags and the GTP TEID for the packet, in the modified OpenFlow flow table shown in FIG. 17 as follows: the IP destination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); the UDP destination port is the GTP-U destination port (2152); and the header fields and message type field is wildcarded with the flag 0XFFF0 and the upper two bytes of the field match the G-PDU message type (255) while the lower two bytes match 0x30, i.e. the packet is a GTP packet not a GTP' packet and the version number is 1.")<br><br>Kempf at [0094] ("In one embodiment, the system implements han-dling of GTP-U control packets. The OpenFlow controller programs the gateway switch flow tables with 5 rules for each gateway switch IP address used for GTP traffic. These rules contain specified values for the following fields: the IP des-tination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); the UDP destination port is the GTP-U destination port (2152); the GTP header flags and message type field is wildcarded with 0xFFF0; the value of the header flags field is 0x30, i.e. the version number is 1 and the PT field is 1; and the value of the message type field is one of 1 (Echo Request), 2 (Echo Response), 26 (Error Indication), 31 (Support for Extension Headers Notification), or 254 (End Marker).")<br><br>Kempf at [0098] ("The header flags and message type fields for the three rules are wildcarded with the following bitmasks and match as follows: bitmask 0xFFF4 and the upper two bytes match the G-PDU message type (255) while the lower two bytes are Ox34, indicating that the version number is 1, the packet is a GTP packet, and there is an extension header present; bitmask 0xFFFF2 and the upper two bytes match the G-PDU message type (255) while the lower two bytes are 0x32, indicating that the version number is 1, the packet is a GTP packet, and there is a sequence number present; and bitmask 0xFF0l and the upper |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 104 of 1100

| No. | '111 Patent Claim 16 | Kempf |
|---|---|---|
| | | two bytes match the G-PDU message type (255) while the lower two bytes are 0x31, indicating that the version number is 1, the packet is a GTP packet, and a N-PDU is present.")<br><br>Kempf at [0114] ("The gtp_type_n_flags field contains the GTP mes-sage type in the upper 8 bits and the GTP header flags in the lower 8 bits. The gtp_teid field contains the GTP TEID. The gtp_ wildcard field indicates whether the GTP type and flags and TEID should be matched. If the lower four bits are 1, the type and flags field should be ignored, while if the upper four bits are 1, the TEID should be ignored. If the lower bits are 0, the type and fields flag should be matched subject to the flags in the gtp_flag_mask field, while if the upper bits are 0 the TEID should be matched. The mask is combined with the message type and header field of the packet using logical AND; the result becomes the value of the match. Only those parts of the field in which the mask has a 1 value are matched.")<br><br>Kempf at [0117] ("The gtp_type_n_flags field contains the GTP mes-sage type in the upper 8 bits and the GTP header flags in the lower 8 bits. The gtp_teid field contains the GRP TEID. When the value of the oxm_type ( oxm_class+oxm_field is GTP _ MATCH and the HM bit is zero, the flaw's GTP header must match these values exactly. If the HM flag is one, the value contains an ersmt_gtp_match field and an ermst_gtp_mask field, as specified by the OpenFlow 1.2 specification. We define ermst_gtp_mask field for selecting flows based on the settings of flag bits:<br><br>struct ermst_gtp_mask {<br>uint32_t gtp_wildcard;<br>uint16_t gtp_flag_mask;<br>};<br><br>Kempf at [0118] ("The gtp_ wildcard field indicates whether the TEID should be matched. If the value is 0xFFFFFFFF, the TEID should be matched and not the flags, if the value is 0x00000000, the flags should be matched and not the TEID. If the gtp_ wildcard indicates the flags should be matched, the gtp_flag_mask is combined with the message type and header field of the packet using logical AND, the result becomes the value of the match. Only those parts of the field in which the mask has a 1 value are matched.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 105 of 1100

| No. | '111 Patent Claim 16 | Kempf |
|---|---|---|
| | | |
| 16[c] | wherein the packet-applicable criterion is that one or more of the flag bits is set. | Kempf discloses wherein the packet-applicable criterion is that one or more of the flag bits is set.<br><br>For example, Kempf flow table matches in which the flag bits is set.<br><br>Kempf at [0081] ("In one embodiment, OpenFlow is modified to pro-vide rules for GTP TEID Routing. FIG. 17 is a diagram of one embodiment of the OpenFlow flow table modification for GTP TEID routing. An OpenFlow switch that supports TEID routing matches on the 2 byte (16 bit) collection of header fields and the 4 byte (32 bit) GTP TEID, in addition to other OpenFlow header fields, in at least one flow table ( e.g., the first flow table). The GTP TEID flag can be wildcarded (i.e. matches are "don't care"). In one embodiment, the EPC pro-tocols do not assign any meaning to TEIDs other than as an endpoint identifier for tunnels, like ports in standard UDP/ TCP transport protocols. In other embodiments, the TEIDs can have a correlated meaning or semantics. The GTP header flags field can also be wildcarded, this can be partially matched by combining the following bitmasks: 0xFF00- Match the Message Type field; 0xe0-Match the Version field; 0xl0-Match the PT field; 0x04-Match the E field; 0x02- Match the S field; and 0x0l-Match the PN field.")<br><br>Kempf at [0082] ("In one embodiment, OpenFlow can be modified to support virtual ports for fast path GTP TEID encapsulation and decapsulation. An OpenFlow mobile gateway can be used to support GTP encapsulation and decapsulation with virtual ports. The GTP encapsulation and decapsulation virtual ports can be used for fast encapsulation and decapsulation of user data packets within GTP-U tunnels, and can be designed simply enough that they can be implemented in hardware or firmware. For this reason, GTP virtual ports may have the following restrictions on traffic they will handle: Protocol Type (PT) field= 1, where GTP encapsulation ports only sup-port GTP, not GTP' (PT field=0); Extension Header flag (E)=0, where no extension headers are supported, Sequence Number flag (S)=0, where no sequence numbers are sup-ported; N-PDU flag (PN)=0; and Message |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 106 of 1100

| No. | '111 Patent Claim 16 | Kempf |
|---|---|---|
| | | type=255, where Only G-PDU messages, i.e. tunneled user data, is supported in the fast path.")<br><br>Kempf at [0083] ("If a packet either needs encapsulation or arrives encapsulated with nonzero header flags, header extensions, and/or the GTP-U packet is not a G-PDU packet (i.e. it is a GTP-U control packet), the processing must proceed via the gateway's slow path (software) control plane. GTP-C and GTP' packets directed to the gateway's IP address are a result of mis-configuration and are in error. They must be sent to the OpenFlow controller, since these packets are handled by the S-GW-C and P-GW-C control plane entities in the cloud computing system or to the billing entity handling GTP' and not the S-GW-D and P-GW-D data plane switches.")<br><br>Kempf at [0088] ("To support slow path encapsulation, the software control plane on the switch maintains a hash table with keys calculated from the GTP-U TEID. The TEID hash keys are calculated using a suitable hash algorithm with low collision frequency, for example SHA-1. The flow table entries contain a record of how the packet header, including the GTP encap-sulation header, should be configured. This includes: the same header fields as for the hardware or firmware encapsu-lation table in FIG.18; values for the GTP header flags (PT, E, S, and PN); the sequence number and/or the N-PDU number if any; if the E flag is 1, then the flow table contains a list of the extension headers, including their types, which the slow path should insert into the GTP header.")<br><br>Kempf at [0092] ("In one embodiment, the system implements a GTP fast path decapsulation virtual port. When requested by the S-GW and P-GW control plane software running in the cloud computing system, the gateway switch installs rules and actions for routing GTP encapsulated packets out of GTP tunnels. The rules match the GTP header flags and the GTP TEID for the packet, in the modified OpenFlow flow table shown in FIG. 17 as follows: the IP destination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); the UDP destination port is the GTP-U destination port (2152); and the header fields and message type field is wildcarded with the flag 0XFFF0 and the upper two bytes of the field match the G-PDU message type (255) while the lower two bytes match 0x30, i.e. the packet is a GTP packet not a GTP' packet and the version number is 1.") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 107 of 1100

| No. | '111 Patent Claim 16 | Kempf |
|---|---|---|
| | | Kempf at [0094] ("In one embodiment, the system implements han-dling of GTP-U control packets. The OpenFlow controller programs the gateway switch flow tables with 5 rules for each gateway switch IP address used for GTP traffic. These rules contain specified values for the following fields: the IP des-tination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); the UDP destination port is the GTP-U destination port (2152); the GTP header flags and message type field is wildcarded with 0xFFF0; the value of the header flags field is 0x30, i.e. the version number is 1 and the PT field is 1; and the value of the message type field is one of 1 (Echo Request), 2 (Echo Response), 26 (Error Indication), 31 (Support for Extension Headers Notification), or 254 (End Marker).")<br><br>Kempf at [0098] ("The header flags and message type fields for the three rules are wildcarded with the following bitmasks and match as follows: bitmask 0xFFF4 and the upper two bytes match the G-PDU message type (255) while the lower two bytes are Ox34, indicating that the version number is 1, the packet is a GTP packet, and there is an extension header present; bitmask 0xFFFF2 and the upper two bytes match the G-PDU message type (255) while the lower two bytes are 0x32, indicating that the version number is 1, the packet is a GTP packet, and there is a sequence number present; and bitmask 0xFF0l and the upper two bytes match the G-PDU message type (255) while the lower two bytes are 0x31, indicating that the version number is 1, the packet is a GTP packet, and a N-PDU is present.")<br><br>Kempf at [0114] ("The gtp_type_n_flags field contains the GTP mes-sage type in the upper 8 bits and the GTP header flags in the lower 8 bits. The gtp_teid field contains the GTP TEID. The gtp_ wildcard field indicates whether the GTP type and flags and TEID should be matched. If the lower four bits are 1, the type and flags field should be ignored, while if the upper four bits are 1, the TEID should be ignored. If the lower bits are 0, the type and fields flag should be matched subject to the flags in the gtp_flag_mask field, while if the upper bits are 0 the TEID should be matched. The mask is combined with the message type and header field of the packet using logical AND; the result becomes the value of the match. Only those parts of the field in which the mask has a 1 value are matched.")<br><br>Kempf at [0117] ("The gtp_type_n_flags field contains the GTP mes-sage type in the upper 8 bits and the GTP header flags in the lower 8 bits. The gtp_teid field contains the GRP |

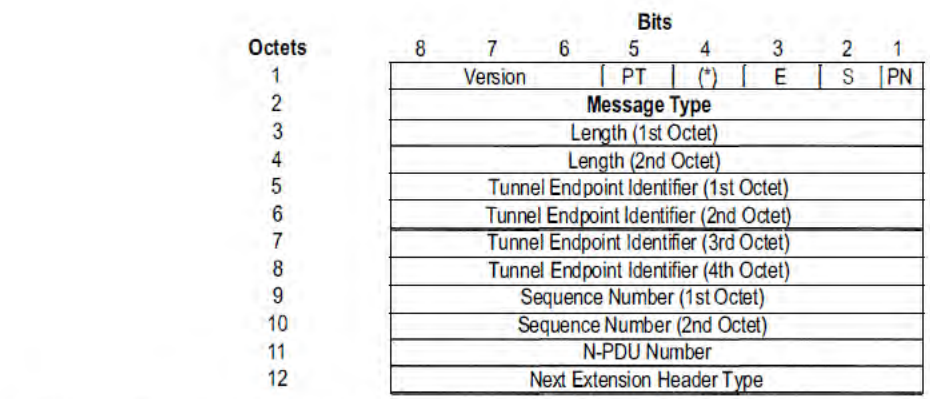


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 108 of 1100

| No. | '111 Patent Claim 16 | Kempf |
|---|---|---|
| | | TEID. When the value of the oxm_type ( oxm_class+oxm_field is GTP _ MATCH and the HM bit is zero, the flaw's GTP header must match these values exactly. If the HM flag is one, the value contains an ersmt_gtp_match field and an ermst_gtp_mask field, as specified by the OpenF!ow 1.2 specification. We define ermst_gtp_mask field for selecting flows based on the settings of flag bits:<br><br>struct ermst_gtp_mask {<br>uint32_t gtp_wildcard;<br>uint16_t gtp_flag_mask;<br>};<br><br>Kempf at [0118] ("The gtp_ wildcard field indicates whether the TEID should be matched. If the value is 0xFFFFFFFF, the TEID should be matched and not the flags, if the value is 0x00000000, the flags should be matched and not the TEID. If the gtp_ wildcard indicates the flags should be matched, the gtp_flag_mask is combined with the message type and header field of the packet using logical AND, the result becomes the value of the match. Only those parts of the field in which the mask has a 1 value are matched.")<br><br>Kempf at Figure 10 |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 109 of 1100

| No. | '111 Patent Claim 16 | Kempf |
|---|---|---|
| | |  |

| Octets | Bits 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
|---|---|---|---|---|---|---|---|---|
| 1 | Version | | | PT | (*) | E | S | PN |
| 2 | Message Type | | | | | | | |
| 3 | Length (1st Octet) | | | | | | | |
| 4 | Length (2nd Octet) | | | | | | | |
| 5 | Tunnel Endpoint Identifier (1st Octet) | | | | | | | |
| 6 | Tunnel Endpoint Identifier (2nd Octet) | | | | | | | |
| 7 | Tunnel Endpoint Identifier (3rd Octet) | | | | | | | |
| 8 | Tunnel Endpoint Identifier (4th Octet) | | | | | | | |
| 9 | Sequence Number (1st Octet) | | | | | | | |
| 10 | Sequence Number (2nd Octet) | | | | | | | |
| 11 | N-PDU Number | | | | | | | |
| 12 | Next Extension Header Type | | | | | | | |

NOTE 0: (*) This bit is a spare bit. It shall be sent as '0'. The receiver shl not evaluate this bit,
NOTE 1: 1) This field shall only be evaluated when indicated by the S flag set to 1.
NOTE 2: 2) This field shall only be evaluated when indicated by the PN flag set to 1.
NOTE 3: 3) This field shall only be evaluated when indicated by the E flag set to 1.
NOTE 4: 4) This field shall be present if and only if any one or more of the S. PN and E flags are set.

FIG. 10

| No. | '111 Patent Claim 17 | Kempf |
|---|---|---|
| 17[a] | The method according to claim 16, wherein the packet is an Transmission Control Protocol (TCP) packet, and | Kempf discloses the method according to claim 16, wherein the packet is an Transmission Control Protocol (TCP) packet.<br><br>For example, Kempf discloses packets in a network that are part of the Transmission Control Protocol.<br><br>*See supra* at Claim 16.<br><br>Kempf at [0046] ("A rule 201 contains key fields from several headers in the protocol stack, for example source and destination Ethernet MAC addresses, source and destination IP addresses, IP protocol type number, incoming and outgoing TCP or UDP port numbers. To define a flow, all the available matching fields may be used. But it is also possible to restrict |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 110 of 1100

| No. | '111 Patent Claim 17 | Kempf |
|---|---|---|
| | | the matching rule to a subset of the available fields by using wildcards for the unwanted fields.")<br><br>Kempf at [0081] ("In one embodiment, OpenFlow is modified to pro-vide rules for GTP TEID Routing. FIG. 17 is a diagram of one embodiment of the OpenFlow flow table modification for GTP TEID routing. An OpenFlow switch that supports TEID routing matches on the 2 byte (16 bit) collection of header fields and the 4 byte (32 bit) GTP TEID, in addition to other OpenFlow header fields, in at least one flow table ( e.g., the first flow table). The GTP TEID flag can be wildcarded (i.e. matches are "don't care"). In one embodiment, the EPC pro-tocols do not assign any meaning to TEIDs other than as an endpoint identifier for tunnels, like ports in standard UDP/ TCP transport protocols. In other embodiments, the TEIDs can have a correlated meaning or semantics. The GTP header flags field can also be wildcarded, this can be partially matched by combining the following bitmasks: 0xFF00- Match the Message Type field; 0xe0-Match the Version field; 0xl0-Match the PT field; 0x04-Match the E field; 0x02- Match the S field; and 0x0l-Match the PN field.")<br><br>Kempf at [0089] ("In one embodiment, the system implements a GTP fast path encapsulation virtual port. When requested by the S-GW-C and P-GW-C control plane software running in the cloud computing system, the OpenFlow controller programs the gateway switch to install rules, actions, and TEID hash table entries for routing packets into GTP tunnels via a fast path GTP encapsulation virtual port. The rules match the packet filter for the input side of GTP tunnel's bearer. Typi-cally this will be a 4 tuple of: IP source address; IP destination address; UDP/TCP/SCTP source port; and UDP/TCP/SCTP destination port. The IP source address and destination address are typically the addresses for user data plane traffic, i.e. a UE or Internet service with which a UE is transacting, and similarly with the port numbers. For a rule matching the GTP-U tunnel input side, the associated instructions and are the following:<br><br>Write-Metadata ( GTP-TEID, OxFFFFFFFF)<br>Apply-Actions (Set-Output-Port GTP-Encap-VP")<br><br>Kempf at [0101] ("In one embodiment, the system implements han-dling of user data plane packets requiring GTP-U encapsula-tion with extension headers, sequence numbers, and N- |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 111 of 1100

| No. | '111 Patent Claim 17 | Kempf |
|---|---|---|
| | | PDU numbers. User data plane packets that require extension head-ers, sequence numbers, or N-PDU numbers during GTP encapsulation require special handling by the software slow path. For these packets, the OpenFlow controller programs a rule matching the 4 tuple: IP source address; IP destination address; UDP/TCP/SCTP source port; and UDP/TCP/SCTP destination port. The instructions for matching packets are:<br><br>Write-Metadata ( GTP-TEID, 0x FFFFFFFF)<br>Apply-Actions (Set-Output-Port LOCAL_GTP _U_ENCAP )") |
| 17[b] | wherein the one or more flag bits comprises comprise a SYN flag bit, an ACK flag bit, a FIN flag bit, a RST flag bit, or any combination thereof. | Kempf discloses wherein the one or more flag bits comprises comprise a SYN flag bit, an ACK flag bit, a FIN flag bit, a RST flag bit, or any combination thereof.<br><br>For example, Kempf discloses packet headers with flag bits. A person of ordinary skill in the art would understand that such flag bits can comprise a SYN flag bit, an ACK flag bit, a FIN flag bit, a RST flag bit, or any combination thereof. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Kempf is found to not meet this limitation, wherein the one or more flag bits comprises comprise a SYN flag bit, an ACK flag bit, a FIN flag bit, a RST flag bit, or any combination thereof would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>Kempf at [0081] ("In one embodiment, OpenFlow is modified to pro-vide rules for GTP TEID Routing. FIG. 17 is a diagram of one embodiment of the OpenFlow flow table modification for GTP TEID routing. An OpenFlow switch that supports TEID routing matches on the 2 byte (16 bit) collection of header fields and the 4 byte (32 bit) GTP TEID, in addition to other OpenFlow header fields, in at least one flow table ( e.g., the first flow table). The GTP TEID flag can be wildcarded (i.e. matches are "don't care"). In one embodiment, the EPC pro-tocols do not assign any meaning to TEIDs other than as an endpoint identifier for tunnels, like ports in standard UDP/ TCP transport protocols. In other embodiments, the TEIDs can have a correlated meaning or semantics. The GTP header flags field can also be wildcarded, this can be partially matched by combining the following bitmasks: 0xFF00- Match the Message Type field; 0xe0-Match the Version field; 0xl0- |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 112 of 1100

| No. | '111 Patent Claim 17 | Kempf |
|---|---|---|
| | | Match the PT field; 0x04-Match the E field; 0x02- Match the S field; and 0x0l-Match the PN field.")<br><br>Kempf at [0082] ("In one embodiment, OpenFlow can be modified to support virtual ports for fast path GTP TEID encapsulation and decapsulation. An OpenFlow mobile gateway can be used to support GTP encapsulation and decapsulation with virtual ports. The GTP encapsulation and decapsulation virtual ports can be used for fast encapsulation and decapsulation of user data packets within GTP-U tunnels, and can be designed simply enough that they can be implemented in hardware or firmware. For this reason, GTP virtual ports may have the following restrictions on traffic they will handle: Protocol Type (PT) field= 1, where GTP encapsulation ports only sup-port GTP, not GTP' (PT field=0); Extension Header flag (E)=0, where no extension headers are supported, Sequence Number flag (S)=0, where no sequence numbers are sup-ported; N-PDU flag (PN)=0; and Message type=255, where Only G-PDU messages, i.e. tunneled user data, is supported in the fast path.")<br><br>Kempf at [0083] ("If a packet either needs encapsulation or arrives encapsulated with nonzero header flags, header extensions, and/or the GTP-U packet is not a G-PDU packet (i.e. it is a GTP-U control packet), the processing must proceed via the gateway's slow path (software) control plane. GTP-C and GTP' packets directed to the gateway's IP address are a result of mis-configuration and are in error. They must be sent to the OpenFlow controller, since these packets are handled by the S-GW-C and P-GW-C control plane entities in the cloud computing system or to the billing entity handling GTP' and not the S-GW-D and P-GW-D data plane switches.")<br><br>Kempf at [0088] ("To support slow path encapsulation, the software control plane on the switch maintains a hash table with keys calculated from the GTP-U TEID. The TEID hash keys are calculated using a suitable hash algorithm with low collision frequency, for example SHA-1. The flow table entries contain a record of how the packet header, including the GTP encap-sulation header, should be configured. This includes: the same header fields as for the hardware or firmware encapsu-lation table in FIG.18; values for the GTP header flags (PT, E, S, and PN); the sequence number and/or the N-PDU number if any; if the E flag is 1, then the flow table contains a list of the extension headers, including their types, which the slow path should insert into the GTP header.") |

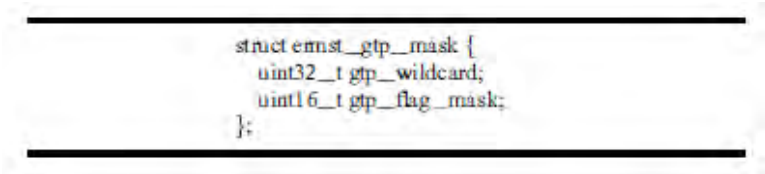


~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 113 of 1100

| No. | '111 Patent Claim 17 | Kempf |
|---|---|---|
| | | Kempf at [0092] ("In one embodiment, the system implements a GTP fast path decapsulation virtual port. When requested by the S-GW and P-GW control plane software running in the cloud computing system, the gateway switch installs rules and actions for routing GTP encapsulated packets out of GTP tunnels. The rules match the GTP header flags and the GTP TEID for the packet, in the modified OpenFlow flow table shown in FIG. 17 as follows: the IP destination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); the UDP destination port is the GTP-U destination port (2152); and the header fields and message type field is wildcarded with the flag 0XFFF0 and the upper two bytes of the field match the G-PDU message type (255) while the lower two bytes match 0x30, i.e. the packet is a GTP packet not a GTP' packet and the version number is 1.")<br><br>Kempf at [0094] ("In one embodiment, the system implements han-dling of GTP-U control packets. The OpenFlow controller programs the gateway switch flow tables with 5 rules for each gateway switch IP address used for GTP traffic. These rules contain specified values for the following fields: the IP des-tination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); the UDP destination port is the GTP-U destination port (2152); the GTP header flags and message type field is wildcarded with 0xFFF0; the value of the header flags field is 0x30, i.e. the version number is 1 and the PT field is 1; and the value of the message type field is one of 1 (Echo Request), 2 (Echo Response), 26 (Error Indication), 31 (Support for Extension Headers Notification), or 254 (End Marker).")<br><br>Kempf at [0098] ("The header flags and message type fields for the three rules are wildcarded with the following bitmasks and match as follows: bitmask 0xFFF4 and the upper two bytes match the G-PDU message type (255) while the lower two bytes are Ox34, indicating that the version number is 1, the packet is a GTP packet, and there is an extension header present; bitmask 0xFFFF2 and the upper two bytes match the G-PDU message type (255) while the lower two bytes are 0x32, indicating that the version number is 1, the packet is a GTP packet, and there is a sequence number present; and bitmask 0xFF0l and the upper two bytes match the G-PDU message type (255) while the lower two bytes are 0x31, indicating that the version number is 1, the packet is a GTP packet, and a N-PDU is present.") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 114 of 1100

| No. | '111 Patent Claim 17 | Kempf |
|---|---|---|
| | | Kempf at [0114] ("The gtp_type_n_flags field contains the GTP mes-sage type in the upper 8 bits and the GTP header flags in the lower 8 bits. The gtp_teid field contains the GTP TEID. The gtp_ wildcard field indicates whether the GTP type and flags and TEID should be matched. If the lower four bits are 1, the type and flags field should be ignored, while if the upper four bits are 1, the TEID should be ignored. If the lower bits are 0, the type and fields flag should be matched subject to the flags in the gtp_flag_mask field, while if the upper bits are 0 the TEID should be matched. The mask is combined with the message type and header field of the packet using logical AND; the result becomes the value of the match. Only those parts of the field in which the mask has a 1 value are matched.")<br><br>Kempf at [0117] ("The gtp_type_n_flags field contains the GTP mes-sage type in the upper 8 bits and the GTP header flags in the lower 8 bits. The gtp_teid field contains the GRP TEID. When the value of the oxm_type ( oxm_class+oxm_field is GTP _ MATCH and the HM bit is zero, the flaw's GTP header must match these values exactly. If the HM flag is one, the value contains an ersmt_gtp_match field and an ermst_gtp_mask field, as specified by the OpenFlow 1.2 specification. We define ermst_gtp_mask field for selecting flows based on the settings of flag bits:<br><br>  <br>struct ermst_gtp_mask {<br>uint32_t gtp_wildcard;<br>uint16_t gtp_flag_mask;<br>};<br><br>Kempf at [0118] ("The gtp_ wildcard field indicates whether the TEID should be matched. If the value is 0xFFFFFFFF, the TEID should be matched and not the flags, if the value is 0x00000000, the flags should be matched and not the TEID. If the gtp_ wildcard indicates the flags should be matched, the gtp_flag_mask is combined with the message type and header field of the packet using logical AND, the result becomes the value of the match. Only those parts of the field in which the mask has a 1 value are matched.")<br><br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Kempf in combination with (1) the knowledge of a person of ordinary skill in the art, alone |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 115 of 1100

| No. | '111 Patent Claim 17 | Kempf |
|---|---|---|
| | | or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 17[b] of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references.<br><br>For example, Copeland discloses TCP packets with flag bits including SYN, ACK, FIN, and R flag bits, i.e., wherein the one or more flag bits comprise a SYN flag bit, an ACK flag bit, a FIN flag bit, a RST flag bit, or any combination thereof.<br><br>Copeland at [0081] ("In a TCP/IP datagram 210, the initial data of the IP datagram is the TCP header 230 information. The initial TCP header 230 information includes the 16-bit source and 16-bit destination port numbers. A 32-bit sequence number for the data in the packet follows the port numbers. Following the sequence number is a 32-bit acknowledgement number. If an ACK flag (discussed below) is set, this number is the next sequence number the sender of the packet expects to receive. Next is a 4-bit data offset, which is the number of 32-bit words in the TCP header. A 6-bit reserved field follows.")<br><br>Copeland at [0082] ("Following the reserved field, the next 6 bits are a series of one-bit flags, shown in FIG. 2 as flags U, A, P, R, S, F. The first flag is the urgent flag (U). If the U flag is set, it indicates that the urgent pointer is valid and points to urgent data that should be acted upon as soon as possible. The next flag is the A ( or ACK or "acknowledgment") flag. The ACK flag indicates that an acknowledgment number is valid, and acknowledges that data has been received. The next flag, the push (P) flag, tells the receiving end to push all buffered data to the receiving application. The reset (R) flag is the following flag, which terminates both ends of the TCP connection. Next, the S (or SYN for "synchronize") flag is set in the initial packet of a TCP connection where both ends have to synchronize their TCP buffers. Following the SYN flag is the F (for FIN or "finish") flag. This flag signifies that the sending end of the communication and the host will not send any more data but still may acknowledge data that is received.")<br><br>Copeland at [0089] ("FIG. 3 illustrates an exemplary TCP/IP session 300. As discussed in reference to FIG. 2, the SYN flag is set whenever one host initiates a session with another host. In the initial packet, Hostl sends a message with only the SYN flag set. The SYN flag is designed to establish a TCP connection and allow both ends to synchronize their TCP buffers. Hostl provides the sequence of the first data packet it will send.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 116 of 1100

| No. | '111 Patent Claim 17 | Kempf |
|---|---|---|
| | | Copeland at [0090] ("Host2 responds with a SYN-ACK packet. In this message, both the SYN flag and the ACK flag are set. Host2 provides the initial sequence number for its data to Hostl. Host2 also sends to Hostl the acknowledgment number that is the next sequence number Host2 expects to receive from host 1. In the SYN-ACK packet sent by Host2, the acknowl-edgment number is the initial sequence number of Hostl plus 1, which should be the next sequence number received.")<br><br>Copeland at [0091] ("Hostl responds to the SYN-ACK with a packet with just the ACK flag set. Hostl acknowledges that the next packet of information received from Host2 will be Host2's initial sequence number plus 1. The three-way handshake is complete and data is transferred.")<br><br>Copeland at [0092] ("Host2 responds to ACK packet with its own ACK packet. Host2 acknowledges the data it has received from Hostl by sending an acknowledgment number one greater than its last received data sequence number. Both hosts send packets with the ACK flag set until the session is to end although the P and U flags may also be set, if warranted.")<br><br>Copeland at [0093] ("As illustrated, when Hostl terminates its end of the session, it sends a packet with the FIN and ACK flags set. The FIN flag informs Host2 that Hostl will send no more data. The ACK flag acknowledges the last data received by Hostl by informing Host2 of the next sequence number it expects to receive.")<br><br>Copeland at [0094] ("Host2 acknowledges the FIN packet by sending its own ACK packet. The ACK packet has the acknowledge-ment number one greater than the sequence number of Hostl's FIN-ACK packet. ACK packets are still delivered between the two hosts, except that HOSTl's packets have no data appended to the TCP/IP end of the headers.")<br><br>Copeland at [0095] ("When Host 2 is ready to terminate the session, it sends its own packet with the FIN and ACK flags set. Hostl responds that it has received the final packet with an ACK packet providing to Host2 an acknowledgment number one greater than the sequence number provided in the FIN-ACK packet of Host2.") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 117 of 1100

| No. | '111 Patent Claim 17 | Kempf |
|---|---|---|
| | | As another example, Uchida discloses the TCP (Transmission Control Protocol) FIN flag, RST flag, and SYN flag *i.e.,* the one or more flag bits comprises comprise a SYN flag bit, an ACK flag bit, a FIN flag bit, a RST flag bit.<br><br>Uchida at [0040] ("A flow end can be detected by various methods as below. For example, in one method, a protocol end message is checked. For example, in the TCP (Transmission Control Protocol), a FIN flag is checked. In this way, the end of communication, that is, the end of a flow using communica-tion, can be detected. In practice, after a FIN flag, communi-cation with an ACK packet is generated in a reverse-direction flow (a flow in which the source and the destination are reversed). Thus, by detecting the ACK flag in the reverse-direction flow after the FIN packet, a flow end can be deter-mined. Further, since the TCP is used in bidirectional com-munication, the forward- and reverse-direction flows can be used as a pair to determine a flow end. Namely, if the end of a flow is detected, a process rule corresponding to the reverse-direction flow of the flow can also be determined to be unnec-essary. Alternatively, a communication end can also be deter-mined when a predetermined time elapses after reception of a SYN packet and a timeout is determined. Still alternatively, a communication end can be determined by reception of a RST packet. These methods will be described in more detail later as specific examples.")<br><br>Uchida at [0050] ("The flow end check unit can use at least one of a TCP (Transmission Control Protocol) FIN flag, RST flag, and SYN flag extracted by the end determination information extraction unit to determine a flow end.")<br><br>Uchida at [0055] ("In the process rule update method, a flow end can be determined by at least one of a TCP (Transmission Control Protocol) FIN flag, RST flag, and SYN flag.")<br><br>Uchida at [0102] ("Next, specific examples 1 to 3 will be described. In the examples 1 to 3, a flow end is determined by combining features of the above individual exemplary embodiments and using TCP (Transmission Control Protocol) flags.")<br><br>Uchida at [0103] ("FIG. 6 is a state transition diagram of TCP connec-tion. "CLOSED" at the top of FIG. 6 represents the end of TCP communication, and portions connected thereto repre-sent states prior to the end of TCP communication. Approxi-mately 2MSL (MSL: Maximum Segment Lifetime) is the maximum amount of time required to reach the above |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 118 of 1100

| No. | '111 Patent Claim 17 | Kempf |
|---|---|---|
| | | "CLOSED," that is, if the packet forwarding apparatus stands by for approximately 2MSL after both FINs flow, the above "CLOSED" is reached. Thus, after a FIN is confirmed in either direction, if this 2MSL elapses, basically, a communi-cation end can be determined. Even if the state does not change smoothly because of packet loss or the like (for example, even if an ACK packet does not arrive after "CLOS-ING"), a retransmitted packet is forwarded immediately after this 2MSL. Thus, the end of TCP communication can be determined if a new FIN packet is not received within the time corresponding to the 2MSL and a margin (2MSL+a) at long-est.")<br><br>Uchida at [0104] ("Hereinafter, the description will be made, assuming that a packet forwarding apparatus Cl according to the present invention relays TCP communication between a com-puter (client) Dl 0 and a server D20 that use network configu-rations illustrated in FIG. 7. In the example of FIG. 7, the computer Dl0 belongs to a network represented by 192.168. 0./24 and is set by 192.168.0.10. The server D20 belongs to a network represented by 192.168.1./24 and is set by 192.168. 1.10. As in the case of the OpenFlow controller described in Non-Patent Documents 1 and 2, a control apparatus ( control-ler) Dl is connected to the packet forwarding apparatus Cl via a dedicated channel and manages connection between the two networks. In the following description, the control appa-ratus (controller) Dl controls the packet forwarding appara-tus Cl so that connection from other networks appears as communication from network number 1 (192.168.1.1) of the respective networks (see process rule actions in FIG. 19). In addition, in the present specific example, since FIN packets are monitored, the end determination information extraction unit Cl 7 monitors a protocol stack, including: fields in which the TCP is determined; and the FIN flag in the TCP header.")<br><br>Uchida at [0105] ("FIG. 8 is a flow chart of a flow end determination process using FIN flags. In FIG. 8, steps relating to a timeout determination are added to steps Slll to S116 in the flow chart in FIG. 3. Thus, the flow chart in FIG. 8 includes more detailed steps than the flow chart of FIG. 3. Hereinafter, operations will be described with reference to FIGS. 3, 6, and 8 and FIGS. 9 to 13. In practice, prior to TCP/IP communi-cation, ARP (Address Resolution Protocol) communication is executed, and a process rule may be set in that stage. However, for ease of description, description of the ARP communication will be omitted. The following description will be made based on communication at the TCP/IP level.") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 119 of 1100

| No. | '111 Patent Claim 17 | Kempf |
|---|---|---|
| | | Uchida at [0106] ("First, the computer Dl0 starts communication with the server D20. For an initial establishment of communica-tion, a packet (SYN) is inputted to the packet forwarding apparatus Cl (start of ACTIVE OPEN through SYN forward-ing in FIG. 6). The packet reception unit Cl0 receives and stores this first packet in the packet storage unit Cll (steps SlOl to S102 in FIG. 3).")<br><br>Uchida at [0107] ("The packet reception unit C10 notifies the packet process information extraction unit C12 and the end determination information extraction unit C17 of reception of the packet. The packet process information extraction unit C12 refers to the packet storage unit C11 and extracts information such as IP source and destination information that is necessary to search for a process rule (step S103 in FIG. 3). Hereinafter, a process corresponding to steps S103 to S110 in FIG. 3 will be executed.")<br><br>Uchida at [0115] ("Upon receiving a notification that the packet has been received by the packet reception unit Cl 0, the end deter-mination information extraction unit Cl 7 refers to the packet storage unit Cll, monitors a TCP FIN flag, and finds a FIN flag (step S201 in FIG. 8).")<br><br>Uchida at [0116] ("Since a FIN flag is set, the end determination infor-mation extraction unit Cl 7 determines that the packet includes information necessary for determining a flow end. Thus, the end determination information extraction unit Cl 7 extracts information for identifying a process rule to be deleted (the ingress port is 1; the source address is 192.168. 0.10; the destination is 192.168.1.10; and the protocol is TCP (the type is Ox0006)) and stands by until forwarding of the packet. Upon receiving a notification that the packet has been transmitted by the packet forwarding unit C16, the end deter-mination information extraction unit Cl 7 further extracts information for identifying a process rule to be deleted from the packet storage unit Cll. Since the IP address is replaced, the extracted information for identifying a process rule to be deleted represents that the source address is 192.168.1.1; the destination is 192.168.1.1 0; and the protocol is TCP (the type is 0x0006). The information is used for marking of the reverse flow. The end determination information extraction unit Cl 7 notifies the flow end check unit C18 of the notification that the FIN packet has been received and these items of information (step S202 in FIG. 8).") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 120 of 1100

| No. | '111 Patent Claim 17 | Kempf |
|---|---|---|
| | | Uchida at [0117] ("Upon receiving the above information from the end determination information extraction unit Cl 7, the flow end check unit C18 checks whether or not a FIN flag is set in a predetermined packet header position (step S203). These steps correspond to steps Slll to S114 in FIG. 3.")<br><br>Uchida at [0121] ("Next, after an ACK reply in response to the FIN packet from the computer DlO is forwarded from the server D20 in the same way as the above normal packet (start of PASSIVE CLOSE in FIG. 6), the server D20 transmits a FIN packet to the computer DlO. When this FIN packet is inputted to the packet forwarding apparatus Cl, the flow end determi-nation process from steps Slll to S116 is started, as in the case of the above start of ACTIVE CLOSE.")<br><br>Uchida at [0122] ("Upon receiving a notification that the packet has been received from the packet reception unit Cl0, the end determination information extraction unit Cl 7 refers to the packet storage unit Cll, monitors a TCP FIN flag, and finds a FIN packet (step S201 in FIG. 8).")<br><br>Uchida at [0123] ("Since a FIN flag is set, the end determination infor-mation extraction unit Cl 7 determines that the packet includes information necessary for determining a flow end. Thus, the end determination information extraction unit Cl 7 extracts information for identifying a process rule to be deleted (the ingress port is 2; the source address is 192.168. 1.10; the destination is 192.168.1.1; and the protocol is TCP (the type is Ox.0006)) and stands by until the packet is trans-mitted. Upon receiving a notification that the packet has been transmitted from the packet forwarding unit C16, the end determination information extraction unit Cl 7 further extracts information for identifying a modified process rule from the packet storage unit Cll. Since the IP address is replaced, the extracted information for identifying a modified process rule represents that the source address is 192.168.1. 10; the destination is 192.168.0.10; and the protocol is TCP (the type is 0x0006). The information is used for marking of the reverse flow. The end determination information extrac-tion unit Cl 7 notifies the flow end check unit C18 of the notification that the FIN packet has been received and these items of information (step S202 in FIG. 8).") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 121 of 1100

| No. | '111 Patent Claim 17 | Kempf |
|---|---|---|
| | | Uchida at [0124] ("Upon receiving the above information from the end determination information extraction unit Cl 7, the flow end check unit C18 checks whether or not a FIN flag is set in a predetermined packet header position (step S203 in FIG. 8). These steps correspond to steps Slll to S114 in FIG. 3.")<br><br>Uchida at [0125] ("At this point, since a FIN packet has been transmit-ted, the flow end check unit C18 uses the information for identifying a process rule to be deleted as a key, extracts the process rule (process rule corresponding to ingress port 2 in FIG. 11) from the process rule storage unit C13, and marks a FIN packet reception flag (steps S204 to S205 in FIG. 8). This process corresponds to the internal state update process in step S115 in FIG. 3.")<br><br>Uchida at [0134] ("Referring back to the state transition diagram of TCP connection in FIG. 6, there are two cases where "CLOSED" at the top of FIG. 6 is reached without a state transition involving FIN flags. One case arises when the ses-sion is closed from SYN_SENT, which is reached when a SYN packet in which a SYN flag is marked is transmitted. The other case arises when a timeout is generated. In such case, while the packet forwarding apparatus cannot monitor the closed session, the packet forwarding apparatus can con-firm a timeout in the following way. In the present specific example, a flow end is determined by this timeout.")<br><br>Uchida at [0135] ("n the present specific example, if a SYN/ ACK packet does not flow in a direction opposite to the SYN packet flow direction within a predetermined time (from "SYN_ RCVD" to "SYN_SENT" in FIG. 6), a timeout is determined.")<br><br>Uchida at [0136] ("FIG. 14 is a flow chart illustrating a flow end deter-mination process using a SYN flag. Since the basic operations are the same as those of the above specific example 1, the following description will be made with a focus on the dif-ference.")<br><br>Uchida at [0137] ("In FIG. 14, upon receiving a notification that the packet has been received by the packet reception unit ClO, the end determination information extraction unit Cl 7 refers to the packet storage, unit Cll, monitors a TCP SYN flag, and finds a SYN packet (step S301 in FIG. 14).") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 122 of 1100

| No. | '111 Patent Claim 17 | Kempf |
|---|---|---|
| | | Uchida at [0138] ("Since a SYN flag is set, the end determination infor-mation extraction unit Cl 7 determines that the packet includes information necessary for determining a flow end. Thus, the end determination information extraction unit Cl 7 extracts information for identifying a process rule to be deleted (the ingress port is 2; the source address is 192.168. 1.10; the destination is 192.168.1.1; and the protocol is TCP (the type is Ox.0006)) and stands by until the packet is trans-mitted. Upon receiving a notification that the packet has been transmitted by the packet forwarding unit C16, the end deter-mination information extraction unit Cl 7 further extracts information for identifying a modified process rule from the packet storage unit Cll. Since the IP address is replaced, the extracted information for identifying a process rule repre-sents that the source address is 192.168.1.10; the destination is 192.168.0.10; and the protocol is TCP (the type is 0x0006). The information is used for marking of the reverse flow. The end determination information extraction unit Cl 7 notifies the flow end check unit C18 of the notification that the SYN packet has been received and these items of information (step S302 in FIG. 14).")<br><br>Uchida at [0139] ("Upon receiving the above information from the end determination information extraction unit Cl 7, the flow end check unit C18 checks whether a SYN flag is set in a prede-termined packet header position and an ACK flag is not marked (step S303 in FIG. 14). These steps correspond to steps Slll to S114 in FIG. 3.")<br><br>Uchida at [0148] (" Next, a third specific example in which a flow end determination is executed by using a TCP RST (reset) flag will be described.")<br><br>Uchida at [0149] ("Referring back to the state transition diagram of TCP connection in FIG. 6, there is a transition from "SYN_ RCVD," which is a communication establishment standby state, to "LISTEN," which is a communication standby state. A TCP RST (reset) flag signifies release of connection and retry of communication. Namely, since a RST packet in which this RST flag is set signifies invalidation of communi-cation, by detecting this RST flag, a flow end can be deter-mined.")<br><br>Uchida at [0150] ("FIG. 16 is a first flow chart illustrating a flow end determination process using a RST flag. Since the basic operations are the same as those of the above specific example 1, the following description will be made with a focus on the difference.") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 123 of 1100

| No. | '111 Patent Claim 17 | Kempf |
|---|---|---|
| | | Uchida at [0151] ("In FIG. 16, upon receiving a notification that the packet has been received by the packet reception unit ClO, the end determination information extraction unit Cl 7 refers to the packet storage unit Cll, monitors a TCP RST flag, and finds a RST packet (step S401 in FIG. 16).")<br><br>Uchida at [0152] ("Since a RST flag is set, the end determination infor-mation extraction unit Cl 7 determines that the packet includes information necessary for determining a flow end. Thus, the end determination information extraction unit Cl 7 extracts information for identifying a process rule to be deleted (the ingress port is 2; the source address is 192.168. 1.10; the destination is 192.168.1.1; and the protocol is TCP (the type is Ox0006)) and stands by until the packet is trans-mitted. Upon receiving a notification that the packet has been transmitted from the packet forwarding unit C16, the end determination information extraction unit Cl 7 notifies the flow end check unit C18 of the notification that the RST packet has been received and these items of information ( step S402 in FIG. 16).")<br><br>Uchida at [0164] ("For example, in a specific example of the present invention, certain TCP flags are monitored. A single packet forwarding apparatus can monitor these flags in a parallel fashion. For example, after a packet that triggers a flow end is detected, the above process may be allowed to branch to the above FIGS. 8, 14, and 16 (17) to realize parallel monitoring.") |

| No. | '111 Patent Claim 18 | Kempf |
|---|---|---|
| 18[a] | The method according to claim 1, wherein the packet comprises distinct header and payload fields, | Kempf discloses the method according to claim 1, wherein the packet comprises distinct header and payload fields.<br><br>*See supra* at Claim 1, 15[a]. |
| 18[b] | the header comprises at least the first and second entities addresses in the packet network, and | Kempf discloses the header comprises at least the first and second entities addresses in the packet network.<br><br>For example, Kempf discloses headers with source and destination addresses of the electronic devices in the network in which the packet is sent. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 124 of 1100

| No. | '111 Patent Claim 18 | Kempf |
|---|---|---|
| | | Kempf at [0046] ("A rule 201 contains key fields from several headers in the protocol stack, for example source and destination Ethernet MAC addresses, source and destination IP addresses, IP protocol type number, incoming and outgoing TCP or UDP port numbers. To define a flow, all the available matching fields may be used. But it is also possible to restrict the matching rule to a subset of the available fields by using wildcards for the unwanted fields.")<br><br>Kempf at [0059] (In the EPC, a bearer is a transmission channel through an EPC packet network which has a defined set of data transmission characteristics ( quality of service data rate and flow control). EPC bearers are typically implemented at the network layer as DiffServ Code Points (DSCPs) or at the MAC layer as IEEE 802.lq VLANs with 802.lp (incorpo-rated into the 802.ld standard0 traffic class priorities,. The PCRF (Policy and Charging Resource Function) 801 identi-fies packet flows from the user equipment (UE) 807 that require bearers based on service requests from subsystems such as the IP multimedia subsystem (IMS). The packet flows to be included in a bearer are identified to the gateways and radio base station (E-NodeB) by 5 tuples, consisting of the IP source and destination address, the IP source and destination port, and the protocol identifier. The five tuples together with a DSCP for the QoS class identify an uplink and downlink packet filter. One bearer is set up per terminal IP address and QoS traffic class. The PCRF supplies a collection of four QoS parameters describing the bearer including: a quality class identifier (QCI) that specifies the QoS for the radio; allocation retention priority (ARP), which is an indicator of how the control plane should prioritize the bearer when requests for modification are made and resource conflicts arise; and a guaranteed bit rate (GBR) and maximum bit rate (MBR, optional) where these specify the guaranteed and maximum bit rates the bearer can receive. These are only defined for guaranteed-i.e. non-best effort-bearers.")<br><br>Kempf at [0061] ("In addition to the QoS parameters, each bearer has an associated GTP tunnel. A GTP tunnel consists of the IP address of the tunnel endpoint nodes (radio base station, S-GW 803, and P-GW 805), a source and destination UDP port, and a Tunnel Endpoint Identifier (TEID). GTP tunnels are unidirectional, so each bearer is associated with two TEIDs, one for the uplink and one for the downlink tunnel. One set of GTP tunnels (uplink and downlink) extends between the radio base station and the S-GW 803 and one set |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 125 of 1100

| No. | '111 Patent Claim 18 | Kempf |
|---|---|---|
| | | extends between the S-GW 803 and the P-GW 805. The UDP destination port number for GTP-U is 2152 while the desti-nation port number for GTP-C is 2123. The source port num-ber is dynamically allocated by the sending node. FIG. 10 is a diagram of one embodiment of the header fields in the primary GTP-U encapsulation header.")<br><br>Kempf at [0079] ("FIG. 16 is a diagram of one embodiment of a process for EPC peering and differential routing for specialized ser-vice treatment. The OpenFlow signaling, indicated by the solid lines and arrows 1601, sets up flow rules and actions on the switches and gateways within the EPC for differential routing. These flow rules direct GTP flows to particular loca-tions. In this example, the operator in this case peers its EPC with two other fixed operators. Routing through each peering point is handled by the respective P-GW-Dl and P-GW-D2 1603A, B. The dashed lines and arrows 1605 show traffic from a UE 1607 that needs to be routed to another peering operator. The flow rules and actions to distinguish which peering point the traffic should traverse are installed in the OpenFlow switches 1609 and gateways 1603A, B by the OpenFlow controller 1611. The OpenFlow controller 1611 calculates these flow rules and actions based on the routing tables it maintains for outside traffic, and the source and destination of the packets, as well as by any specialized for-warding treatment required for DSCP marked packets.")<br><br>Kempf at [0086] ("In one embodiment, an OpenFlow GTP gateway maintains a hash table mapping GTP TEIDs into the tunnel header fields for their bearers. FIG. 18 is a diagram of the structure of a flow table row. The TEID hash keys are calcu-lated using a suitable hash algorithm with low collision fre-quency, for example SHA-1. The gateway maintains one such flow table row for each GTP TEID/bearer. The TEID field contains the GTP TEID for the tunnel. The VLAN tags and MPLS labels fields contain an ordered list of VLAN tags and/or MPLS labels defining tunnels into which the packet needs to be routed. The VLAN priority bits and MPLS traffic class bits are included in the labels. Such tunnels may or may not be required. If they are not required, then these fields are empty. The tunnel origin source IP address contains the address on the encapsulating gateway to which any control traffic involving the tunnel should be directed (for example, error indications). The tunnel end destination IP address field contains the IP address of the gateway to which the tunneled packet should be routed, at which the packet will be decap-sulated and removed from the GTP tunnel. The QoS DSCP field contains the DiffServe Code Point, if any, for the bearer in the case of a dedicated bearer. This field may be empty if the bearer is a default bearer. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 126 of 1100

| No. | '111 Patent Claim 18 | Kempf |
|---|---|---|
| | | with best effort QoS, but will contain nonzero values if the bearer QoS is more than best effort.")<br><br>Kempf at [0089] ("In one embodiment, the system implements a GTP fast path encapsulation virtual port. When requested by the S-GW-C and P-GW-C control plane software running in the cloud computing system, the OpenFlow controller programs the gateway switch to install rules, actions, and TEID hash table entries for routing packets into GTP tunnels via a fast path GTP encapsulation virtual port. The rules match the packet filter for the input side of GTP tunnel's bearer. Typi-cally this will be a 4 tuple of: IP source address; IP destination address; UDP/TCP/SCTP source port; and UDP/TCP/SCTP destination port. The IP source address and destination address are typically the addresses for user data plane traffic, i.e. a UE or Internet service with which a UE is transacting, and similarly with the port numbers. For a rule matching the GTP-U tunnel input side, the associated instructions and are the following:<br><br>Write-Metadata ( GTP-TEID, OxFFFFFFFF)<br>Apply-Actions (Set-Output-Port GTP-Encap-VP")<br><br>Kempf at [0101] ("In one embodiment, the system implements han-dling of user data plane packets requiring GTP-U encapsula-tion with extension headers, sequence numbers, and N-PDU numbers. User data plane packets that require extension head-ers, sequence numbers, or N-PDU numbers during GTP encapsulation require special handling by the software slow path. For these packets, the OpenFlow controller programs a rule matching the 4 tuple: IP source address; IP destination address; UDP/TCP/SCTP source port; and UDP/TCP/SCTP destination port. The instructions for matching packets are:<br><br>Write-Metadata ( GTP-TEID, 0x FFFFFFFF)<br>Apply-Actions (Set-Output-Port LOCAL_GTP _U_ENCAP )") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 127 of 1100

| No. | '111 Patent Claim 18 | Kempf |
|---|---|---|
| 18[c] | wherein the packet-applicable criterion is that the first entity address, the second entity address, or both match a predetermined address or addresses. | Kempf discloses wherein the packet-applicable criterion is that the first entity address, the second entity address, or both match a predetermined address or addresses.<br><br>For example, Kempf discloses the packet header field used in the flow table matching is a sources and/or destination address of the electronic devices.<br><br>Kempf at [0044] ("FIG. 1 is a diagram of one embodiment of an example network with an OpenFlow switch, conforming to the OpenFlow 1.0 specification. The OpenFlow 1.0 protocol enables a controller 101 to connect to an OpenFlow 1.0 enabled switch 109 using a secure channel 103 and control a single forwarding table 107 in the switch 109. The controller 101 is an external software component executed by a remote computing device that enables a user to configure the Open-Flow 1.0 switch 109. The secure channel 103 can be provided by any type of network including a local area network (LAN) or a wide area network (WAN), such as the Internet.")<br><br>Kempf at [0045] ("FIG. 2 is a diagram illustrating one embodiment of the contents of a flow table entry. The forwarding table 107 is populated with entries consisting of a rule 201 defining matches for fields in packet headers; an action 203 associated to the flow match; and a collection of statistics 205 on the flow. When an incoming packet is received a lookup for a matching rule is made in the flow table 107. If the incoming packet matches a particular rule, the associated action defined in that flow table entry is performed on the packet.")<br><br>Kempf at [0046] ("A rule 201 contains key fields from several headers in the protocol stack, for example source and destination Ethernet MAC addresses, source and destination IP addresses, IP protocol type number, incoming and outgoing TCP or UDP port numbers. To define a flow, all the available matching fields may be used. But it is also possible to restrict the matching rule to a subset of the available fields by using wildcards for the unwanted fields.")<br><br>Kempf at [0047] ("The actions that are defined by the specification of OpenFlow 1.0 are Drop, which drops the matching packets; Forward, which forwards the packet to one or all outgoing ports, the incoming physical port itself, the controller via the secure channel, or the local networking stack (if it exists). OpenFlow 1.0 protocol data units (PDU s) are defined |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 128 of 1100

| No. | '111 Patent Claim 18 | Kempf |
|---|---|---|
| | | with a set of structures specified using the C programming language. Some of the more commonly used messages are: report switch configuration message; modify state messages (in-cluding a modify flow entry message and port modification message); read state messages, where while the system is running, the datapath may be queried about its current state using this message; and send packet message, which is used when the controller wishes to send a packet out through the datapath.")<br><br>Kempf at [0050] ("FIG. 4 illustrates one embodiment of the processing of packets through an OpenFlow 1.1 switched packet pro-cessing pipeline. A received packet is compared against each of the flow tables 401. After each flow table match, the actions are accumulated into an action set. If processing requires matching against another flow table, the actions in the matched rule include an action directing processing to the next table in the pipeline. Absent the inclusion of an action in the set to execute all accumulated actions immediately, the actions are executed at the end 403 of the packet processing pipeline. An action allows the writing of data to a metadata register, which is carried along in the packet processing pipe-line like the packet header.")<br><br>Kempf at [0051] ("FIG. 5 is a flowchart of one embodiment of the OpenFlow 1.1 rule matching process. OpenFlow 1.1 contains support for packet tagging. OpenFlow 1.1 allows matching based on header fields and multi-protocol label switching (MPLS) labels. One virtual LAN (VLAN) label and one MPLS label can be matched per table. The rule matching process is initiated with the arrival of a packet to be processed (Block 501 ). Starting at the first table 0 a lookup is performed to determine a match with the received packet (Block 503). If there is no match in this table, then one of a set of default actions is taken (i.e., send packet to controller, drop the packet or continue to next table) (Block 509). If there is a match, then an update to the action set is made along with counters, packet or match set fields and meta data (Block 505). A check is made to determine the next table to process, which can be the next table sequentially or one specified by an action of a matching rule (Block 507). Once all of the tables have been processed, then the resulting action set is executed (Block 511). FIG. 6 is a diagram of the fields, which a matching process can utilize for identifying rules to apply to a packet.")<br><br>Kempf at [0087] ("In one embodiment, slow path support for GTP is implemented with an OpenFlow gateway switch. An Open-Flow mobile gateway switch also contains support on |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 129 of 1100

| No. | '111 Patent Claim 18 | Kempf |
|---|---|---|
| | | the software control plane for slow path packet processing. This path is taken by G-PDU (message type 255) packets with nonzero header fields or extension headers, and user data plane packets requiring encapsulation with such fields or addition of extension headers, and by G TP-U control packets. For this purpose, the switch supports three local ports in the software control plane: LOCAL_GTP _CONTROL-the switch fast path forwards GTP encapsulated packets directed to the gateway IP address that contain GTP-U control mes-sages and the local switch software control plane initiates local control plane actions depending on the GTP-U control message; LOCAL_GTP _U_DECAP-the switch fast path forwards G-PDU packets to this port that have nonzero header fields or extension headers (i.e. E!=0, S!=0, or PN!=0). These packets require specialized handling. The local switch software slow path processes the packets and performs the specialized handling; and LOCAL_GTP _U_ENCAP-the switch fast path forwards user data plane packets to this port that require encapsulation in a GTP tunnel with nonzero header fields or extension headers (i.e. E!=0, S!=0, or PN!=0). These packets require specialized handling. The local switch software slow path encapsulates the packets and performs the specialized handling. In addition to forwarding the packet, the switch fast path makes the OpenFlow metadata field avail-able to the slow path software.")<br><br>Kempf at [0089] ("In one embodiment, the system implements a GTP fast path encapsulation virtual port. When requested by the S-GW-C and P-GW-C control plane software running in the cloud computing system, the OpenFlow controller programs the gateway switch to install rules, actions, and TEID hash table entries for routing packets into GTP tunnels via a fast path GTP encapsulation virtual port. The rules match the packet filter for the input side of GTP tunnel's bearer. Typi-cally this will be a 4 tuple of: IP source address; IP destination address; UDP/TCP/SCTP source port; and UDP/TCP/SCTP destination port. The IP source address and destination address are typically the addresses for user data plane traffic, i.e. a UE or Internet service with which a UE is transacting, and similarly with the port numbers. For a rule matching the GTP-U tunnel input side, the associated instructions and are the following:<br><br>Write-Metadata ( GTP-TEID, OxFFFFFFFF)<br>Apply-Actions (Set-Output-Port GTP-Encap-VP") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 130 of 1100

| No. | '111 Patent Claim 18 | Kempf |
|---|---|---|
| | | Kempf at [0092] ("In one embodiment, the system implements a GTP fast path decapsulation virtual port. When requested by the S-GW and P-GW control plane software running in the cloud computing system, the gateway switch installs rules and actions for routing GTP encapsulated packets out of GTP tunnels. The rules match the GTP header flags and the GTP TEID for the packet, in the modified OpenFlow flow table shown in FIG. 17 as follows: the IP destination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); the UDP destination port is the GTP-U destination port (2152); and the header fields and message type field is wildcarded with the flag 0XFFF0 and the upper two bytes of the field match the G-PDU message type (255) while the lower two bytes match 0x30, i.e. the packet is a GTP packet not a GTP' packet and the version number is 1.")<br><br>Kempf at [0094] ("In one embodiment, the system implements han-dling of GTP-U control packets. The OpenFlow controller programs the gateway switch flow tables with 5 rules for each gateway switch IP address used for GTP traffic. These rules contain specified values for the following fields: the IP des-tination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); the UDP destination port is the GTP-U destination port (2152); the GTP header flags and message type field is wildcarded with 0xFFF0; the value of the header flags field is 0x30, i.e. the version number is 1 and the PT field is 1; and the value of the message type field is one of 1 (Echo Request), 2 (Echo Response), 26 (Error Indication), 31 (Support for Extension Headers Notification), or 254 (End Marker).")<br><br>Kempf at [0097] ("In one embodiment, the system implements han-dling of G-PDU packets with extension headers, sequence numbers, and N-PDU numbers. G-PDU packets with exten-sion headers, sequence numbers, and N-PDU numbers need to be forwarded to the local switch software control plane for processing. The OpenFlow controller programs 3 rules for this purpose. They have the following common header fields: the IP destination address is an IP address on which the gateway is expecting GTP traffic; and the IP protocol type is UDP (17); the UDP destination port is the GTP-U destination port (2152).")<br><br>Kempf at [0101] ("In one embodiment, the system implements han-dling of user data plane packets requiring GTP-U encapsula-tion with extension headers, sequence numbers, and N-PDU numbers. User data plane packets that require extension head-ers, sequence numbers, |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 131 of 1100

| No. | '111 Patent Claim 18 | Kempf |
|---|---|---|
| | | or N-PDU numbers during GTP encapsulation require special handling by the software slow path. For these packets, the OpenFlow controller programs a rule matching the 4 tuple: IP source address; IP destination address; UDP/TCP/SCTP source port; and UDP/TCP/SCTP destination port. The instructions for matching packets are:<br><br>Write-Metadata ( GTP-TEID, 0x FFFFFFFF)<br>Apply-Actions (Set-Output-Port LOCAL_GTP _U_ENCAP )")<br><br>Kempf at [0104] ("In one embodiment, the system implements han-dling of GTP-C and GTP' control packets. Any GTP-C and GTP' control packets that are directed to IP addresses on a gateway switch are in error. These packets need to be handled by the S-GW-C, P-GW-C, and GTP' protocol entities in the cloud computing system, not the S-GW-D and P-GW-D enti-ties in the switches. To catch such packets, the OpenFlow controller must program the switch with the following two rules: the IP destination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); for one rule, the UDP destination port is the GTP-U destination port (2152), for the other, the UDP destination port is the GTP-C destination port (2123); the GTP header flags and message type fields are wildcarded.") |

| No. | '111 Patent Claim 19 | Kempf |
|---|---|---|
| 19 | The method according to claim 18, wherein the addresses are Internet Protocol (IP) addresses. | Kempf discloses the method according to claim 18, wherein the addresses are Internet Protocol (IP) addresses.<br><br>For example, Kempf discloses packets with header fields comprised of Internet Protocol source and destination addresses.<br><br>*See supra* at Claim 18.<br><br>Kempf at [0046] ("A rule 201 contains key fields from several headers in the protocol stack, for example source and destination Ethernet MAC addresses, source and destination IP addresses, IP protocol type number, incoming and outgoing TCP or UDP port numbers. To define a flow, all the available matching fields may be used. But it is also possible to restrict |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 132 of 1100

| No. | '111 Patent Claim 19 | Kempf |
|---|---|---|
| | | the matching rule to a subset of the available fields by using wildcards for the unwanted fields.")<br><br>Kempf at [0052] ("Actions allow manipulating of tag stacks by pushing and popping labels. Combined with multiple tables, VLAN or MPLS label stacks can be processed by matching one label per table. FIG. 7 is a flow chart of one embodiment of a header parsing process. The parsing process matches a packet header by initializing a set of match fields (Block 701) and checking for the presence of a set of different header types. The process checks for a VLAN tag (Block 703). If the VLAN tag is present, then there are a series of processing steps for the VLAN tag (Blocks 705-707). If the switch supports MPLS (Block 709), then there are a series of steps for detecting and processing the MPLS header information (Blocks 711-715). If the switch supports address resolution protocol (ARP), then there are a series of steps for processing the ARP header (Blocks 719 and 721). If the packet has an IP header (Block 723), then there are a series of steps for processing the IP header (Blocks 725-733). This process is performed for each received packet.)<br><br>Kempf at [0059] ("In the EPC, a bearer is a transmission channel through an EPC packet network which has a defined set of data transmission characteristics ( quality of service data rate and flow control). EPC bearers are typically implemented at the network layer as DiffServ Code Points (DSCPs) or at the MAC layer as IEEE 802.lq VLANs with 802.lp (incorpo-rated into the 802.ld standard0 traffic class priorities,. The PCRF (Policy and Charging Resource Function) 801 identi-fies packet flows from the user equipment (UE) 807 that require bearers based on service requests from subsystems such as the IP multimedia subsystem (IMS). The packet flows to be included in a bearer are identified to the gateways and radio base station (E-NodeB) by 5 tuples, consisting of the IP source and destination address, the IP source and destination port, and the protocol identifier. The five tuples together with a DSCP for the QoS class identify an uplink and downlink packet filter. One bearer is set up per terminal IP address and QoS traffic class. The PCRF supplies a collection of four QoS parameters describing the bearer including: a quality class identifier (QCI) that specifies the QoS for the radio; allocation retention priority (ARP), which is an indicator of how the control plane should prioritize the bearer when requests for modification are made and resource conflicts arise; and a guaranteed bit rate (GBR) and maximum bit rate (MBR, optional) where these specify the guaranteed and maximum bit |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 133 of 1100

| No. | '111 Patent Claim 19 | Kempf |
|---|---|---|
| | | rates the bearer can receive. These are only defined for guaranteed-i.e. non-best effort-bearers")<br><br>Kempf at [0061] ("In addition to the QoS parameters, each bearer has an associated GTP tunnel. A GTP tunnel consists of the IP address of the tunnel endpoint nodes (radio base station, S-GW 803, and P-GW 805), a source and destination UDP port, and a Tunnel Endpoint Identifier (TEID). GTP tunnels are unidirectional, so each bearer is associated with two TEIDs, one for the uplink and one for the downlink tunnel. One set of GTP tunnels (uplink and downlink) extends between the radio base station and the S-GW 803 and one set extends between the S-GW 803 and the P-GW 805. The UDP destination port number for GTP-U is 2152 while the desti-nation port number for GTP-C is 2123. The source port num-ber is dynamically allocated by the sending node. FIG. 10 is a diagram of one embodiment of the header fields in the primary GTP-U encapsulation header.")<br><br>Kempf at [0083] ("If a packet either needs encapsulation or arrives encapsulated with nonzero header flags, header extensions, and/or the GTP-U packet is not a G-PDU packet (i.e. it is a GTP-U control packet), the processing must proceed via the gateway's slow path (software) control plane. GTP-C and GTP' packets directed to the gateway's IP address are a result of mis-configuration and are in error. They must be sent to the OpenFlow controller, since these packets are handled by the S-GW-C and P-GW-C control plane entities in the cloud computing system or to the billing entity handling GTP' and not the S-GW-D and P-GW-D data plane switches.")<br><br>Kempf at [0086] ("In one embodiment, an OpenFlow GTP gateway maintains a hash table mapping GTP TEIDs into the tunnel header fields for their bearers. FIG. 18 is a diagram of the structure of a flow table row. The TEID hash keys are calcu-lated using a suitable hash algorithm with low collision fre-quency, for example SHA-1. The gateway maintains one such flow table row for each GTP TEID/bearer. The TEID field contains the GTP TEID for the tunnel. The VLAN tags and MPLS labels fields contain an ordered list of VLAN tags and/or MPLS labels defining tunnels into which the packet needs to be routed. The VLAN priority bits and MPLS traffic class bits are included in the labels. Such tunnels may or may not be required. If they are not required, then these fields are empty. The tunnel origin source IP address contains the address on the encapsulating gateway to which any control traffic involving the tunnel should be directed (for example, error indications). The tunnel |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 134 of 1100

| No. | '111 Patent Claim 19 | Kempf |
|---|---|---|
| | | end destination IP address field contains the IP address of the gateway to which the tunneled packet should be routed, at which the packet will be decap-sulated and removed from the GTP tunnel. The QoS DSCP field contains the DiffServe Code Point, if any, for the bearer in the case of a dedicated bearer. This field may be empty if the bearer is a default bearer with best effort QoS, but will contain nonzero values if the bearer QoS is more than best effort.")<br><br>Kempf at [0087] ("In one embodiment, slow path support for GTP is implemented with an OpenFlow gateway switch. An Open-Flow mobile gateway switch also contains support on the software control plane for slow path packet processing. This path is taken by G-PDU (message type 255) packets with nonzero header fields or extension headers, and user data plane packets requiring encapsulation with such fields or addition of extension headers, and by G TP-U control packets. For this purpose, the switch supports three local ports in the software control plane: LOCAL_GTP _CONTROL-the switch fast path forwards GTP encapsulated packets directed to the gateway IP address that contain GTP-U control mes-sages and the local switch software control plane initiates local control plane actions depending on the GTP-U control message; LOCAL_GTP _U_DECAP-the switch fast path forwards G-PDU packets to this port that have nonzero header fields or extension headers (i.e. E!=0, S!=0, or PN!=0). These packets require specialized handling. The local switch software slow path processes the packets and performs the specialized handling; and LOCAL_GTP _U_ENCAP-the switch fast path forwards user data plane packets to this port that require encapsulation in a GTP tunnel with nonzero header fields or extension headers (i.e. E!=0, S!=0, or PN!=0). These packets require specialized handling. The local switch software slow path encapsulates the packets and performs the specialized handling. In addition to forwarding the packet, the switch fast path makes the OpenFlow metadata field avail-able to the slow path software.")<br><br>Kempf at [0089] ("In one embodiment, the system implements a GTP fast path encapsulation virtual port. When requested by the S-GW-C and P-GW-C control plane software running in the cloud computing system, the OpenFlow controller programs the gateway switch to install rules, actions, and TEID hash table entries for routing packets into GTP tunnels via a fast path GTP encapsulation virtual port. The rules match the packet filter for the input side of GTP tunnel's bearer. Typi-cally this will be a 4 tuple of: IP source address; IP destination address; UDP/TCP/SCTP source port; and UDP/TCP/SCTP |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 135 of 1100

| No. | '111 Patent Claim 19 | Kempf |
|---|---|---|
| | | destination port. The IP source address and destination address are typically the addresses for user data plane traffic, i.e. a UE or Internet service with which a UE is transacting, and similarly with the port numbers. For a rule matching the GTP-U tunnel input side, the associated instructions and are the following:<br><br>Write-Metadata ( GTP-TEID, OxFFFFFFFF)<br>Apply-Actions (Set-Output-Port GTP-Encap-VP")<br><br>Kempf at [0092] ("In one embodiment, the system implements a GTP fast path decapsulation virtual port. When requested by the S-GW and P-GW control plane software running in the cloud computing system, the gateway switch installs rules and actions for routing GTP encapsulated packets out of GTP tunnels. The rules match the GTP header flags and the GTP TEID for the packet, in the modified OpenFlow flow table shown in FIG. 17 as follows: the IP destination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); the UDP destination port is the GTP-U destination port (2152); and the header fields and message type field is wildcarded with the flag 0XFFF0 and the upper two bytes of the field match the G-PDU message type (255) while the lower two bytes match 0x30, i.e. the packet is a GTP packet not a GTP' packet and the version number is 1.")<br><br>Kempf at [0094] ("In one embodiment, the system implements han-dling of GTP-U control packets. The OpenFlow controller programs the gateway switch flow tables with 5 rules for each gateway switch IP address used for GTP traffic. These rules contain specified values for the following fields: the IP des-tination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); the UDP destination port is the GTP-U destination port (2152); the GTP header flags and message type field is wildcarded with 0xFFF0; the value of the header flags field is 0x30, i.e. the version number is 1 and the PT field is 1; and the value of the message type field is one of 1 (Echo Request), 2 (Echo Response), 26 (Error Indication), 31 (Support for Extension Headers Notification), or 254 (End Marker).")<br><br>Kempf at [0097] ("In one embodiment, the system implements han-dling of G-PDU packets with extension headers, sequence numbers, and N-PDU numbers. G-PDU packets with exten-sion headers, sequence numbers, and N-PDU numbers need to be forwarded to the |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 136 of 1100

| No. | '111 Patent Claim 19 | Kempf |
|---|---|---|
| | | local switch software control plane for processing. The OpenFlow controller programs 3 rules for this purpose. They have the following common header fields: the IP destination address is an IP address on which the gateway is expecting GTP traffic; and the IP protocol type is UDP (17); the UDP destination port is the GTP-U destination port (2152).")<br><br>Kempf at [0101] ("In one embodiment, the system implements han-dling of user data plane packets requiring GTP-U encapsula-tion with extension headers, sequence numbers, and N-PDU numbers. User data plane packets that require extension head-ers, sequence numbers, or N-PDU numbers during GTP encapsulation require special handling by the software slow path. For these packets, the OpenFlow controller programs a rule matching the 4 tuple: IP source address; IP destination address; UDP/TCP/SCTP source port; and UDP/TCP/SCTP destination port. The instructions for matching packets are:<br><br>Write-Metadata ( GTP-TEID, 0x FFFFFFFF)<br>Apply-Actions (Set-Output-Port LOCAL_GTP _U_ENCAP )")<br><br>Kempf at [0104] ("In one embodiment, the system implements han-dling of GTP-C and GTP' control packets. Any GTP-C and GTP' control packets that are directed to IP addresses on a gateway switch are in error. These packets need to be handled by the S-GW-C, P-GW-C, and GTP' protocol entities in the cloud computing system, not the S-GW-D and P-GW-D enti-ties in the switches. To catch such packets, the OpenFlow controller must program the switch with the following two rules: the IP destination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); for one rule, the UDP destination port is the GTP-U destination port (2152), for the other, the UDP destination port is the GTP-C destination port (2123); the GTP header flags and message type fields are wildcarded.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 137 of 1100

| No. | '111 Patent Claim 20 | Kempf |
|---|---|---|
| 20[a] | The method according to claim 1, wherein the packet is an Transmission Control Protocol (TCP) packet that comprises source and destination TCP ports, a TCP sequence number, and a TCP sequence mask fields, and | Kempf discloses the method according to claim 1, wherein the packet is an Transmission Control Protocol (TCP) packet that comprises source and destination TCP ports, a TCP sequence number, and a TCP sequence mask fields.<br><br>For example, Kempf discloses packets that belong to the Transmission Control Protocol that include source and destination ports, sequence numbers, and mask fields.<br><br>*See supra* at Claim 1, 17[a].<br><br>Kempf at [0082] ("In one embodiment, OpenFlow can be modified to support virtual ports for fast path GTP TEID encapsulation and decapsulation. An OpenFlow mobile gateway can be used to support GTP encapsulation and decapsulation with virtual ports. The GTP encapsulation and decapsulation virtual ports can be used for fast encapsulation and decapsulation of user data packets within GTP-U tunnels, and can be designed simply enough that they can be implemented in hardware or firmware. For this reason, GTP virtual ports may have the following restrictions on traffic they will handle: Protocol Type (PT) field= 1, where GTP encapsulation ports only sup-port GTP, not GTP' (PT field=0); Extension Header flag (E)=0, where no extension headers are supported, Sequence Number flag (S)=0, where no sequence numbers are sup-ported; N-PDU flag (PN)=0; and Message type=255, where Only G-PDU messages, i.e. tunneled user data, is supported in the fast path.")<br><br>Kempf at [0088] ("To support slow path encapsulation, the software control plane on the switch maintains a hash table with keys calculated from the GTP-U TEID. The TEID hash keys are calculated using a suitable hash algorithm with low collision frequency, for example SHA-1. The flow table entries contain a record of how the packet header, including the GTP encap-sulation header, should be configured. This includes: the same header fields as for the hardware or firmware encapsu-lation table in FIG.18; values for the GTP header flags (PT, E, S, and PN); the sequence number and/or the N-PDU number if any; if the E flag is 1, then the flow table contains a list of the extension headers, including their types, which the slow path should insert into the GTP header.")<br><br>Kempf at [0097] ("In one embodiment, the system implements han-dling of G-PDU packets with extension headers, sequence numbers, and N-PDU numbers. G-PDU packets with |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 138 of 1100

| No. | '111 Patent Claim 20 | Kempf |
|---|---|---|
| | | exten-sion headers, sequence numbers, and N-PDU numbers need to be forwarded to the local switch software control plane for processing. The OpenFlow controller programs 3 rules for this purpose. They have the following common header fields: the IP destination address is an IP address on which the gateway is expecting GTP traffic; and the IP protocol type is UDP (17); the UDP destination port is the GTP-U destination port (2152).")<br><br>Kempf at [0098] ("The header flags and message type fields for the three rules are wildcarded with the following bitmasks and match as follows: bitmask 0xFFF4 and the upper two bytes match the G-PDU message type (255) while the lower two bytes are Ox34, indicating that the version number is 1, the packet is a GTP packet, and there is an extension header present; bitmask 0xFFFF2 and the upper two bytes match the G-PDU message type (255) while the lower two bytes are 0x32, indicating that the version number is 1, the packet is a GTP packet, and there is a sequence number present; and bitmask 0xFF0l and the upper two bytes match the G-PDU message type (255) while the lower two bytes are 0x31, indicating that the version number is 1, the packet is a GTP packet, and a N-PDU is present.")<br><br>Kempf at [0101] ("In one embodiment, the system implements han-dling of user data plane packets requiring GTP-U encapsula-tion with extension headers, sequence numbers, and N-PDU numbers. User data plane packets that require extension head-ers, sequence numbers, or N-PDU numbers during GTP encapsulation require special handling by the software slow path. For these packets, the OpenFlow controller programs a rule matching the 4 tuple: IP source address; IP destination address; UDP/TCP/SCTP source port; and UDP/TCP/SCTP destination port. The instructions for matching packets are:<br><br>Write-Metadata ( GTP-TEID, 0x FFFFFFFF)<br>Apply-Actions (Set-Output-Port LOCAL_GTP _U_ENCAP )")<br><br>Kempf at [0114] ("The gtp_type_n_flags field contains the GTP mes-sage type in the upper 8 bits and the GTP header flags in the lower 8 bits. The gtp_teid field contains the GTP TEID. The gtp_ wildcard field indicates whether the GTP type and flags and TEID should be matched. If the lower four bits are 1, the type and flags field should be ignored, while if the upper four bits are 1, the TEID should be ignored. If the lower bits are 0, the type and |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 139 of 1100

| No. | '111 Patent Claim 20 | Kempf |
|---|---|---|
| | | fields flag should be matched subject to the flags in the gtp_flag_mask field, while if the upper bits are 0 the TEID should be matched. The mask is combined with the message type and header field of the packet using logical AND; the result becomes the value of the match. Only those parts of the field in which the mask has a 1 value are matched.") |
| 20[b] | wherein the packet-applicable criterion is that the source TCP port, the destination TCP port, the TCP sequence number, the TCP sequence mask, or any combination thereof, matches a predetermined value or values. | Kempf discloses wherein the packet-applicable criterion is that the source TCP port, the destination TCP port, the TCP sequence number, the TCP sequence mask, or any combination thereof, matches a predetermined value or values.<br><br>For example, Kempf discloses header fields including TCP include source and destination ports, sequence numbers, and mask fields as matching fields in the flow table.<br><br>Kempf at [0046] ("A rule 201 contains key fields from several headers in the protocol stack, for example source and destination Ethernet MAC addresses, source and destination IP addresses, IP protocol type number, incoming and outgoing TCP or UDP port numbers. To define a flow, all the available matching fields may be used. But it is also possible to restrict the matching rule to a subset of the available fields by using wildcards for the unwanted fields.")<br><br>Kempf at [0081] ("In one embodiment, OpenFlow is modified to pro-vide rules for GTP TEID Routing. FIG. 17 is a diagram of one embodiment of the OpenFlow flow table modification for GTP TEID routing. An OpenFlow switch that supports TEID routing matches on the 2 byte (16 bit) collection of header fields and the 4 byte (32 bit) GTP TEID, in addition to other OpenFlow header fields, in at least one flow table ( e.g., the first flow table). The GTP TEID flag can be wildcarded (i.e. matches are "don't care"). In one embodiment, the EPC pro-tocols do not assign any meaning to TEIDs other than as an endpoint identifier for tunnels, like ports in standard UDP/ TCP transport protocols. In other embodiments, the TEIDs can have a correlated meaning or semantics. The GTP header flags field can also be wildcarded, this can be partially matched by combining the following bitmasks: 0xFF00- Match the Message Type field; 0xe0-Match the Version field; 0xl0-Match the PT field; 0x04-Match the E field; 0x02- Match the S field; and 0x0l-Match the PN field.") |


~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 140 of 1100

| No. | '111 Patent Claim 20 | Kempf |
|---|---|---|
| | | Kempf at [0082] ("In one embodiment, OpenFlow can be modified to support virtual ports for fast path GTP TEID encapsulation and decapsulation. An OpenFlow mobile gateway can be used to support GTP encapsulation and decapsulation with virtual ports. The GTP encapsulation and decapsulation virtual ports can be used for fast encapsulation and decapsulation of user data packets within GTP-U tunnels, and can be designed simply enough that they can be implemented in hardware or firmware. For this reason, GTP virtual ports may have the following restrictions on traffic they will handle: Protocol Type (PT) field= 1, where GTP encapsulation ports only sup-port GTP, not GTP' (PT field=0); Extension Header flag (E)=0, where no extension headers are supported, Sequence Number flag (S)=0, where no sequence numbers are sup-ported; N-PDU flag (PN)=0; and Message type=255, where Only G-PDU messages, i.e. tunneled user data, is supported in the fast path.")<br><br>Kempf at [0088] ("To support slow path encapsulation, the software control plane on the switch maintains a hash table with keys calculated from the GTP-U TEID. The TEID hash keys are calculated using a suitable hash algorithm with low collision frequency, for example SHA-1. The flow table entries contain a record of how the packet header, including the GTP encap-sulation header, should be configured. This includes: the same header fields as for the hardware or firmware encapsu-lation table in FIG.18; values for the GTP header flags (PT, E, S, and PN); the sequence number and/or the N-PDU number if any; if the E flag is 1, then the flow table contains a list of the extension headers, including their types, which the slow path should insert into the GTP header.")<br><br>Kempf at [0089] ("In one embodiment, the system implements a GTP fast path encapsulation virtual port. When requested by the S-GW-C and P-GW-C control plane software running in the cloud computing system, the OpenFlow controller programs the gateway switch to install rules, actions, and TEID hash table entries for routing packets into GTP tunnels via a fast path GTP encapsulation virtual port. The rules match the packet filter for the input side of GTP tunnel's bearer. Typi-cally this will be a 4 tuple of: IP source address; IP destination address; UDP/TCP/SCTP source port; and UDP/TCP/SCTP destination port. The IP source address and destination address are typically the addresses for user data plane traffic, i.e. a UE or Internet service with which a UE is transacting, and similarly with the port numbers. For a rule matching the GTP-U tunnel input side, the associated instructions and are the following: |


~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 141 of 1100

| No. | '111 Patent Claim 20 | Kempf |
|---|---|---|
| | | Write-Metadata ( GTP-TEID, OxFFFFFFFF)<br>Apply-Actions (Set-Output-Port GTP-Encap-VP")<br><br>Kempf at [0101] ("In one embodiment, the system implements han-dling of user data plane packets requiring GTP-U encapsula-tion with extension headers, sequence numbers, and N-PDU numbers. User data plane packets that require extension head-ers, sequence numbers, or N-PDU numbers during GTP encapsulation require special handling by the software slow path. For these packets, the OpenFlow controller programs a rule matching the 4 tuple: IP source address; IP destination address; UDP/TCP/SCTP source port; and UDP/TCP/SCTP destination port. The instructions for matching packets are:<br><br>Write-Metadata ( GTP-TEID, 0x FFFFFFFF)<br>Apply-Actions (Set-Output-Port LOCAL_GTP _U_ENCAP )")<br><br>Kempf at [0097] ("In one embodiment, the system implements han-dling of G-PDU packets with extension headers, sequence numbers, and N-PDU numbers. G-PDU packets with exten-sion headers, sequence numbers, and N-PDU numbers need to be forwarded to the local switch software control plane for processing. The OpenFlow controller programs 3 rules for this purpose. They have the following common header fields: the IP destination address is an IP address on which the gateway is expecting GTP traffic; and the IP protocol type is UDP (17); the UDP destination port is the GTP-U destination port (2152).")<br><br>Kempf at [0098] ("The header flags and message type fields for the three rules are wildcarded with the following bitmasks and match as follows: bitmask 0xFFF4 and the upper two bytes match the G-PDU message type (255) while the lower two bytes are Ox34, indicating that the version number is 1, the packet is a GTP packet, and there is an extension header present; bitmask 0xFFFF2 and the upper two bytes match the G-PDU message type (255) while the lower two bytes are 0x32, indicating that the version number is 1, the packet is a GTP packet, and there is a sequence number present; and bitmask 0xFF0l and the upper two bytes match the G-PDU message type (255) while the lower two bytes are 0x31, indicating that the version number is 1, the packet is a GTP packet, and a N-PDU is present.") |


~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 142 of 1100

| No. | '111 Patent Claim 21 | Kempf |
|---|---|---|
| 21 | The method according to claim 1, wherein the packet network comprises a Wide Area Network (WAN), Local Area Network (LAN), the Internet, Metropolitan Area Network (MAN), Internet Service Provider (ISP) backbone datacenter network, or inter - datacenter network. | Kempf discloses the method according to claim 1, wherein the packet network comprises a Wide Area Network (WAN), Local Area Network (LAN), the Internet, Metropolitan Area Network (MAN), Internet Service Provider (ISP) backbone datacenter network, or inter - datacenter network.<br><br>For example, Kempf discloses a packet network including a local area network, wide area network, and the Internet.<br><br>*See supra* at Claim 1.<br><br>Kempf at [0044] ("FIG. 1 is a diagram of one embodiment of an example network with an OpenFlow switch, conforming to the OpenFlow 1.0 specification. The OpenFlow 1.0 protocol enables a controller 101 to connect to an OpenFlow 1.0 enabled switch 109 using a secure channel 103 and control a single forwarding table 107 in the switch 109. The controller 101 is an external software component executed by a remote computing device that enables a user to configure the Open-Flow 1.0 switch 109. The secure channel 103 can be provided by any type of network including a local area network (LAN) or a wide area network (WAN), such as the Internet.") |

| No. | '111 Patent Claim 22 | Kempf |
|---|---|---|
| 22 | The method according to claim 1, wherein the first entity is a server device and the second entity is a client device, or wherein the first entity is a client device and the second entity is a server device. | Kempf discloses the method according to claim 1, wherein the first entity is a server device and the second entity is a client device, or wherein the first entity is a client device and the second entity is a server device.<br><br>For example, Kempf discloses electronic devices including subscriber end stations such as servers, laptops, smart phones, mobile phones, etc. that communicate packets between each other.<br><br>*See supra* at Claim 1. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 143 of 1100

| No. | '111 Patent Claim 22 | Kempf |
|---|---|---|
| | | Kempf at [0033] ("As used herein, a network element (e.g., a router, switch, bridge, etc.) is a piece of networking equipment, including hardware and software, that communicatively interconnects other equipment on the network (e.g., other network elements, end stations, etc.). Some network elements are "multiple services network elements" that provide sup-port for multiple networking functions (e.g., routing, bridg-ing, switching, Layer 2 aggregation, session border control, multicasting, and/or subscriber management), and/or provide support for multiple application services (e.g., data, voice, and video). Subscriber end stations ( e.g., servers, worksta-tions, laptops, palm tops, mobile phones, smart phones, mul-timedia phones, Voice Over Internet Protocol (VOIP) phones, portable media players, GPS units, gaming systems, set-top boxes (STBs), etc.) access content/services provided over the Internet and/or content/services provided on virtual private networks (VPN s) overlaid on the Internet. The content and/or services are typically provided by one or more end stations ( e.g., server end stations) belonging to a service or content provider or end stations participating in a peer to peer service, and may include public web pages (free content, store fronts, search services, etc.), private web pages ( e.g., username/pass-word accessed web pages providing email services, etc.), corporate networks over VPNs, IPTV, etc. Typically, sub-scriber end stations are coupled ( e.g., through customer premise equipment coupled to an access network (wired or wirelessly)) to edge network elements, which are coupled (e.g., through one or more core network elements to other edge network elements) to other end stations (e.g., server end stations).<br><br>Kempf at [0034] ("The embodiments of the present invention provide a method and system for avoiding the disadvantages of the prior art. The disadvantages of the prior art are that prior imple-mentations of the evolved packet core use a pool of servers that are dedicated to a specific network entity, such as a server pool that is dedicated to hosting a mobility management entity (MME). When additional signaling demands require that extra capacity, then a new MME instance is instantiated in the server pool. However, when demand is high for the services of a policy and charging rules function (PCRF) and low for MMEs, the server pool dedicated to the PCRF servers will be heavily utilized, but the server pool for the MMEs is underutilized. These underutilized server pools continue to require maintenance and incur operating expenses, but are not providing optimum performance for the network operator.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 144 of 1100

| No. | '111 Patent Claim 22 | Kempf |
|---|---|---|
| | | Kempf at [0035] ("In some situations, managed services companies build and run mobile operator networks, while the mobile operator itself handles marketing, billing, and customer rela-tions. The signaling and data traffic for each mobile operator network is kept private and isolated from the traffic of their competitors, even though their network and their competi-tors' networks may be managed by the same managed ser-vices company. The managed services company must main-tain a completely separate server pool and physical signaling network for each mobile operator it supports. As a result, there is a large duplication of resources and an underutiliza-tion of server capacity. This increases operating expenses for the managed services companies and the mobile operator network due to the additional equipment, power and cooling requirements.")<br><br>Kempf at [0065] ("A cloud computing system can be composed of any number of computing devices having any range of capabili-ties (e.g., processing power or storage capacity). The cloud computing system can be a private or public system. The computing devices can be in communication with one another across any communication system or network. A cloud com-puting system can support a single cloud or service or any number of discrete clouds or services. Services, applications and similar programs can be virtualized or executed as stan-dard code. In one embodiment, cloud computing systems can support web services applications. Web services applications consist of a load balancing front end that dispatches requests to a pool of Web servers. The requests originate from appli-cations on remote machines on the Internet and therefore the security and privacy requirements are much looser than for applications in a private corporate network.")<br><br>Kempf at [0143] ("In other embodiments, the split EPC architecture can be implemented in non-cloud and non-virtualized sys-tems. The control plane entities of the EPC architecture can be stored and executed on a single server or distributed across any number of servers or similar computing devices. Simi-larly, the control plane entities can be executed as standard software code and modules without virtualization or similar systems. These control plane entities can communicate with one another through local system or procedure calls, remote procedure calls or similar mechanisms. In further embodi-ments, a subset of the control plane entities can be virtualized or executed in a cloud computing system while another subset of the control plane entities can be executed in a server, distributed server system or similar system. The control plane entities can communicate with the data plane through the |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 145 of 1100

| No. | '111 Patent Claim 22 | Kempf |
|---|---|---|
| | | use of the OpenFlow protocol as described herein above or through other control protocols as described herein below.") |

| No. | '111 Patent Claim 23 | Kempf |
|---|---|---|
| 23[a] | The method according to claim 22, wherein the server device comprises a web server, and | Kempf discloses the method according to claim 22, wherein the server device comprises a web server.<br><br>For example, Kempf discloses servers that include web servers for web service applications.<br><br>*See supra* at Claim 22.<br><br>Kempf at [0033] ("As used herein, a network element (e.g., a router, switch, bridge, etc.) is a piece of networking equipment, including hardware and software, that communicatively interconnects other equipment on the network (e.g., other network elements, end stations, etc.). Some network elements are "multiple services network elements" that provide sup-port for multiple networking functions (e.g., routing, bridg-ing, switching, Layer 2 aggregation, session border control, multicasting, and/or subscriber management), and/or provide support for multiple application services (e.g., data, voice, and video). Subscriber end stations ( e.g., servers, worksta-tions, laptops, palm tops, mobile phones, smart phones, mul-timedia phones, Voice Over Internet Protocol (VOIP) phones, portable media players, GPS units, gaming systems, set-top boxes (STBs), etc.) access content/services provided over the Internet and/or content/services provided on virtual private networks (VPN s) overlaid on the Internet. The content and/or services are typically provided by one or more end stations ( e.g., server end stations) belonging to a service or content provider or end stations participating in a peer to peer service, and may include public web pages (free content, store fronts, search services, etc.), private web pages ( e.g., username/pass-word accessed web pages providing email services, etc.), corporate networks over VPNs, IPTV, etc. Typically, sub-scriber end stations are coupled ( e.g., through customer premise equipment coupled to an access network (wired or wirelessly)) to edge network elements, |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 146 of 1100

| No. | '111 Patent Claim 23 | Kempf |
|---|---|---|
| | | which are coupled (e.g., through one or more core network elements to other edge network elements) to other end stations (e.g., server end stations).")<br><br>Kempf at [0034] ("The embodiments of the present invention provide a method and system for avoiding the disadvantages of the prior art. The disadvantages of the prior art are that prior imple-mentations of the evolved packet core use a pool of servers that are dedicated to a specific network entity, such as a server pool that is dedicated to hosting a mobility management entity (MME). When additional signaling demands require that extra capacity, then a new MME instance is instantiated in the server pool. However, when demand is high for the services of a policy and charging rules function (PCRF) and low for MMEs, the server pool dedicated to the PCRF servers will be heavily utilized, but the server pool for the MMEs is underutilized. These underutilized server pools continue to require maintenance and incur operating expenses, but are not providing optimum performance for the network operator.")<br><br>Kempf at [0035] ("In some situations, managed services companies build and run mobile operator networks, while the mobile operator itself handles marketing, billing, and customer rela-tions. The signaling and data traffic for each mobile operator network is kept private and isolated from the traffic of their competitors, even though their network and their competi-tors' networks may be managed by the same managed ser-vices company. The managed services company must main-tain a completely separate server pool and physical signaling network for each mobile operator it supports. As a result, there is a large duplication ofresources and an underutiliza-tion of server capacity. This increases operating expenses for the managed services companies and the mobile operator network due to the additional equipment, power and cooling requirements.")<br><br>Kempf at [0065] ("A cloud computing system can be composed of any number of computing devices having any range of capabili-ties (e.g., processing power or storage capacity). The cloud computing system can be a private or public system. The computing devices can be in communication with one another across any communication system or network. A cloud com-puting system can support a single cloud or service or any number of discrete clouds or services. Services, applications and similar programs can be virtualized or executed as stan-dard code. In one embodiment, cloud computing systems can support web services applications. Web services applications consist of a load balancing front end that |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 147 of 1100

| No. | '111 Patent Claim 23 | Kempf |
|---|---|---|
| | | dispatches requests to a pool of Web servers. The requests originate from appli-cations on remote machines on the Internet and therefore the security and privacy requirements are much looser than for applications in a private corporate network.")<br><br>Kempf at [0143] ("In other embodiments, the split EPC architecture can be implemented in non-cloud and non-virtualized sys-tems. The control plane entities of the EPC architecture can be stored and executed on a single server or distributed across any number of servers or similar computing devices. Simi-larly, the control plane entities can be executed as standard software code and modules without virtualization or similar systems. These control plane entities can communicate with one another through local system or procedure calls, remote procedure calls or similar mechanisms. In further embodi-ments, a subset of the control plane entities can be virtualized or executed in a cloud computing system while another subset of the control plane entities can be executed in a server, distributed server system or similar system. The control plane entities can communicate with the data plane through the use of the OpenFlow protocol as described herein above or through other control protocols as described herein below.") |
| 23[b] | wherein the client device comprises a smartphone, a tablet computer, a personal computer, a laptop computer, or a wearable computing device. | Kempf discloses wherein the client device comprises a smartphone, a tablet computer, a personal computer, a laptop computer, or a wearable computing device.<br><br>For example, Kempf discloses subscriber end points including smartphones, mobile phones, laptops, etc.<br><br>Kempf at [0033] ("As used herein, a network element (e.g., a router, switch, bridge, etc.) is a piece of networking equipment, including hardware and software, that communicatively interconnects other equipment on the network (e.g., other network elements, end stations, etc.). Some network elements are "multiple services network elements" that provide sup-port for multiple networking functions (e.g., routing, bridg-ing, switching, Layer 2 aggregation, session border control, multicasting, and/or subscriber management), and/or provide support for multiple application services (e.g., data, voice, and video). Subscriber end stations ( e.g., servers, worksta-tions, laptops, palm tops, mobile phones, smart phones, mul-timedia phones, Voice Over Internet Protocol (VOIP) phones, portable media players, GPS units, gaming systems, set-top boxes (STBs), etc.) access content/services provided |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 148 of 1100

| No. | '111 Patent Claim 23 | Kempf |
|---|---|---|
| | | over the Internet and/or content/services provided on virtual private networks (VPN s) overlaid on the Internet. The content and/or services are typically provided by one or more end stations ( e.g., server end stations) belonging to a service or content provider or end stations participating in a peer to peer service, and may include public web pages (free content, store fronts, search services, etc.), private web pages ( e.g., username/pass-word accessed web pages providing email services, etc.), corporate networks over VPNs, IPTV, etc. Typically, sub-scriber end stations are coupled ( e.g., through customer premise equipment coupled to an access network (wired or wirelessly)) to edge network elements, which are coupled (e.g., through one or more core network elements to other edge network elements) to other end stations (e.g., server end stations).")<br><br>Kempf at [0065] ("A cloud computing system can be composed of any number of computing devices having any range of capabili-ties (e.g., processing power or storage capacity). The cloud computing system can be a private or public system. The computing devices can be in communication with one another across any communication system or network. A cloud com-puting system can support a single cloud or service or any number of discrete clouds or services. Services, applications and similar programs can be virtualized or executed as stan-dard code. In one embodiment, cloud computing systems can support web services applications. Web services applications consist of a load balancing front end that dispatches requests to a pool of Web servers. The requests originate from appli-cations on remote machines on the Internet and therefore the security and privacy requirements are much looser than for applications in a private corporate network.") |

| No. | '111 Patent Claim 24 | Kempf |
|---|---|---|
| 24 | The method according to claim 22, wherein the communication between the network node and the controller is based on, or uses, a standard protocol. | Kempf discloses the method according to claim 22, wherein the communication between the network node and the controller is based on, or uses, a standard protocol.<br><br>For example, Kempf discloses communication between network elements and the OpenFlow controller based on standard protocols including GPRS, GTP, OpenFlow, etc.<br><br>*See supra* at Claim 22.<br><br>Kempf at [0004] ("The GPRS tunneling protocol (GTP) is an important communication protocol utilized within the GPRS core net-work. GTP enables end user devices ( e.g., |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 149 of 1100

| No. | '111 Patent Claim 24 | Kempf |
|---|---|---|
| | | cellular phones) in a GSM network to move from place to place while continuing to connect to the Internet. The end user devices are connected to other devices through a gateway GPRS support node (GGSN). The GGSN tracks the end user device's data from the end user device's serving GPRS support node (GGSN) that is handling the session originating from the end user device.")<br><br>Kempf at [0006] ("A method implements a control plane of an evolved packet core (EPC) of a third generation partnership project (3GPP) long term evolution (LTE) network in a cloud com-puting system. The cloud computing system includes a cloud manager and a controller. The controller executes a plurality of control plane modules. The control plane communicates with the data plane of the EPC implemented in a plurality of network elements of the 3GPP LTE network through a control protocol. The EPC with the control plane implemented in the cloud computing system utilizes resources more efficiently than an architecture with the control plane implemented in the plurality of network elements of the 3GPP LTE network. The method comprises the steps of initializing the plurality of control plane modules of the EPC within the controller. Each control plane module in the plurality of control plane modules is initialized as a separate virtual machine by the cloud man-ager. Each control plane module provides a set of control plane functions for managing the data plane. The cloud man-ager monitors resource utilization of each control plane mod-ule and the control plane traffic handled by each control plane module. The cloud manager detects a threshold level of resource utilization or traffic load for one of the plurality of control plane modules of the EPC. A new control plane mod-ule is initialized as a separate virtual machine by the cloud manager in response to detecting the threshold level. The new control plane module shares the load of the one of the plural-ity of control plane modules and signals the plurality of net-work elements in the data plane to establish flow rules and actions to establish differential routing of flows in the data plane using the control protocol, wherein the control protocol is an OpenFlow protocol, and wherein flow matches are encoded using an extensible match structure in which the flow match is encoded as a type-length-value (TLV).")<br><br>Kempf at [0007] ("A cloud computer system implements a control plane of an evolved packet core (EPC) of a third generation partnership project (3GPP) long term evolution (LTE) net-work. The control plane communicates with the data plane of the EPC that is implemented in a plurality of network ele-ments of the 3GPP LTE network through a |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 150 of 1100

| No. | '111 Patent Claim 24 | Kempf |
|---|---|---|
| | | control protocol. The EPC with the control plane implemented in the cloud computing system utilizes resources more efficiently than an architecture with the control plane implemented in the plu-rality of network elements of the 3GPP LTE network. The cloud computing system, comprises a controller configured to execute a plurality of control plane modules of the EPC, each control plane module configured to provide a set of control plane functions for managing the data plane and to signal the plurality of network elements in the data plane to establish flow rules and actions to establish differential rout-ing of flows in the data plane using the control protocol, wherein the control protocol is an OpenFlow protocol, and wherein flow matches are encoded using an extensible match structure in which the flow match is encoded as a type-length-value (TLV) and a cloud manager communicatively coupled to the controller. The cloud manager is configured to initialize each of the plurality of control plane modules within the controller as a separate virtual machine, monitor resource utilization of each control plane module and the control plane traffic handled by each control plane module, detect whether a threshold level ofresource utilization or traffic load has been reached by any of the plurality of control plane modules of the EPC, and initialize a new control plane module as a separate virtual machine in response to detecting the threshold level, the new control plane module to share the load of the one of the plurality of control plane modules that exceeded the threshold level.")<br><br>Kempf at [0038] ("Implementing the control plane of an EPC in a cloud computing facility and the data plane of the EPC using a set of OpenFlow switches, as well as managing communication between the control plane and the dataplane using the Open-Flow protocol (e.g., OpenFlow 1.1), creates a problem that the OpenFlow protocol does not support GTP or GTP tunnel endpoint identifier (TEID) routing, which is necessary for implementing the dataplane of the EPC")<br><br>Kempf at [0039] ("The embodiments of the invention overcome these disadvantages of the prior art. The disadvantages of the prior art are avoided by splitting the control plane and the data plane for the EPC architecture and to implement the control plane by deploying the EPC control plane entities in a cloud computing facility, while the data plane is implemented by a distributed collection of OpenFlow switches. The OpenFlow protocol is used to connect the two, with enhancements to support GTP routing. While the EPC architecture already has a split between the control plane and the data plane, in the sense that the serving gateway (S-GW) and the PDN gateway (P-GW) are data plane entities. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 151 of 1100

| No. | '111 Patent Claim 24 | Kempf |
|---|---|---|
| | | while the MME, PCRF, and home subscriber server (HSS) are control plane entities, this split was made at the level of the mobility management pro-tocol, GTP.")<br><br>Kempf at [0040] ("The standard EPC architecture assumes a standard routed IP network for transport on top of which the mobile network entities and protocols are implemented. The enhanced EPC architecture described herein is instead at the level ofIP routing and media access control (MAC) switch-ing. Instead of using L2 routing and L3 internal gateway protocols to distribute IP routing and managing Ethernet and IP routing as a collection of distributed control entities, L2 and L3 routing management is centralized in a cloud facility and the routing is controlled from the cloud facility using the OpenFlow protocol. As used herein, the "OpenFlow proto-col" refers to the OpenFlow network protocol and switching specification defined in the OpenFlow Switch Specification at www.openflowswitch.org a web site hosted by Stanford Uni-versity. As used herein, an "OpenFlow switch" refers to a network element implementing the OpenFlow protocol.)<br><br>Kempf at [0044] ("FIG. 1 is a diagram of one embodiment of an example network with an OpenFlow switch, conforming to the OpenFlow 1.0 specification. The OpenFlow 1.0 protocol enables a controller 101 to connect to an OpenFlow 1.0 enabled switch 109 using a secure channel 103 and control a single forwarding table 107 in the switch 109. The controller 101 is an external software component executed by a remote computing device that enables a user to configure the Open-Flow 1.0 switch 109. The secure channel 103 can be provided by any type of network including a local area network (LAN) or a wide area network (WAN), such as the Internet.")<br><br>Kempf at [0074] ("The operation of the EPC cloud computer system as follows. The UE 1317, E-NodeB 1317, S-GW-C 1307, and P-GW-C signal 1307 to the MME, PCRF, and HSS 1307 using the standard EPC protocols, to establish, modify, and delete bearers and GTP tunnels. This signaling triggers pro-cedure calls with the OpenFlow controller to modify the routing in the EPC as requested. The OpenFlow controller configures the standard OpenFlow switches, the Openflow S-GW-D 1315, and P-GW-D 1311 with flow rules and actions to enable the routing requested by the control plane entities. Details of this configuration are described in further detail herein below.) |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 152 of 1100

| No. | '111 Patent Claim 24 | Kempf |
|---|---|---|
| | | Kempf at [0081] ("In one embodiment, OpenFlow is modified to pro-vide rules for GTP TEID Routing. FIG. 17 is a diagram of one embodiment of the OpenFlow flow table modification for GTP TEID routing. An OpenFlow switch that supports TEID routing matches on the 2 byte (16 bit) collection of header fields and the 4 byte (32 bit) GTP TEID, in addition to other OpenFlow header fields, in at least one flow table ( e.g., the first flow table). The GTP TEID flag can be wildcarded (i.e. matches are "don't care"). In one embodiment, the EPC pro-tocols do not assign any meaning to TEIDs other than as an endpoint identifier for tunnels, like ports in standard UDP/ TCP transport protocols. In other embodiments, the TEIDs can have a correlated meaning or semantics. The GTP header flags field can also be wildcarded, this can be partially matched by combining the following bitmasks: 0xFF00- Match the Message Type field; 0xe0-Match the Version field; 0xl0-Match the PT field; 0x04-Match the E field; 0x02- Match the S field; and 0x0l-Match the PN field.")<br><br>Kempf at [0145] ("In other embodiments, other control protocols can be utilized in place of OpenFlow as described herein. The use of OpenFlow is presented by way of example and not limita-tion. Other control protocols can also be utilized to manage the communication between the control plane and data plane and configuration of the data plane of the split EPC architec-ture. An example of such a protocol is FORCES, an IETF standard protocol for splitting the control plane and forward-ing plane in networks. The FORCES protocol specification is described in RFC 5810. RFC 5812 describes the architecture of a FORCES forwarding element, the equivalent of an Open-Flow switch. The FORCES protocol itself does not directly support programming routes into the forwarding element, it is, instead, a framework for handling the interaction between the FORCES controller and a FORCES forwarding element. The forwarding element architecture describes how to design the protocol that actually allows a FORCES controller to program a FORCES forwarding element. One skilled in the art would understand that a FORCES based system could include features described herein above in relation to the OpenFlow embodiment, such as the GTP OpenFlow exten-sion, to allow the controller to program the switches for GTP TEID routing.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 153 of 1100

| No. | '111 Patent Claim 27 | Kempf |
|---|---|---|
| 27 | The method according to claim 1, wherein the network node comprises a router, a switch, or a bridge. | Kempf discloses the method according to claim 1, wherein the network node comprises a router, a switch, or a bridge.<br><br>For example, Kempf discloses network elements such as a router, switch, bridge, etc.<br><br>*See supra* at Claim 1.<br><br>Kempf at [0033] ("As used herein, a network element (e.g., a router, switch, bridge, etc.) is a piece of networking equipment, including hardware and software, that communicatively interconnects other equipment on the network (e.g., other network elements, end stations, etc.). Some network elements are "multiple services network elements" that provide sup-port for multiple networking functions (e.g., routing, bridg-ing, switching, Layer 2 aggregation, session border control, multicasting, and/or subscriber management), and/or provide support for multiple application services (e.g., data, voice, and video). Subscriber end stations ( e.g., servers, worksta-tions, laptops, palm tops, mobile phones, smart phones, mul-timedia phones, Voice Over Internet Protocol (VOIP) phones, portable media players, GPS units, gaming systems, set-top boxes (STBs), etc.) access content/services provided over the Internet and/or content/services provided on virtual private networks (VPN s) overlaid on the Internet. The content and/or services are typically provided by one or more end stations ( e.g., server end stations) belonging to a service or content provider or end stations participating in a peer to peer service, and may include public web pages (free content, store fronts, search services, etc.), private web pages ( e.g., username/pass-word accessed web pages providing email services, etc.), corporate networks over VPNs, IPTV, etc. Typically, sub-scriber end stations are coupled ( e.g., through customer premise equipment coupled to an access network (wired or wirelessly)) to edge network elements, which are coupled (e.g., through one or more core network elements to other edge network elements) to other end stations (e.g., server end stations).")<br><br>Kempf at [0038] ("Implementing the control plane of an EPC in a cloud computing facility and the data plane of the EPC using a set of OpenFlow switches, as well as managing communication between the control plane and the dataplane using the Open-Flow protocol (e.g., OpenFlow 1.1), creates a problem that the OpenFlow protocol does not support GTP or GTP tunnel endpoint identifier (TEID) routing, which is necessary for implementing the dataplane of the EPC") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 154 of 1100

| No. | '111 Patent Claim 27 | Kempf |
|---|---|---|
| | | Kempf at [0039] ("The embodiments of the invention overcome these disadvantages of the prior art. The disadvantages of the prior art are avoided by splitting the control plane and the data plane for the EPC architecture and to implement the control plane by deploying the EPC control plane entities in a cloud computing facility, while the data plane is implemented by a distributed collection of OpenFlow switches. The OpenFlow protocol is used to connect the two, with enhancements to support GTP routing. While the EPC architecture already has a split between the control plane and the data plane, in the sense that the serving gateway (S-GW) and the PDN gateway (P-GW) are data plane entities while the MME, PCRF, and home subscriber server (HSS) are control plane entities, this split was made at the level of the mobility management pro-tocol, GTP.")<br><br>Kempf at [0040] ("The standard EPC architecture assumes a standard routed IP network for transport on top of which the mobile network entities and protocols are implemented. The enhanced EPC architecture described herein is instead at the level ofIP routing and media access control (MAC) switch-ing. Instead of using L2 routing and L3 internal gateway protocols to distribute IP routing and managing Ethernet and IP routing as a collection of distributed control entities, L2 and L3 routing management is centralized in a cloud facility and the routing is controlled from the cloud facility using the OpenFlow protocol. As used herein, the "OpenFlow proto-col" refers to the OpenFlow network protocol and switching specification defined in the OpenFlow Switch Specification at www.openflowswitch.org a web site hosted by Stanford Uni-versity. As used herein, an "OpenFlow switch" refers to a network element implementing the OpenFlow protocol.")<br><br>Kempf at [0041] ("The standard EPC control plane entities-the MME, PCRF, and HSS-are likewise deployed in the cloud, along with the control plane parts of the S-GW and P-GW, namely, the S-GW-C and the P-GW-C. The data plane con-sists of standard OpenFlow switches with enhancements as needed for routing GTP packets, rather than IP routers and Ethernet switches. At a minimum, the data plane parts of the S-GW and P-GW, namely, the S-GW-Dand the P-GW-D, and the packet routing part of the E-NodeB in the E-UTRAN require OpenFlow enhancements for GTP routing. Addi-tional enhancements for GTP routing may be needed on other switches within the EPC architecture depending on how much fine grained control over the routing an operator requires.") |


~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 155 of 1100

| No. | '111 Patent Claim 27 | Kempf |
|---|---|---|
| | | |

| No. | '111 Patent Claim 28 | Kempf |
|---|---|---|
| 28 | The method according to claim 1, wherein the packet network is an Internet Protocol (IP) network, and the packet is an IP packet. | Kempf discloses the method according to claim 1, wherein the packet network is an Internet Protocol (IP) network, and the packet is an IP packet.<br><br>For example, Kempf discloses routing IP packets on an IP network.<br><br>*See supra* at Claim 1.<br><br>Kempf at [0003] ("The general packet radios system (GPRS) is a sys-tem that is used for transmitting Internet Protocol packets between user devices such as cellular phones and the Internet. The GPRS system includes the GPRS core network, which is an integrated part of the global system for mobile communi-cation (GSM). These systems are widely utilized by cellular phone network providers to enable cellular phone services over large areas.")<br><br>Kempf at [0040] ("The standard EPC architecture assumes a standard routed IP network for transport on top of which the mobile network entities and protocols are implemented. The enhanced EPC architecture described herein is instead at the level ofIP routing and media access control (MAC) switch-ing. Instead of using L2 routing and L3 internal gateway protocols to distribute IP routing and managing Ethernet and IP routing as a collection of distributed control entities, L2 and L3 routing management is centralized in a cloud facility and the routing is controlled from the cloud facility using the OpenFlow protocol. As used herein, the "OpenFlow proto-col" refers to the OpenFlow network protocol and switching specification defined in the OpenFlow Switch Specification at www.openflowswitch.org a web site hosted by Stanford Uni-versity. As used herein, an "OpenFlow switch" refers to a network element implementing the OpenFlow protocol.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 156 of 1100

| No. | '111 Patent Claim 28 | Kempf |
|---|---|---|
| | | Kempf at [0136] ("Characteristics of the GTP bearer are changed using a modify bearer request procedure. Such changes may, for example, include the QoS assigned to the IP packets. This procedure is used in a variety of EPC message sequences, for example, a UE triggered service request.")<br><br>Kempf at [0137] ("FIG. 21 is a diagram of one embodiment of the OpenFlow message sequence for the modify bearer request procedure. As with session creation, the EPC cloud control plane MME issues a modify bearer request message to the SGW-C and the SGW-C issues a modify bearer request mes-sage to the PGW-C. The PGW-C then optionally begins a policy and charging enforcement function (PCEF) initiated Internet Protocol connectivity access network (IP-CAN) ses-sion modification process with the PCRF. When this process completes, the PGW-C issues a GTP routing update RPC to the OpenFlow controller including the new bearer update information. The OpenFlow controller then issues GTP extended OpenFlow messages to the SGW-D, GxOFSes, and the PGW-D.") |

| No. | '111 Patent Claim 29 | Kempf |
|---|---|---|
| 29 | The method according to claim 28, wherein the packet network is an Transmission Control Protocol (TCP) network, and the packet is an TCP packet. | Kempf discloses the method according to claim 28, wherein the packet network is an Transmission Control Protocol (TCP) network, and the packet is an TCP packet.<br><br>For example, Kempf discloses routing TCP packets in a TCP network.<br><br>*See supra* at Claim 28.<br><br>Kempf at [0046] ("A rule 201 contains key fields from several headers in the protocol stack, for example source and destination Ethernet MAC addresses, source and destination IP addresses, IP protocol type number, incoming and outgoing TCP or UDP port numbers. To define a flow, all the available matching fields may be used. But it is also possible to restrict the matching rule to a subset of the available fields by using wildcards for the unwanted fields.")<br><br>Kempf at [0081] ("In one embodiment, OpenFlow is modified to pro-vide rules for GTP TEID Routing. FIG. 17 is a diagram of one embodiment of the OpenFlow flow table modification for GTP TEID routing. An OpenFlow switch that supports TEID routing |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 157 of 1100

| No. | '111 Patent Claim 29 | Kempf |
|---|---|---|
| | | matches on the 2 byte (16 bit) collection of header fields and the 4 byte (32 bit) GTP TEID, in addition to other OpenFlow header fields, in at least one flow table ( e.g., the first flow table). The GTP TEID flag can be wildcarded (i.e. matches are "don't care"). In one embodiment, the EPC pro-tocols do not assign any meaning to TEIDs other than as an endpoint identifier for tunnels, like ports in standard UDP/ TCP transport protocols. In other embodiments, the TEIDs can have a correlated meaning or semantics. The GTP header flags field can also be wildcarded, this can be partially matched by combining the following bitmasks: 0xFF00- Match the Message Type field; 0xe0-Match the Version field; 0xl0-Match the PT field; 0x04-Match the E field; 0x02- Match the S field; and 0x0l-Match the PN field.")<br><br>Kempf at [0089] ("In one embodiment, the system implements a GTP fast path encapsulation virtual port. When requested by the S-GW-C and P-GW-C control plane software running in the cloud computing system, the OpenFlow controller programs the gateway switch to install rules, actions, and TEID hash table entries for routing packets into GTP tunnels via a fast path GTP encapsulation virtual port. The rules match the packet filter for the input side of GTP tunnel's bearer. Typi-cally this will be a 4 tuple of: IP source address; IP destination address; UDP/TCP/SCTP source port; and UDP/TCP/SCTP destination port. The IP source address and destination address are typically the addresses for user data plane traffic, i.e. a UE or Internet service with which a UE is transacting, and similarly with the port numbers. For a rule matching the GTP-U tunnel input side, the associated instructions and are the following:<br><br>Write-Metadata ( GTP-TEID, OxFFFFFFFF)<br>Apply-Actions (Set-Output-Port GTP-Encap-VP")<br><br>Kempf at [0101] ("In one embodiment, the system implements han-dling of user data plane packets requiring GTP-U encapsula-tion with extension headers, sequence numbers, and N-PDU numbers. User data plane packets that require extension head-ers, sequence numbers, or N-PDU numbers during GTP encapsulation require special handling by the software slow path. For these packets, the OpenFlow controller programs a rule matching the 4 tuple: IP source address; IP destination address; UDP/TCP/SCTP source port; and UDP/TCP/SCTP destination port. The instructions for matching packets are: |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 158 of 1100

| No. | '111 Patent Claim 29 | Kempf |
|---|---|---|
| | | Write-Metadata ( GTP-TEID, 0x FFFFFFFF)<br>Apply-Actions (Set-Output-Port LOCAL_GTP _U_ENCAP )") |

| No. | '111 Patent Claim 30 | Kempf |
|---|---|---|
| 30[a] | The method according to claim 1, further comprising: receiving, by the network node from the first entity over the packet network, one or more additional packets; | Kempf discloses the method according to claim 1, further comprising: receiving, by the network node from the first entity over the packet network, one or more additional packets.<br><br>For example, Kempf discloses communication between electronic devices in which additional data packets are sent from one electronic device to another destination device via the network elements.<br><br>*See supra at* Claim 1, 1[c].<br><br>Kempf at [0003] ("The general packet radios system (GPRS) is a sys-tem that is used for transmitting Internet Protocol packets between user devices such as cellular phones and the Internet. The GPRS system includes the GPRS core network, which is an integrated part of the global system for mobile communi-cation (GSM). These systems are widely utilized by cellular phone network providers to enable cellular phone services over large areas.")<br><br>Kempf at [0004] ("The GPRS tunneling protocol (GTP) is an important communication protocol utilized within the GPRS core net-work. GTP enables end user devices ( e.g., cellular phones) in a GSM network to move from place to place while continuing to connect to the Internet. The end user devices are connected to other devices through a gateway GPRS support node (GGSN). The GGSN tracks the end user device's data from the end user device's serving GPRS support node (GGSN) that is handling the session originating from the end user device.")<br><br>Kempf at [0032] ("The techniques shown in the figures can be imple-mented using code and data stored and executed on one or more electronic devices ( e.g., an end station, a network ele-ment, etc.). Such electronic devices store and communicate (internally and/or with other |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 159 of 1100

| No. | '111 Patent Claim 30 | Kempf |
|---|---|---|
| | | electronic devices over a net-work) code and data using non-transitory machine-readable or computer-readable media, such as non-transitory machine-readable or computer-readable storage media ( e.g., magnetic disks; optical disks; random access memory; read only memory; flash memory devices; and phase-change memory). In addition, such electronic devices typically include a set of one or more processors coupled to one or more other compo-nents, such as one or more storage devices, user input/output devices (e.g., a keyboard, a touch screen, and/or a display), and network connections. The coupling of the set of proces-sors and other components is typically through one or more busses and bridges (also termed as bus controllers). The stor-age devices represent one or more non-transitory machine-readable or computer-readable storage media and non-tran-sitory machine-readable or computer-readable communication media. Thus, the storage device of a given electronic device typically stores code and/or data for execu-tion on the set of one or more processors of that electronic device. Of course, one or more parts of an embodiment of the invention may be implemented using different combinations of software, firmware, and/or hardware.")<br><br>Kempf at [0033] ("As used herein, a network element (e.g., a router, switch, bridge, etc.) is a piece of networking equipment, including hardware and software, that communicatively interconnects other equipment on the network (e.g., other network elements, end stations, etc.). Some network elements are "multiple services network elements" that provide sup-port for multiple networking functions (e.g., routing, bridg-ing, switching, Layer 2 aggregation, session border control, multicasting, and/or subscriber management), and/or provide support for multiple application services (e.g., data, voice, and video). Subscriber end stations ( e.g., servers, worksta-tions, laptops, palm tops, mobile phones, smart phones, mul-timedia phones, Voice Over Internet Protocol (VOIP) phones, portable media players, GPS units, gaming systems, set-top boxes (STBs), etc.) access content/services provided over the Internet and/or content/services provided on virtual private networks (VPN s) overlaid on the Internet. The content and/or services are typically provided by one or more end stations ( e.g., server end stations) belonging to a service or content provider or end stations participating in a peer to peer service, and may include public web pages (free content, store fronts, search services, etc.), private web pages ( e.g., username/pass-word accessed web pages providing email services, etc.), corporate networks over VPNs, IPTV, etc. Typically, sub-scriber end stations are coupled ( e.g., through customer premise equipment coupled to an access network (wired or wirelessly)) to edge network elements, |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 160 of 1100

| No. | '111 Patent Claim 30 | Kempf |
|---|---|---|
| | | which are coupled (e.g., through one or more core network elements to other edge network elements) to other end stations (e.g., server end stations).")<br><br>Kempf at [0040] ("The standard EPC architecture assumes a standard routed IP network for transport on top of which the mobile network entities and protocols are implemented. The enhanced EPC architecture described herein is instead at the level ofIP routing and media access control (MAC) switch-ing. Instead of using L2 routing and L3 internal gateway protocols to distribute IP routing and managing Ethernet and IP routing as a collection of distributed control entities, L2 and L3 routing management is centralized in a cloud facility and the routing is controlled from the cloud facility using the OpenFlow protocol. As used herein, the "OpenFlow proto-col" refers to the OpenFlow network protocol and switching specification defined in the OpenFlow Switch Specification at www.openflowswitch.org a web site hosted by Stanford Uni-versity. As used herein, an "OpenFlow switch" refers to a network element implementing the OpenFlow protocol.")<br><br>Kempf at [0079] ("FIG. 16 is a diagram of one embodiment of a process for EPC peering and differential routing for specialized ser-vice treatment. The OpenFlow signaling, indicated by the solid lines and arrows 1601, sets up flow rules and actions on the switches and gateways within the EPC for differential routing. These flow rules direct GTP flows to particular loca-tions. In this example, the operator in this case peers its EPC with two other fixed operators. Routing through each peering point is handled by the respective P-GW-Dl and P-GW-D2 1603A, B. The dashed lines and arrows 1605 show traffic from a UE 1607 that needs to be routed to another peering operator. The flow rules and actions to distinguish which peering point the traffic should traverse are installed in the OpenFlow switches 1609 and gateways 1603A, B by the OpenFlow controller 1611. The OpenFlow controller 1611 calculates these flow rules and actions based on the routing tables it maintains for outside traffic, and the source and destination of the packets, as well as by any specialized for-warding treatment required for DSCP marked packets.") |
| 30[b] | checking, by the network node, if any one of the one or more additional packets satisfies the criterion; | Kempf discloses checking, by the network node, if any one of the one or more additional packets satisfies the criterion.<br><br>*See supra at* Claim 1[d], 30[a]. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 161 of 1100

| No. | '111 Patent Claim 30 | Kempf |
|---|---|---|
| 30[c] | responsive to an additional packet not satisfying the criterion, sending, by the network node over the packet network, the additional packet to the second entity; and | Kempf discloses responsive to an additional packet not satisfying the criterion, sending, by the network node over the packet network, the additional packet to the second entity.<br><br>*See supra at* Claim 1[e], 30[a]. |
| 30[d] | responsive to the additional packet satisfying the criterion, sending the additional packet, by the network node over the packet network, in response to the instruction. | Kempf discloses responsive to the additional packet satisfying the criterion, sending the additional packet, by the network node over the packet network, in response to the instruction.<br><br>*See supra at* Claim 1[f], 30[a]. |

| No. | '111 Patent Claim 31 | Kempf |
|---|---|---|
| 31[a] | The method according to claim 1, wherein the packet network is a Software Defined Network (SDN), | Kempf discloses the method according to claim 1, wherein the packet network is a Software Defined Network (SDN).<br><br>For example, Kempf discloses a packet network using the OpenFlow protocol. A person of ordinary skill in the art would understand that the OpenFlow protocol is used in Software Defined Networks.<br><br>*See supra* at Claim 1.<br><br>Kempf at [0004] ("The GPRS tunneling protocol (GTP) is an important communication protocol utilized within the GPRS core net-work. GTP enables end user devices ( e.g., cellular phones) in a GSM network to move from place to place while continuing to connect to the Internet. The end user devices are connected to other devices through a gateway GPRS support node (GGSN). The GGSN tracks the end user device's data from the end user device's serving GPRS support node (GGSN) that is handling the session originating from the end user device.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 162 of 1100

| No. | '111 Patent Claim 31 | Kempf |
|---|---|---|
| | | Kempf at [0006] ("A method implements a control plane of an evolved packet core (EPC) of a third generation partnership project (3GPP) long term evolution (LTE) network in a cloud com-puting system. The cloud computing system includes a cloud manager and a controller. The controller executes a plurality of control plane modules. The control plane communicates with the data plane of the EPC implemented in a plurality of network elements of the 3GPP LTE network through a control protocol. The EPC with the control plane implemented in the cloud computing system utilizes resources more efficiently than an architecture with the control plane implemented in the plurality of network elements of the 3GPP LTE network. The method comprises the steps of initializing the plurality of control plane modules of the EPC within the controller. Each control plane module in the plurality of control plane modules is initialized as a separate virtual machine by the cloud man-ager. Each control plane module provides a set of control plane functions for managing the data plane. The cloud man-ager monitors resource utilization of each control plane mod-ule and the control plane traffic handled by each control plane module. The cloud manager detects a threshold level of resource utilization or traffic load for one of the plurality of control plane modules of the EPC. A new control plane mod-ule is initialized as a separate virtual machine by the cloud manager in response to detecting the threshold level. The new control plane module shares the load of the one of the plural-ity of control plane modules and signals the plurality of net-work elements in the data plane to establish flow rules and actions to establish differential routing of flows in the data plane using the control protocol, wherein the control protocol is an OpenFlow protocol, and wherein flow matches are encoded using an extensible match structure in which the flow match is encoded as a type-length-value (TLV).")<br><br>Kempf at [0007] ("A cloud computer system implements a control plane of an evolved packet core (EPC) of a third generation partnership project (3GPP) long term evolution (LTE) net-work. The control plane communicates with the data plane of the EPC that is implemented in a plurality of network ele-ments of the 3GPP LTE network through a control protocol. The EPC with the control plane implemented in the cloud computing system utilizes resources more efficiently than an architecture with the control plane implemented in the plu-rality of network elements of the 3GPP LTE network. The cloud computing system, comprises a controller configured to execute a plurality of control plane modules of the EPC, each control plane module configured to provide a set of control plane |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 163 of 1100

| No. | '111 Patent Claim 31 | Kempf |
|---|---|---|
| | | functions for managing the data plane and to signal the plurality of network elements in the data plane to establish flow rules and actions to establish differential rout-ing of flows in the data plane using the control protocol, wherein the control protocol is an OpenFlow protocol, and wherein flow matches are encoded using an extensible match structure in which the flow match is encoded as a type-length-value (TLV) and a cloud manager communicatively coupled to the controller. The cloud manager is configured to initialize each of the plurality of control plane modules within the controller as a separate virtual machine, monitor resource utilization of each control plane module and the control plane traffic handled by each control plane module, detect whether a threshold level ofresource utilization or traffic load has been reached by any of the plurality of control plane modules of the EPC, and initialize a new control plane module as a separate virtual machine in response to detecting the threshold level, the new control plane module to share the load of the one of the plurality of control plane modules that exceeded the threshold level.")<br><br>Kempf at [0038] ("Implementing the control plane of an EPC in a cloud computing facility and the data plane of the EPC using a set of OpenFlow switches, as well as managing communication between the control plane and the dataplane using the Open-Flow protocol (e.g., OpenFlow 1.1), creates a problem that the OpenFlow protocol does not support GTP or GTP tunnel endpoint identifier (TEID) routing, which is necessary for implementing the dataplane of the EPC")<br><br>Kempf at [0039] ("The embodiments of the invention overcome these disadvantages of the prior art. The disadvantages of the prior art are avoided by splitting the control plane and the data plane for the EPC architecture and to implement the control plane by deploying the EPC control plane entities in a cloud computing facility, while the data plane is implemented by a distributed collection of OpenFlow switches. The OpenFlow protocol is used to connect the two, with enhancements to support GTP routing. While the EPC architecture already has a split between the control plane and the data plane, in the sense that the serving gateway (S-GW) and the PDN gateway (P-GW) are data plane entities while the MME, PCRF, and home subscriber server (HSS) are control plane entities, this split was made at the level of the mobility management pro-tocol, GTP.")<br><br>Kempf at [0040] ("The standard EPC architecture assumes a standard routed IP network for transport on top of which the mobile network entities and protocols are implemented. The |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 164 of 1100

| No. | '111 Patent Claim 31 | Kempf |
|---|---|---|
| | | enhanced EPC architecture described herein is instead at the level ofIP routing and media access control (MAC) switch-ing. Instead of using L2 routing and L3 internal gateway protocols to distribute IP routing and managing Ethernet and IP routing as a collection of distributed control entities, L2 and L3 routing management is centralized in a cloud facility and the routing is controlled from the cloud facility using the OpenFlow protocol. As used herein, the "OpenFlow proto-col" refers to the OpenFlow network protocol and switching specification defined in the OpenFlow Switch Specification at www.openflowswitch.org a web site hosted by Stanford Uni-versity. As used herein, an "OpenFlow switch" refers to a network element implementing the OpenFlow protocol.)<br><br>Kempf at [0044] ("FIG. 1 is a diagram of one embodiment of an example network with an OpenFlow switch, conforming to the OpenFlow 1.0 specification. The OpenFlow 1.0 protocol enables a controller 101 to connect to an OpenFlow 1.0 enabled switch 109 using a secure channel 103 and control a single forwarding table 107 in the switch 109. The controller 101 is an external software component executed by a remote computing device that enables a user to configure the Open-Flow 1.0 switch 109. The secure channel 103 can be provided by any type of network including a local area network (LAN) or a wide area network (WAN), such as the Internet.") |
| 31[b] | the packet is routed as part of a data plane and | Kempf discloses the packet is routed as part of a data plane.<br><br>For example, Kempf discloses routing packets on a data plane using a control protocol.<br><br>Kempf at [0006] ("A method implements a control plane of an evolved packet core (EPC) of a third generation partnership project (3GPP) long term evolution (LTE) network in a cloud com-puting system. The cloud computing system includes a cloud manager and a controller. The controller executes a plurality of control plane modules. The control plane communicates with the data plane of the EPC implemented in a plurality of network elements of the 3GPP LTE network through a control protocol. The EPC with the control plane implemented in the cloud computing system utilizes resources more efficiently than an architecture with the control plane implemented in the plurality of network elements of the 3GPP LTE network. The method comprises the steps of initializing the plurality of control plane modules of the EPC within the controller. Each control plane module in the plurality of control plane modules is initialized as a separate virtual machine by the cloud |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 165 of 1100

| No. | '111 Patent Claim 31 | Kempf |
|---|---|---|
| | | man-ager. Each control plane module provides a set of control plane functions for managing the data plane. The cloud man-ager monitors resource utilization of each control plane mod-ule and the control plane traffic handled by each control plane module. The cloud manager detects a threshold level of resource utilization or traffic load for one of the plurality of control plane modules of the EPC. A new control plane mod-ule is initialized as a separate virtual machine by the cloud manager in response to detecting the threshold level. The new control plane module shares the load of the one of the plural-ity of control plane modules and signals the plurality of net-work elements in the data plane to establish flow rules and actions to establish differential routing of flows in the data plane using the control protocol, wherein the control protocol is an OpenFlow protocol, and wherein flow matches are encoded using an extensible match structure in which the flow match is encoded as a type-length-value (TLV).")<br><br>Kempf at [0007] ("A cloud computer system implements a control plane of an evolved packet core (EPC) of a third generation partnership project (3GPP) long term evolution (LTE) net-work. The control plane communicates with the data plane of the EPC that is implemented in a plurality of network ele-ments of the 3GPP LTE network through a control protocol. The EPC with the control plane implemented in the cloud computing system utilizes resources more efficiently than an architecture with the control plane implemented in the plu-rality of network elements of the 3GPP LTE network. The cloud computing system, comprises a controller configured to execute a plurality of control plane modules of the EPC, each control plane module configured to provide a set of control plane functions for managing the data plane and to signal the plurality of network elements in the data plane to establish flow rules and actions to establish differential rout-ing of flows in the data plane using the control protocol, wherein the control protocol is an OpenFlow protocol, and wherein flow matches are encoded using an extensible match structure in which the flow match is encoded as a type-length-value (TLV) and a cloud manager communicatively coupled to the controller. The cloud manager is configured to initialize each of the plurality of control plane modules within the controller as a separate virtual machine, monitor resource utilization of each control plane module and the control plane traffic handled by each control plane module, detect whether a threshold level ofresource utilization or traffic load has been reached by any of the plurality of control plane modules of the EPC, and initialize a new control plane module as a separate virtual machine in response to detecting |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 166 of 1100

| No. | '111 Patent Claim 31 | Kempf |
|---|---|---|
| | | the threshold level, the new control plane module to share the load of the one of the plurality of control plane modules that exceeded the threshold level.")<br><br>Kempf at [0038] ("Implementing the control plane of an EPC in a cloud computing facility and the data plane of the EPC using a set of OpenFlow switches, as well as managing communication between the control plane and the dataplane using the Open-Flow protocol (e.g., OpenFlow 1.1), creates a problem that the OpenFlow protocol does not support GTP or GTP tunnel endpoint identifier (TEID) routing, which is necessary for implementing the dataplane of the EPC")<br><br>Kempf at [0039] ("The embodiments of the invention overcome these disadvantages of the prior art. The disadvantages of the prior art are avoided by splitting the control plane and the data plane for the EPC architecture and to implement the control plane by deploying the EPC control plane entities in a cloud computing facility, while the data plane is implemented by a distributed collection of OpenFlow switches. The OpenFlow protocol is used to connect the two, with enhancements to support GTP routing. While the EPC architecture already has a split between the control plane and the data plane, in the sense that the serving gateway (S-GW) and the PDN gateway (P-GW) are data plane entities while the MME, PCRF, and home subscriber server (HSS) are control plane entities, this split was made at the level of the mobility management pro-tocol, GTP.")<br><br>Kempf at [0040] ("The standard EPC architecture assumes a standard routed IP network for transport on top of which the mobile network entities and protocols are implemented. The enhanced EPC architecture described herein is instead at the level ofIP routing and media access control (MAC) switch-ing. Instead of using L2 routing and L3 internal gateway protocols to distribute IP routing and managing Ethernet and IP routing as a collection of distributed control entities, L2 and L3 routing management is centralized in a cloud facility and the routing is controlled from the cloud facility using the OpenFlow protocol. As used herein, the "OpenFlow proto-col" refers to the OpenFlow network protocol and switching specification defined in the OpenFlow Switch Specification at www.openflowswitch.org a web site hosted by Stanford Uni-versity. As used herein, an "OpenFlow switch" refers to a network element implementing the OpenFlow protocol.) |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 167 of 1100

| No. | '111 Patent Claim 31 | Kempf |
|---|---|---|
| | | Kempf at [0041] ("The standard EPC control plane entities-the MME, PCRF, and HSS-are likewise deployed in the cloud, along with the control plane parts of the S-GW and P-GW, namely, the S-GW-C and the P-GW-C. The data plane con-sists of standard OpenFlow switches with enhancements as needed for routing GTP packets, rather than IP routers and Ethernet switches. At a minimum, the data plane parts of the S-GW and P-GW, namely, the S-GW-Dand the P-GW-D, and the packet routing part of the E-NodeB in the E-UTRAN require OpenFlow enhancements for GTP routing. Addi-tional enhancements for GTP routing may be needed on other switches within the EPC architecture depending on how much fine grained control over the routing an operator requires.")<br><br>Kempf at [0078] ("FIG. 15 is a diagram of one embodiment of how the EPC in the cloud computing system enables a managed ser-vices company to manage multiple operator networks out of a single data center. The managed services cloud computing facility 1501 runs separate instances of the EPC control plane for every mobile operator with which the managed services company has a contract. Each EPC instance is in a VPC 1503A,B that isolates the mobile operator's traffic from other tenants in the cloud computing facility 1501 of the data cen-ter. The EPC control plane instance for a mobile operator is connected to the mobile operator's geographically distributed EPC OpenFlow data plane switching fabric 1507 A,B and the mobile operator's base stations through a virtual edge router 1509A,B. The virtual edge router 1509A,B routes traffic from the data center to and from the appropriate mobile operator EPC data plane switching fabric 1507 A,B. In some cases, the mobile operators may even share base stations and EPC switching fabrics, though the example embodiment in FIG. 15 shows a case where the two mobile operators have separate switching fabrics.")<br><br>Kempf at [0087] ("In one embodiment, slow path support for GTP is implemented with an OpenFlow gateway switch. An Open-Flow mobile gateway switch also contains support on the software control plane for slow path packet processing. This path is taken by G-PDU (message type 255) packets with nonzero header fields or extension headers, and user data plane packets requiring encapsulation with such fields or addition of extension headers, and by G TP-U control packets. For this purpose, the switch supports three local ports in the software control plane: LOCAL_GTP _CONTROL-the switch fast path forwards GTP encapsulated packets directed to the gateway IP address that contain GTP-U control mes-sages and the local switch software control plane initiates local control plane actions. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 168 of 1100

| No. | '111 Patent Claim 31 | Kempf |
|---|---|---|
| | | depending on the GTP-U control message; LOCAL_GTP _U_DECAP-the switch fast path forwards G-PDU packets to this port that have nonzero header fields or extension headers (i.e. E!=0, S!=0, or PN!=0). These packets require specialized handling. The local switch software slow path processes the packets and performs the specialized handling; and LOCAL_GTP _U_ENCAP-the switch fast path forwards user data plane packets to this port that require encapsulation in a GTP tunnel with nonzero header fields or extension headers (i.e. E!=0, S!=0, or PN!=0). These packets require specialized handling. The local switch software slow path encapsulates the packets and performs the specialized handling. In addition to forwarding the packet, the switch fast path makes the OpenFlow metadata field avail-able to the slow path software.")<br><br>Kempf at [0093] ("The virtual port simply removes the GTP tunnel header and forwards the enclosed user data plane packet out the bound physical port.")<br><br>Kempf at [0101] ("In one embodiment, the system implements han-dling of user data plane packets requiring GTP-U encapsula-tion with extension headers, sequence numbers, and N-PDU numbers. User data plane packets that require extension head-ers, sequence numbers, or N-PDU numbers during GTP encapsulation require special handling by the software slow path. For these packets, the OpenFlow controller programs a rule matching the 4 tuple: IP source address; IP destination address; UDP/TCP/SCTP source port; and UDP/TCP/SCTP destination port. The instructions for matching packets are:<br><br>Write-Metadata ( GTP-TEID, 0x FFFFFFFF)<br>Apply-Actions (Set-Output-Port LOCAL_GTP _U_ENCAP )")<br><br>Kempf at [0145] ("In other embodiments, other control protocols can be utilized in place of OpenFlow as described herein. The use of OpenFlow is presented by way of example and not limita-tion. Other control protocols can also be utilized to manage the communication between the control plane and data plane and configuration of the data plane of the split EPC architec-ture. An example of such a protocol is FORCES, an IETF standard protocol for splitting the control plane and forward-ing plane in networks. The FORCES protocol specification is described in RFC 5810. RFC 5812 describes the architecture of a FORCES forwarding element, the equivalent of an Open-Flow switch. The FORCES protocol itself does not directly support programming routes into the forwarding element, it is, instead, a |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 169 of 1100

| No. | '111 Patent Claim 31 | Kempf |
|---|---|---|
| | | framework for handling the interaction between the FORCES controller and a FORCES forwarding element. The forwarding element architecture describes how to design the protocol that actually allows a FORCES controller to program a FORCES forwarding element. One skilled in the art would understand that a FORCES based system could include features described herein above in relation to the OpenFlow embodiment, such as the GTP OpenFlow exten-sion, to allow the controller to program the switches for GTP TEID routing.") |
| 31[c] | the network node communication with the controller serves as a control plane. | Kempf discloses the network node communication with the controller serves as a control.<br><br>For example, Kempf discloses communication between network elements and an OpenFlow controller over a control plane.<br><br>Kempf at [0006] ("A method implements a control plane of an evolved packet core (EPC) of a third generation partnership project (3GPP) long term evolution (LTE) network in a cloud com-puting system. The cloud computing system includes a cloud manager and a controller. The controller executes a plurality of control plane modules. The control plane communicates with the data plane of the EPC implemented in a plurality of network elements of the 3GPP LTE network through a control protocol. The EPC with the control plane implemented in the cloud computing system utilizes resources more efficiently than an architecture with the control plane implemented in the plurality of network elements of the 3GPP LTE network. The method comprises the steps of initializing the plurality of control plane modules of the EPC within the controller. Each control plane module in the plurality of control plane modules is initialized as a separate virtual machine by the cloud man-ager. Each control plane module provides a set of control plane functions for managing the data plane. The cloud man-ager monitors resource utilization of each control plane mod-ule and the control plane traffic handled by each control plane module. The cloud manager detects a threshold level of resource utilization or traffic load for one of the plurality of control plane modules of the EPC. A new control plane mod-ule is initialized as a separate virtual machine by the cloud manager in response to detecting the threshold level. The new control plane module shares the load of the one of the plural-ity of control plane modules and signals the plurality of net-work elements in the data plane to establish flow rules and actions to establish differential routing of flows in the data plane using the control |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 170 of 1100

| No. | '111 Patent Claim 31 | Kempf |
|---|---|---|
| | | protocol, wherein the control protocol is an OpenFlow protocol, and wherein flow matches are encoded using an extensible match structure in which the flow match is encoded as a type-length-value (TLV).")<br><br>Kempf at [0007] ("A cloud computer system implements a control plane of an evolved packet core (EPC) of a third generation partnership project (3GPP) long term evolution (LTE) net-work. The control plane communicates with the data plane of the EPC that is implemented in a plurality of network ele-ments of the 3GPP LTE network through a control protocol. The EPC with the control plane implemented in the cloud computing system utilizes resources more efficiently than an architecture with the control plane implemented in the plu-rality of network elements of the 3GPP LTE network. The cloud computing system, comprises a controller configured to execute a plurality of control plane modules of the EPC, each control plane module configured to provide a set of control plane functions for managing the data plane and to signal the plurality of network elements in the data plane to establish flow rules and actions to establish differential rout-ing of flows in the data plane using the control protocol, wherein the control protocol is an OpenFlow protocol, and wherein flow matches are encoded using an extensible match structure in which the flow match is encoded as a type-length-value (TLV) and a cloud manager communicatively coupled to the controller. The cloud manager is configured to initialize each of the plurality of control plane modules within the controller as a separate virtual machine, monitor resource utilization of each control plane module and the control plane traffic handled by each control plane module, detect whether a threshold level ofresource utilization or traffic load has been reached by any of the plurality of control plane modules of the EPC, and initialize a new control plane module as a separate virtual machine in response to detecting the threshold level, the new control plane module to share the load of the one of the plurality of control plane modules that exceeded the threshold level.")<br><br>Kempf at [0038] ("Implementing the control plane of an EPC in a cloud computing facility and the data plane of the EPC using a set of OpenFlow switches, as well as managing communication between the control plane and the dataplane using the Open-Flow protocol (e.g., OpenFlow 1.1), creates a problem that the OpenFlow protocol does not support GTP or GTP tunnel endpoint identifier (TEID) routing, which is necessary for implementing the dataplane of the EPC") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 171 of 1100

| No. | '111 Patent Claim 31 | Kempf |
|---|---|---|
| | | Kempf at [0039] ("The embodiments of the invention overcome these disadvantages of the prior art. The disadvantages of the prior art are avoided by splitting the control plane and the data plane for the EPC architecture and to implement the control plane by deploying the EPC control plane entities in a cloud computing facility, while the data plane is implemented by a distributed collection of OpenFlow switches. The OpenFlow protocol is used to connect the two, with enhancements to support GTP routing. While the EPC architecture already has a split between the control plane and the data plane, in the sense that the serving gateway (S-GW) and the PDN gateway (P-GW) are data plane entities while the MME, PCRF, and home subscriber server (HSS) are control plane entities, this split was made at the level of the mobility management pro-tocol, GTP.")<br><br>Kempf at [0040] ("The standard EPC architecture assumes a standard routed IP network for transport on top of which the mobile network entities and protocols are implemented. The enhanced EPC architecture described herein is instead at the level ofIP routing and media access control (MAC) switch-ing. Instead of using L2 routing and L3 internal gateway protocols to distribute IP routing and managing Ethernet and IP routing as a collection of distributed control entities, L2 and L3 routing management is centralized in a cloud facility and the routing is controlled from the cloud facility using the OpenFlow protocol. As used herein, the "OpenFlow proto-col" refers to the OpenFlow network protocol and switching specification defined in the OpenFlow Switch Specification at www.openflowswitch.org a web site hosted by Stanford Uni-versity. As used herein, an "OpenFlow switch" refers to a network element implementing the OpenFlow protocol.)<br><br>Kempf at [0041] ("The standard EPC control plane entities-the MME, PCRF, and HSS-are likewise deployed in the cloud, along with the control plane parts of the S-GW and P-GW, namely, the S-GW-C and the P-GW-C. The data plane con-sists of standard OpenFlow switches with enhancements as needed for routing GTP packets, rather than IP routers and Ethernet switches. At a minimum, the data plane parts of the S-GW and P-GW, namely, the S-GW-Dand the P-GW-D, and the packet routing part of the E-NodeB in the E-UTRAN require OpenFlow enhancements for GTP routing. Addi-tional enhancements for GTP routing may be needed on other switches within the EPC architecture depending on how much fine grained control over the routing an operator requires.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 172 of 1100

| No. | '111 Patent Claim 31 | Kempf |
|---|---|---|
| | | Kempf at [0078] ("FIG. 15 is a diagram of one embodiment of how the EPC in the cloud computing system enables a managed ser-vices company to manage multiple operator networks out of a single data center. The managed services cloud computing facility 1501 runs separate instances of the EPC control plane for every mobile operator with which the managed services company has a contract. Each EPC instance is in a VPC 1503A,B that isolates the mobile operator's traffic from other tenants in the cloud computing facility 1501 of the data cen-ter. The EPC control plane instance for a mobile operator is connected to the mobile operator's geographically distributed EPC OpenFlow data plane switching fabric 1507 A,B and the mobile operator's base stations through a virtual edge router 1509A,B. The virtual edge router 1509A,B routes traffic from the data center to and from the appropriate mobile operator EPC data plane switching fabric 1507 A,B. In some cases, the mobile operators may even share base stations and EPC switching fabrics, though the example embodiment in FIG. 15 shows a case where the two mobile operators have separate switching fabrics.")<br><br>Kempf at [0087] ("In one embodiment, slow path support for GTP is implemented with an OpenFlow gateway switch. An Open-Flow mobile gateway switch also contains support on the software control plane for slow path packet processing. This path is taken by G-PDU (message type 255) packets with nonzero header fields or extension headers, and user data plane packets requiring encapsulation with such fields or addition of extension headers, and by G TP-U control packets. For this purpose, the switch supports three local ports in the software control plane: LOCAL_GTP _CONTROL-the switch fast path forwards GTP encapsulated packets directed to the gateway IP address that contain GTP-U control mes-sages and the local switch software control plane initiates local control plane actions depending on the GTP-U control message; LOCAL_GTP _U_DECAP-the switch fast path forwards G-PDU packets to this port that have nonzero header fields or extension headers (i.e. E!=0, S!=0, or PN!=0). These packets require specialized handling. The local switch software slow path processes the packets and performs the specialized handling; and LOCAL_GTP _U_ENCAP-the switch fast path forwards user data plane packets to this port that require encapsulation in a GTP tunnel with nonzero header fields or extension headers (i.e. E!=0, S!=0, or PN!=0). These packets require specialized handling. The local switch software slow path encapsulates the packets and performs the specialized handling. In addition to forwarding the packet, the switch fast path makes the OpenFlow metadata field avail-able to the slow path software.") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 173 of 1100

| No. | '111 Patent Claim 31 | Kempf |
|---|---|---|
| | | Kempf at [0093] ("The virtual port simply removes the GTP tunnel header and forwards the enclosed user data plane packet out the bound physical port.")<br><br>Kempf at [0101] ("In one embodiment, the system implements han-dling of user data plane packets requiring GTP-U encapsula-tion with extension headers, sequence numbers, and N-PDU numbers. User data plane packets that require extension head-ers, sequence numbers, or N-PDU numbers during GTP encapsulation require special handling by the software slow path. For these packets, the OpenFlow controller programs a rule matching the 4 tuple: IP source address; IP destination address; UDP/TCP/SCTP source port; and UDP/TCP/SCTP destination port. The instructions for matching packets are:<br><br>Write-Metadata ( GTP-TEID, 0x FFFFFFFF)<br>Apply-Actions (Set-Output-Port LOCAL_GTP _U_ENCAP )")<br><br>Kempf at [0145] ("In other embodiments, other control protocols can be utilized in place of OpenFlow as described herein. The use of OpenFlow is presented by way of example and not limita-tion. Other control protocols can also be utilized to manage the communication between the control plane and data plane and configuration of the data plane of the split EPC architec-ture. An example of such a protocol is FORCES, an IETF standard protocol for splitting the control plane and forward-ing plane in networks. The FORCES protocol specification is described in RFC 5810. RFC 5812 describes the architecture of a FORCES forwarding element, the equivalent of an Open-Flow switch. The FORCES protocol itself does not directly support programming routes into the forwarding element, it is, instead, a framework for handling the interaction between the FORCES controller and a FORCES forwarding element. The forwarding element architecture describes how to design the protocol that actually allows a FORCES controller to program a FORCES forwarding element. One skilled in the art would understand that a FORCES based system could include features described herein above in relation to the OpenFlow embodiment, such as the GTP OpenFlow exten-sion, to allow the controller to program the switches for GTP TEID routing.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 174 of 1100

**EXHIBIT D-2**
Defendant's Preliminary Invalidity Contentions
*Orckit Corporation v. Cisco Systems, Inc.*, 2:22-cv-00276-JRG-RSP

---

**Chart for U.S. Patent 10,652,111 ("the '111 Patent")**
**U.S. Patent Publication No. 2013/0322242 to Swenson et al. ("Swenson")**

As shown in the chart below, all Asserted Claims of the '111 Patent are invalid under (1) AIA-35 U.S.C. § 102 (a) because Swenson meets each element of those claims, and/or (2) 35 U.S.C. § 103 because Swenson renders those claims obvious either alone, or in combination with the knowledge of a person having ordinary skill in the art, and in further combination with the references specifically identified below and in the following claim chart and/or one or more references identified in Defendant's Preliminary Invalidity Contentions. The following quotations and diagrams come from Swenson titled "Real-Time Network Monitoring And Subscriber Identification With An On-Demand Appliance", which was filed on May 31, 2013, and published on December 5, 2013.

Motivations to combine the disclosures in Swenson with disclosures in other publications known in the art, as explained in this chart, include at least the similarity in subject matter between the references to the extent they concern methods relating to routing certain network traffic to entities for further analysis and inspection. Insofar as the references cite other patents or publications, or suggest additional changes, one of ordinary skill in the art would look beyond a single reference to other references in the field.

These invalidity contentions are based on Defendant's present understanding of the Asserted Claims, and Orckit's apparent construction of the claims in its November 3, 2022 Disclosure of Asserted Claims and Infringement Contentions Pursuant to P.R. 3-1, and Orckit's January 19, 2023 First Amended Disclosure of Asserted Claims and Infringement Contentions Pursuant to P.R. 3-1 (Orckit's "Infringement Disclosures"), which is deficient at least insofar as it fails to cite any documents or identify accused structures, acts, or materials in the Accused Products with particularity. Defendant does not agree with Orckit's application of the claims, or that the claims satisfy the requirements of 35 U.S.C. § 112. Defendant's contentions herein are not, and should in no way be seen as, admissions or adoptions as to any particular claim scope or construction, or as any admission that any particular element is met by any accused product in any particular way. Defendant objects to any attempt to imply claim construction from this chart. Defendant's prior art invalidity contentions are made in a variety of alternatives and do not represent Defendant's agreement or view as to the meaning, definiteness, written description support for, or enablement of any claim contained therein.



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 175 of 1100

The following contentions are subject to revision and amendment pursuant to Federal Rule of Civil Procedure 26(e), the Local Rules, and the Orders of record in this matter subject to further investigation and discovery regarding the prior art and the Court's construction of the claims at issue.

| No. | '111 Patent Claim 1 | Swenson |
|---|---|---|
| 1[preamble] | A method for use with a packet network including a network node for transporting packets between first and second entities under control of a controller that is external to the network node, the method comprising: | Swenson discloses a method for use with a packet network including a network node for transporting packets between first and second entities under control of a controller that is external to the network node, the method comprising.<br><br>For example, Swenson discloses a method for monitoring and steering traffic using a steering device that provides a gateway between user devices and servers under the control of an external network controller. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met.<br><br>Swenson at Abstract ("A system and a method are disclosed for selectively monitor-ing traffic in a service provider network. The system receives a notice for a beginning of a network data flow, which responds to a request from a user device for content at an origin server. The system then determines whether to monitor the data flow from the origin server to the user device. If so determined, the system collects statistic information of the data flow and stores the statistic information to a flow record in a database. The system also maps the flow record to a subscriber of the service provider network by analyzing the statistic information of the data flow and estimates bandwidth provided to the data flow by the service provider's network based on the analysis of the statistic information of the data flow.")<br><br>Swenson at [0018] ("Embodiments disclosed include a network control-ler system for real-time data gathering on the state of existing network traffic flows and mapping flow data to respective users in the network to predict available bandwidth and level of congestion. By gathering a history of flow statistics in the network, the network controller system establishes a relation-ship between base stations ( or other network segments) and their capability to deliver the amount of data typically required by a particular user of the network. The very recent history of network flows can be used to predict the near future congestions in a substantially real-time fashion. Furthermore, the history of flow statistics can be used to build a long-term map of user behavior on the network, which can more effec-tively predict on demand data delivery requirements for the collection of users |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 176 of 1100

| No. | '111 Patent Claim 1 | Swenson |
|---|---|---|
| | | utilizing a given network access point in a consistent manner. The network controller keeps a flow state database, which groups flow data in a number of ways, such as on per station/cell tower, per subscriber, per time-of-day, or per geography area basis. As new flows are presented to the system for inspection, database can be queried to estimate the network congestion level for new flows to determine whether existing, new or future flows require optimizations in order to maintain the desired level of user satisfaction.")<br><br>Swenson at [0019] ("In one embodiment, an on-demand network moni-toring method is adopted to gather data about network flows as they traverse the network. For example, network flows can be monitored selectively or on-demand based on the types of the content carried in the flows. Furthermore, the network monitoring can also be performed selectively at inline level, as well as out-of-band to improve efficiency. Both TCP and UDP flows are monitored to gather information about the state of the network, such as the average network throughput for each flow and end-to-end latency between, for example, a client device and an origin server providing multimedia con-tent to the client device. For each TCP or UDP flow, the system tracks the number of bytes sent ( and in some embodi-ments acknowledged). In TCP, the current window size may also be tracked. Records on network flows are stored in a flow statistics database, which can be indexed by subscriber iden-tification (ID), cell tower (base station), and network segment etc. As many flow records accumulate, this database repre-sents both historical and current network condition and capacity for delivering data. Network throughput can be mea-sured by calculating an average number of bytes delivered over a period of time. Steps may be taken to filter out spurious data from small flows with size less than a certain threshold that, when measured, cause very noisy results in measuring bandwidth and/or latency. For example, any flow having delivery time of less than 500 ms can be filtered.")<br><br>Swenson at [0023] ("FIG. 1 illustrates a high-level block diagram of an example communications environment 100 for selective on-demand real-time network monitoring and subscriber identi-fication. The environment 100 comprises user devices 110, an origin server 160, a steering device 130, a network controller 140, a video optimizer 150, and a network 120. The network 120 is a communication network that transmits data between the user devices 110, the steering device 130 and the origin server 160 and/or the video optimizer 150. In one embodi-ment the network 120 includes wireless network and the Internet.") |


~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 177 of 1100

| No. | '111 Patent Claim 1 | Swenson |
|---|---|---|
| | | Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0028] ("In contrast to conventional inline TCP throughput monitoring devices that monitor every single data packets transmitted and received, the network controller 140 is an "out-of-band" computer server that interfaces with the steer-ing device 130 to selectively inspect user flows of interest. The network controller 140 may further identify user flows (e.g., among the flows of interest) for optimization. In one embodiment, the network controller 140 may be imple-mented at the steering device 130 to monitor traffic. In other embodiments, the network controller 140 is coupled to and communicates with the steering device 130 for traffic moni-toring and optimization. When queried by the steering device 130, the network controller 140 determines if a given network flow should be ignored, monitored further or optimized. Opti-mization of a flow is often decided at the beginning of the flow because it is rarely possible to switch to optimized content mid-stream once non-optimized content delivery has begun. However, the network controller 140 may determine that existing flows associated with a particular subscriber or other entity should be optimized. In turn, new flows ( e.g., resulting from seek requests in media, new media requests, resume after pause, etc.) determined to be associated with the entity may be optimized. The network controller 140 uses the net-work state as well as historical traffic data in its decision for monitoring and optimization. Knowledge on the current net-work state, such as congestion, deems critical when it comes to data optimization.")<br><br>Swenson at [0045] ("The steering device interface 316 interacts with an external routing appliance, such as the steering device 130 to divert portions of the network traffic ( e.g., large object net-work flows). Existing routing appliances in most carrier net-works are |

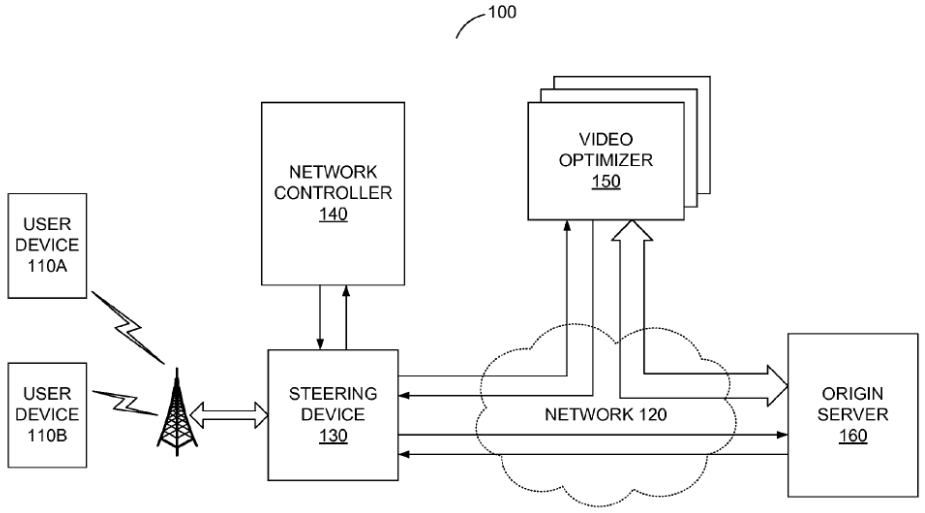


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 178 of 1100

| No. | '111 Patent Claim 1 | Swenson |
|---|---|---|
| | | designed to handle large amounts of network traf-fic. They are not, however, ideal devices to operate for monitoring and analysis individual flows. Through the steer-ing device interface 316, the network controller 140 may communicate with the external routing appliances, such as the steering device 130, to steer a portion of network traffic to the network controller 140 when certain conditions are met. Generally, network flows of interest to the network controller 140 contain larger media objects, such as videos and images. In one embodiment, the smaller flows, such as web page and text information, are not exchanged over the steering device interface 316.")<br><br>Swenson at [0056] ("FIGS. 4A and 4B each illustrates one embodiment of an example working mode of the network controller for providing selective on-demand real-time network monitoring and subscriber identification. Shown with the network con-troller 140 are the user device 110, the steering device 130, and the origin server 160. The network controller 140 is coupled to the steering device 130 through the steering device interface 316. In one embodiment, the network controller 140 and the steering device 130 communicate with each other using the Internet content adaption protocol (ICAP). The steering device interface 316 executes an ICAP server 406, which interacts with an ICAP client 404 running on the steer-ing device 130. Similar or different protocols may be used for communication between the network controller 140 and the steering device 130 in other embodiments.")<br><br>Swenson at Figure 1 |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 179 of 1100

| No. | '111 Patent Claim 1 | Swenson |
|---|---|---|
| | |  <br>**FIG. 1** |
| 1[a] | sending, by the controller to the network node over the packet network, an instruction and a packet-applicable criterion; | Swenson discloses sending, by the controller to the network node over the packet network, an instruction and a packet-applicable criterion.<br><br>For example, Swenson discloses sending instructions by the network controller to the steering device in which the steering device categorizes packet flows based on certain desired criteria or conditions. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 180 of 1100

| No. | '111 Patent Claim 1 | Swenson |
|---|---|---|
| | | interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0027] ("However, information on the wireless/cellular user devices 110 side is often not available at the steering device 130 that sits between the cellular network and the wired Internet. For example, there is often no information about the identifiers of the towers associated with the mobile devices 110. Tower association information only broadcasted when the mobile devices first attached to the network. In addition, user devices 110 do not usually report any identification information except their IP addresses. Therefore, monitoring of the network traffic and detection of the congestion is auto-mated and managed by the detector 140 so that network can be optimized for end user's experience without the mobile user's knowledge.")<br><br>Swenson at [0028] ("In contrast to conventional inline TCP throughput monitoring devices that monitor every single data packets transmitted and received, the network controller 140 is an "out-of-band" computer server that interfaces with the steer-ing device 130 to selectively inspect user flows of interest. The network controller 140 may further identify user flows (e.g., among the flows of interest) for optimization. In one embodiment, the network controller 140 may be imple-mented at the steering device 130 to monitor traffic. In other embodiments, the network controller 140 is coupled to and communicates with the steering device 130 for traffic moni-toring and optimization. When queried by the steering device 130, the network controller 140 determines if a given network flow should be ignored, monitored further or optimized. Opti-mization of a flow is often decided at the beginning of the flow because it is rarely possible to switch to optimized content mid-stream once non-optimized content delivery has begun. However, the network controller 140 may determine that existing flows associated with a particular subscriber or other entity should be optimized. In turn, new flows ( e.g., resulting from seek requests in media, new media requests, resume after pause, etc.) determined to be associated with the entity may be optimized. The network controller 140 uses the net-work state as well as historical traffic data in its decision for monitoring and optimization. Knowledge on the current net-work state, such as congestion, deems critical when it comes to data optimization.") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 181 of 1100

| No. | '111 Patent Claim 1 | Swenson |
|---|---|---|
| | | Swenson at [0038] ("Turning back to FIG. 1, the network controller 140 allows network operators to apply fine granular optimization policies to ensure high quality of experience (QoE) based on cell tower congestion, device types, subscriber profiles and service plans with lower hardware and software costs. The architecture of the network controller 140 provides an excel-lent fit for the net neutrality guideline of"reasonable network management", and better compliance to the copyright law (DMCA) than solutions that rely on long-term caching. Hav-ing the ability of monitoring network traffic on a per sub-scriber, per flow, or per video file basis, the network controller 140 also selectively monitors and optimizes only a subset of traffic that benefits from optimization the most, thus achiev-ing both scalability and efficiency for optimization at a com-petitive price-point. The core element of the network control-ler 140 lies in its mechanisms for congestion detection and mitigation, which allows optimization resources to be utilized in the most efficient and surgical manner.")<br><br>Swenson at [0039] ("Referring now to FIG. 3, it illustrates one embodi-ment of an example architecture of the network controller 140 for providing selective real-time network monitoring and subscriber identification. The network controller 140 com-prises a flow analyzer 312, a policy engine 314, a steering device interface 316, a video optimizer redirector 318, a flow cache 322, and a subscriber log 324. In other embodiments, the network controller 140 may include additional, fewer, or different components for various applications. Conventional components such as network interfaces, security functions, failover servers, management and network operations con-soles, and the like are not shown so as to not obscure the details of the system architecture.")<br><br>Swenson at [0040] ("The flow analyzer 312 monitors large flows in the network, analyzes collected flow statistics to determine net-work throughput, and accordingly selects flows to be opti-mized. The flow analyzer 312 does not need to see all the flows in order to make an accurate estimate of network con-ditions. The flow analyzer 312 processes the traffic statistics stored in the flow cache 3 22 and user information stored in the subscriber log 324, for example, by associating network flows identified by source IP addresses to a mobile subscriber or user, which is identified by his or her current subscriber ID or device ID. The user flows are also mapped to a congestion level at the current sub-network (e.g., a cell with which the user devices are associated), so that an optimization decision can be made at the beginning of the data transmission.") |


~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 182 of 1100

| No. | '111 Patent Claim 1 | Swenson |
|---|---|---|
| | | Swenson at [0045] ("The steering device interface 316 interacts with an external routing appliance, such as the steering device 130 to divert portions of the network traffic ( e.g., large object net-work flows). Existing routing appliances in most carrier net-works are designed to handle large amounts of network traf-fic. They are not, however, ideal devices to operate for monitoring and analysis individual flows. Through the steer-ing device interface 316, the network controller 140 may communicate with the external routing appliances, such as the steering device 130, to steer a portion of network traffic to the network controller 140 when certain conditions are met. Generally, network flows of interest to the network controller 140 contain larger media objects, such as videos and images. In one embodiment, the smaller flows, such as web page and text information, are not exchanged over the steering device interface 316.")<br><br>Swenson at [0073] ("FIG. 7 is a block diagram illustrating one embodi-ment of an example of internal components of the flow cache. The flow cache map 700 comprises a plurality of flow cache entries, such as flow cache entries 710 and 712 indexed by a hash. Not shown in the example diagram is a possible linked list behind each flow cache entry which allows chaining of flow cache entries for a given hash index. The hash into the flow cache may be based on source IP address, MAC address, subscriber ID, or other identifier indicative of a given sub-scriber, group of subscribers or subscriber's device.") |
| 1[b] | receiving, by the network node from the controller, the instruction and the criterion; | Swenson discloses receiving, by the network node from the controller, the instruction and the criterion.<br><br>*See supra at* 1[a]. |
| 1[c] | receiving, by the network node from the first entity over the packet network, a packet addressed to the second entity; | Swenson discloses receiving, by the network node from the first entity over the packet network, a packet addressed to the second entity.<br><br>For example, Swenson discloses receiving network traffic flows from the user device at the steering device that is intended to be transmitted to the origin server.<br><br>Swenson at [0005] ("Mobile devices, such as smart phones and tablets, have become prevalent in recent years. Given the fast advance in mobile computing power and far-reaching wireless Inter-net access, more and more users view streamed videos on their |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 183 of 1100

| No. | '111 Patent Claim 1 | Swenson |
|---|---|---|
| | | mobile devices. The detection of network congestion has become increasingly important for network operators attempting to maximize user experience on the network. Even as network operators are ever increasing the capacity of their networks, the demand for bandwidth is growing at an even faster pace. Managing network growth and dealing with con-gestion in the infrastructure is particularly important in the mobile space because of the high cost of radio spectrum and radio access network (RAN) equipment utilized by wireless mobile networks. These high costs prevent mobile service providers from engineering excess capacity into each net-work access point through the purchase of additional RAN infrastructure. The same situation can, however, also happens to other types of network infrastructure.")<br><br>Swenson at [0023] ("FIG. 1 illustrates a high-level block diagram of an example communications environment 100 for selective on-demand real-time network monitoring and subscriber identi-fication. The environment 100 comprises user devices 110, an origin server 160, a steering device 130, a network controller 140, a video optimizer 150, and a network 120. The network 120 is a communication network that transmits data between the user devices 110, the steering device 130 and the origin server 160 and/or the video optimizer 150. In one embodi-ment the network 120 includes wireless network and the Internet.")<br><br>Swenson at [0025] ("In one embodiment, the user devices 110 are com-puting devices with network capabilities. Oftentimes, for example, the user devices 110 are wireless enabled mobile computing device with a web browser and media display capability. The user devices 110 as mobile computing devices may include laptops, netbooks, tablets, smart telephones, or personal digital assistants (PDAs ). While only two user devices HOA and HOB are illustrated in FIG. 1, the environ-ment 100 may include thousands or millions of such devices. The web browsers may be software applications running on mobile devices 110 for retrieving web content from the origin server 160 and presenting the web content on a display coupled to the mobile device. Web content accessed by the user devices 110 include text, images, audio and video con-tent. The multimedia content can be played back by the browsers, for example, HTML5 compatible browsers, plug-in or a standalone media player. The browsers can also invoke the media players or plug-ins available on the user |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 184 of 1100

| No. | '111 Patent Claim 1 | Swenson |
|---|---|---|
| | | devices 110 and passes images, audio and/or video to the media player or plug-in for playback.")<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0028] ("In contrast to conventional inline TCP throughput monitoring devices that monitor every single data packets transmitted and received, the network controller 140 is an "out-of-band" computer server that interfaces with the steer-ing device 130 to selectively inspect user flows of interest. The network controller 140 may further identify user flows (e.g., among the flows of interest) for optimization. In one embodiment, the network controller 140 may be imple-mented at the steering device 130 to monitor traffic. In other embodiments, the network controller 140 is coupled to and communicates with the steering device 130 for traffic moni-toring and optimization. When queried by the steering device 130, the network controller 140 determines if a given network flow should be ignored, monitored further or optimized. Opti-mization of a flow is often decided at the beginning of the flow because it is rarely possible to switch to optimized content mid-stream once non-optimized content delivery has begun. However, the network controller 140 may determine that existing flows associated with a particular subscriber or other entity should be optimized. In turn, new flows ( e.g., resulting from seek requests in media, new media requests, resume after pause, etc.) determined to be associated with the entity may be optimized. The network controller 140 uses the net-work state as well as historical traffic data in its decision for monitoring and optimization. Knowledge on the current net-work state, such as congestion, deems critical when it comes to data optimization.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 185 of 1100

| No. | '111 Patent Claim 1 | Swenson |
|---|---|---|
| | | Swenson at [0058] ("Referring now to FIG. 4A, network traffic flows from the user device 110 through the steering device 130 and arrive at the origin server 160 over the network request path. For example, a browser on the user device 110 may request web content from the origin server 160. A HTTP request message initiated at the user device 110 is forwarded to the steering device 130 over the network link 411. A data switch 402 inside the steering device 130 then relays the request message to the origin server 160 over the network link 412. On the opposite direction, network traffic originated from the origin server 160 flows through the steering device 130 back to the user device 110 over the network response path. For example, the origin server 160 responds to the user request by sending web content over the network link 413 to the steering device 130, which forwards the web content to the user device 110 over the network link 416. Note that the network links 411 and 416 are two opposite directions on the same physical link, so are the network link pair 414 and 415. On the other hand, the network link pair 412 and 413 may or may not share the same network path because traffic between the steering device 130 and origin server 160 on opposite directions may be routed differently over one or more routers.") |
| 1[d] | checking, by the network node, if the packet satisfies the criterion; | Swenson discloses checking, by the network node, if the packet satisfies the criterion.<br><br>For example, Swenson discloses determining by the steering device packet flows that match one or more signatures, conditions, or criteria of the packet.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0028] ("In contrast to conventional inline TCP throughput monitoring devices that monitor every single data packets transmitted and received, the network controller 140 is an "out-of-band" computer server that interfaces with the steer-ing device 130 to |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 186 of 1100

| No. | '111 Patent Claim 1 | Swenson |
|---|---|---|
| | | selectively inspect user flows of interest. The network controller 140 may further identify user flows (e.g., among the flows of interest) for optimization. In one embodiment, the network controller 140 may be imple-mented at the steering device 130 to monitor traffic. In other embodiments, the network controller 140 is coupled to and communicates with the steering device 130 for traffic moni-toring and optimization. When queried by the steering device 130, the network controller 140 determines if a given network flow should be ignored, monitored further or optimized. Opti-mization of a flow is often decided at the beginning of the flow because it is rarely possible to switch to optimized content mid-stream once non-optimized content delivery has begun. However, the network controller 140 may determine that existing flows associated with a particular subscriber or other entity should be optimized. In turn, new flows ( e.g., resulting from seek requests in media, new media requests, resume after pause, etc.) determined to be associated with the entity may be optimized. The network controller 140 uses the net-work state as well as historical traffic data in its decision for monitoring and optimization. Knowledge on the current net-work state, such as congestion, deems critical when it comes to data optimization.")<br><br>Swenson at [0045] ("The steering device interface 316 interacts with an external routing appliance, such as the steering device 130 to divert portions of the network traffic ( e.g., large object net-work flows). Existing routing appliances in most carrier net-works are designed to handle large amounts of network traf-fic. They are not, however, ideal devices to operate for monitoring and analysis individual flows. Through the steer-ing device interface 316, the network controller 140 may communicate with the external routing appliances, such as the steering device 130, to steer a portion of network traffic to the network controller 140 when certain conditions are met. Generally, network flows of interest to the network controller 140 contain larger media objects, such as videos and images. In one embodiment, the smaller flows, such as web page and text information, are not exchanged over the steering device interface 316.")<br><br>Swenson at [0059] ("In one embodiment, as the steering device 130 monitors network responses, it is looking for flows that match one or more signatures for video and images. When a match-ing flow is detected, the steering device 130 forwards the HTTP request and a portion of the HTTP response to the network controller 140 over the ICAP client interface 404. After receiving the request and the portion of response at the ICAP server interface |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 187 of 1100

| No. | '111 Patent Claim 1 | Swenson |
|---|---|---|
| | | 406, the flow analyzer 312 of the net-work controller 140 performs a deep flow inspection to deter-mine if the flow is worth bandwidth monitoring and/or user detection. For example, the flow inspection performed by the flow analyzer 312 may determine if the flow indeed contains large or medium object ( e.g., larger than 50 kB), and/or if the source IP address of the flow is from a user or a group of users that are required to be monitored by policies. The flow ana-lyzer 312 may also determine if the flow needs to be opti-mized based on historical flow statistical data.")<br><br>Swenson at [0060] ("If the flow is deemed of interest, the steering device 130 is notified to steer the flow through the network controller 140. This is known as the "continue" working mode for bandwidth monitoring. In the "continue" mode, the network controller 140 interfaces with the steering device 130 to func-tion, on-demand, as a traditional inline network element for flows deemed of interest. Thus, the network controller 140 ingests the network flow for inspection and subsequently forwards the network flow on the network response path. For example, for this particular flow, the origin server 160 responds to the user request by sending video or images over the network link 413 to the steering device 130, which for-wards the video or images to the network controller 140 over a network link 414. After the network controller 140 updates the flow statistics, the video or images are returned to the steering device 130 over a network link 415, which transmits the video or images to the user device 110 over the network link 416.")<br><br>Swenson at [0065] ("FIG. 5 is a block diagram illustrating an example event trace of "continue" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 512 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 512 to the intended origin server 160 and receives a response 514 back from the origin server 160. The steering device 130 then sends an ICAP request message 516 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 518. Upon receiving the instruc-tion, the steering device 130 re-writes the response 514 to an HTTP redirect response 520, causing the user device 110 to request the video file |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 188 of 1100

| No. | '111 Patent Claim 1 | Swenson |
|---|---|---|
| | | from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 520 directly to the user device 110. In case the flow dose not contain video or image objects, or the network controller 140 determines not to monitor the flow, the steering device 13 0 would forward the response to the user device 110.")<br><br>Swenson at [0069] ("FIG. 6 is a block diagram illustrating an example event trace of "counting" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 612 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 612 to the intended origin server 160 and receives a response 614 back from the origin server 160. The steering device 130 then sends an ICAP request message 616 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 618. Upon receiving the instruc-tion, the steering device 130 re-writes the response 614 to an HTTP redirect response 620, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 620 directly to the user device 110. In case the flow dose not contain video or image objects that need to be redirected, the steering device 130 would forward the response to the user device 110.")<br><br>Swenson at [0079] ("In the bandwidth calculation, flows are categorized into buckets based on the size of the objects being transferred. Small objects may not be factored into the bandwidth calcu-lation since they may come and go within a single interval. For example, flows with payload size less than 50 kB may be ignored because a transfer of 50 kB may never reach the full potential throughput of the link. While larger flows may reach the full throughput of the link for a long period of time intervals, they are grouped into 50-75 kB, 75-100 kB and 100 kB+ buckets because the characteristics of these flow sizes can be different, hence the bandwidth for each of the buckets is measured and calculated separately. In other embodiments, the flow size ranges (e.g., 50-75 kB, 75-100 kB and l00kB+) of the buckets may be altered depending on the network traffic and size of objects transmitted. Furthermore, the bucket sizes can also be adjusted based on network topology, |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 189 of 1100

| No. | '111 Patent Claim 1 | Swenson |
|---|---|---|
| | | such as buffer size, prior to transmission to the client. The calculated bandwidth per bucket is stored in a queue structure that allows for the computing and updating of minimum, maximum, and/or average measurements for each bucket. In one embodiment, the 100 kB+ bucket's current tail entry is checked against the average bandwidth for the 100 kB+ bucket. If the current entry is less than the average multiplied by the number of entries in the queue, the current entry is added to the bandwidth calculation for the current interval. This scheme can filter out large bursts of data from tempo-rarily idle flows. If the bandwidth exceeds the value, a number of bytes (e.g., 125 kB) will be subtracted from the current entry to account for TCP buffers in the network.") |
| 1[e] | responsive to the packet not satisfying the criterion, sending, by the network node over the packet network, the packet to the second entity; and | Swenson discloses responsive to the packet not satisfying the criterion, sending, by the network node over the packet network, the packet to the second entity.<br><br>For example, Swenson discloses monitoring and categorizing network traffic by the steering device based on instructions and desired criteria sent by the network controller to determine if packet flows require further inspection. Based on the instruction and desired criteria, the network controller monitors and optimizes only a subset of network traffic. Packet flows that do not meet the desired criteria from the network controller's instructions at the steering device are not sent for further inspection and are sent to their originally intended destination.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0028] ("In contrast to conventional inline TCP throughput monitoring devices that monitor every single data packets transmitted and received, the network controller 140 |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 190 of 1100

| No. | ’111 Patent Claim 1 | Swenson |
|---|---|---|
| | | is an "out-of-band" computer server that interfaces with the steer-ing device 130 to selectively inspect user flows of interest. The network controller 140 may further identify user flows (e.g., among the flows of interest) for optimization. In one embodiment, the network controller 140 may be imple-mented at the steering device 130 to monitor traffic. In other embodiments, the network controller 140 is coupled to and communicates with the steering device 130 for traffic moni-toring and optimization. When queried by the steering device 130, the network controller 140 determines if a given network flow should be ignored, monitored further or optimized. Opti-mization of a flow is often decided at the beginning of the flow because it is rarely possible to switch to optimized content mid-stream once non-optimized content delivery has begun. However, the network controller 140 may determine that existing flows associated with a particular subscriber or other entity should be optimized. In turn, new flows ( e.g., resulting from seek requests in media, new media requests, resume after pause, etc.) determined to be associated with the entity may be optimized. The network controller 140 uses the net-work state as well as historical traffic data in its decision for monitoring and optimization. Knowledge on the current net-work state, such as congestion, deems critical when it comes to data optimization.”)<br><br>Swenson at [0038] (“Turning back to FIG. 1, the network controller 140 allows network operators to apply fine granular optimization policies to ensure high quality of experience (QoE) based on cell tower congestion, device types, subscriber profiles and service plans with lower hardware and software costs. The architecture of the network controller 140 provides an excel-lent fit for the net neutrality guideline of"reasonable network management", and better compliance to the copyright law (DMCA) than solutions that rely on long-term caching. Hav-ing the ability of monitoring network traffic on a per sub-scriber, per flow, or per video file basis, the network controller 140 also selectively monitors and optimizes only a subset of traffic that benefits from optimization the most, thus achiev-ing both scalability and efficiency for optimization at a com-petitive price-point. The core element of the network control-ler 140 lies in its mechanisms for congestion detection and mitigation, which allows optimization resources to be utilized in the most efficient and surgical manner.”)<br><br>Swenson at [0042] (“The network controller 140 collects real-time statis-tical data on the network flows from core network side with-out probes deployed in the RAN network. The statistical data is stored and compared against historical flow data to estimate level of |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 191 of 1100

| No. | '111 Patent Claim 1 | Swenson |
|---|---|---|
| | | congestion and available network bandwidth. Instead of collecting traffic statistics for every flow and every session, the network controller 140 samples only large flows involving media objects such as videos and images above a certain size ( e.g., above 50 kB). The network controller 140 can choose to be a pass-through device to monitor the large flows as well as to determine whether to optimize the flows. Measuring only larger flows has the advantage to mitigate corruptions caused by origin server latency and network glitches. Furthermore, focusing on the large flows helps the network controller to reduce the background noise and to increase noise-to-signal ratio in bandwidth measuring by removing the impact of millions of tiny or small flows with delivery time in millisec-onds. Therefore the reliability of bandwidth estimation and congestion detection is much higher.")<br><br>Swenson at [0045] ("The steering device interface 316 interacts with an external routing appliance, such as the steering device 130 to divert portions of the network traffic ( e.g., large object net-work flows). Existing routing appliances in most carrier net-works are designed to handle large amounts of network traf-fic. They are not, however, ideal devices to operate for monitoring and analysis individual flows. Through the steer-ing device interface 316, the network controller 140 may communicate with the external routing appliances, such as the steering device 130, to steer a portion of network traffic to the network controller 140 when certain conditions are met. Generally, network flows of interest to the network controller 140 contain larger media objects, such as videos and images. In one embodiment, the smaller flows, such as web page and text information, are not exchanged over the steering device interface 316.")<br><br>Swenson at [0059] ("In one embodiment, as the steering device 130 monitors network responses, it is looking for flows that match one or more signatures for video and images. When a match-ing flow is detected, the steering device 130 forwards the HTTP request and a portion of the HTTP response to the network controller 140 over the ICAP client interface 404. After receiving the request and the portion of response at the ICAP server interface 406, the flow analyzer 312 of the net-work controller 140 performs a deep flow inspection to deter-mine if the flow is worth bandwidth monitoring and/or user detection. For example, the flow inspection performed by the flow analyzer 312 may determine if the flow indeed contains large or medium object ( e.g., larger than 50 kB), and/or if the source IP address of the flow is from a user or a group of users that are required to be monitored by policies. The |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 192 of 1100

| No. | '111 Patent Claim 1 | Swenson |
|---|---|---|
| | | flow ana-lyzer 312 may also determine if the flow needs to be opti-mized based on historical flow statistical data.")<br><br>Swenson at [0060] ("If the flow is deemed of interest, the steering device 130 is notified to steer the flow through the network controller 140. This is known as the "continue" working mode for bandwidth monitoring. In the "continue" mode, the network controller 140 interfaces with the steering device 130 to func-tion, on-demand, as a traditional inline network element for flows deemed of interest. Thus, the network controller 140 ingests the network flow for inspection and subsequently forwards the network flow on the network response path. For example, for this particular flow, the origin server 160 responds to the user request by sending video or images over the network link 413 to the steering device 130, which for-wards the video or images to the network controller 140 over a network link 414. After the network controller 140 updates the flow statistics, the video or images are returned to the steering device 130 over a network link 415, which transmits the video or images to the user device 110 over the network link 416.")<br><br>Swenson at [0065] ("FIG. 5 is a block diagram illustrating an example event trace of "continue" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 512 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 512 to the intended origin server 160 and receives a response 514 back from the origin server 160. The steering device 130 then sends an ICAP request message 516 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 518. Upon receiving the instruc-tion, the steering device 130 re-writes the response 514 to an HTTP redirect response 520, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 520 directly to the user device 110. In case the flow dose not contain video or image objects, or the network controller 140 determines not to monitor the flow, the steering device 13 0 would forward the response to the user device 110.") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 193 of 1100

| No. | '111 Patent Claim 1 | Swenson |
|---|---|---|
| | | Swenson at [0069] ("FIG. 6 is a block diagram illustrating an example event trace of "counting" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 612 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 612 to the intended origin server 160 and receives a response 614 back from the origin server 160. The steering device 130 then sends an ICAP request message 616 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 618. Upon receiving the instruc-tion, the steering device 130 re-writes the response 614 to an HTTP redirect response 620, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 620 directly to the user device 110. In case the flow dose not contain video or image objects that need to be redirected, the steering device 130 would forward the response to the user device 110.") |
| 1[f] | responsive to the packet satisfying the criterion, sending the packet, by the network node over the packet network, to an entity that is included in the instruction and is other than the second entity. | Swenson discloses responsive to the packet satisfying the criterion, sending the packet, by the network node over the packet network, to an entity that is included in the instruction and is other than the second entity.<br><br>For example, Swenson discloses determining by the steering device monitors flows that match one or more signatures or criteria of the packet. Swenson further discloses that when a matching flow is detected the steering device forwards the packet to the network controller.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 194 of 1100

| No. | '111 Patent Claim 1 | Swenson |
|---|---|---|
| | | network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0028] ("In contrast to conventional inline TCP throughput monitoring devices that monitor every single data packets transmitted and received, the network controller 140 is an "out-of-band" computer server that interfaces with the steer-ing device 130 to selectively inspect user flows of interest. The network controller 140 may further identify user flows (e.g., among the flows of interest) for optimization. In one embodiment, the network controller 140 may be imple-mented at the steering device 130 to monitor traffic. In other embodiments, the network controller 140 is coupled to and communicates with the steering device 130 for traffic moni-toring and optimization. When queried by the steering device 130, the network controller 140 determines if a given network flow should be ignored, monitored further or optimized. Opti-mization of a flow is often decided at the beginning of the flow because it is rarely possible to switch to optimized content mid-stream once non-optimized content delivery has begun. However, the network controller 140 may determine that existing flows associated with a particular subscriber or other entity should be optimized. In turn, new flows ( e.g., resulting from seek requests in media, new media requests, resume after pause, etc.) determined to be associated with the entity may be optimized. The network controller 140 uses the net-work state as well as historical traffic data in its decision for monitoring and optimization. Knowledge on the current net-work state, such as congestion, deems critical when it comes to data optimization.")<br><br>Swenson at [0029] ("As a flow is sent to the network controller 140 for inspection, historical network traffic data stored at the net-work controller 140 may be searched. The historical network traffic data includes information such as subscriber informa-tion, the cell towers to which the user devices attached, rout-ers through which the traffic is passing, geography regions, the backhaul segments, and time-of-day of the flows. For example, in a mobile network, the cell tower to which a user device is attached can be most useful, since it is the location where most congestion occurs due to limited bandwidth and high cost of the radio access network infrastructure. The network controller 140 looks into the historical traffic data for the average of the bandwidth per user at the particular cell tower. The network controller 140 can then estimate the amount ofbandwidth or degree of congestion for the new flow based on the historical record.") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 195 of 1100

| No. | '111 Patent Claim 1 | Swenson |
|---|---|---|
| | | Swenson at [0038] ("Turning back to FIG. 1, the network controller 140 allows network operators to apply fine granular optimization policies to ensure high quality of experience (QoE) based on cell tower congestion, device types, subscriber profiles and service plans with lower hardware and software costs. The architecture of the network controller 140 provides an excel-lent fit for the net neutrality guideline of"reasonable network management", and better compliance to the copyright law (DMCA) than solutions that rely on long-term caching. Hav-ing the ability of monitoring network traffic on a per sub-scriber, per flow, or per video file basis, the network controller 140 also selectively monitors and optimizes only a subset of traffic that benefits from optimization the most, thus achiev-ing both scalability and efficiency for optimization at a com-petitive price-point. The core element of the network control-ler 140 lies in its mechanisms for congestion detection and mitigation, which allows optimization resources to be utilized in the most efficient and surgical manner.")<br><br>Swenson at [0039] ("Referring now to FIG. 3, it illustrates one embodi-ment of an example architecture of the network controller 140 for providing selective real-time network monitoring and subscriber identification. The network controller 140 com-prises a flow analyzer 312, a policy engine 314, a steering device interface 316, a video optimizer redirector 318, a flow cache 322, and a subscriber log 324. In other embodiments, the network controller 140 may include additional, fewer, or different components for various applications. Conventional components such as network interfaces, security functions, failover servers, management and network operations con-soles, and the like are not shown so as to not obscure the details of the system architecture.")<br><br>Swenson at [0045] ("The steering device interface 316 interacts with an external routing appliance, such as the steering device 130 to divert portions of the network traffic ( e.g., large object net-work flows). Existing routing appliances in most carrier net-works are designed to handle large amounts of network traf-fic. They are not, however, ideal devices to operate for monitoring and analysis individual flows. Through the steer-ing device interface 316, the network controller 140 may communicate with the external routing appliances, such as the steering device 130, to steer a portion of network traffic to the network controller 140 when certain conditions are met. Generally, network flows of interest to the network controller 140 contain larger media objects, such as videos and |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 196 of 1100

| No. | '111 Patent Claim 1 | Swenson |
|---|---|---|
| | | images. In one embodiment, the smaller flows, such as web page and text information, are not exchanged over the steering device interface 316.")<br><br>Swenson at [0059] ("In one embodiment, as the steering device 130 monitors network responses, it is looking for flows that match one or more signatures for video and images. When a match-ing flow is detected, the steering device 130 forwards the HTTP request and a portion of the HTTP response to the network controller 140 over the ICAP client interface 404. After receiving the request and the portion of response at the ICAP server interface 406, the flow analyzer 312 of the net-work controller 140 performs a deep flow inspection to deter-mine if the flow is worth bandwidth monitoring and/or user detection. For example, the flow inspection performed by the flow analyzer 312 may determine if the flow indeed contains large or medium object ( e.g., larger than 50 kB), and/or if the source IP address of the flow is from a user or a group of users that are required to be monitored by policies. The flow ana-lyzer 312 may also determine if the flow needs to be opti-mized based on historical flow statistical data.")<br><br>Swenson at [0060] ("If the flow is deemed of interest, the steering device 130 is notified to steer the flow through the network controller 140. This is known as the "continue" working mode for bandwidth monitoring. In the "continue" mode, the network controller 140 interfaces with the steering device 130 to func-tion, on-demand, as a traditional inline network element for flows deemed of interest. Thus, the network controller 140 ingests the network flow for inspection and subsequently forwards the network flow on the network response path. For example, for this particular flow, the origin server 160 responds to the user request by sending video or images over the network link 413 to the steering device 130, which for-wards the video or images to the network controller 140 over a network link 414. After the network controller 140 updates the flow statistics, the video or images are returned to the steering device 130 over a network link 415, which transmits the video or images to the user device 110 over the network link 416.")<br><br>Swenson at [0065] ("FIG. 5 is a block diagram illustrating an example event trace of "continue" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 512 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 197 of 1100

| No. | '111 Patent Claim 1 | Swenson |
|---|---|---|
| | | one embodiment, the steering device 130 for-wards the HTTP get request 512 to the intended origin server 160 and receives a response 514 back from the origin server 160. The steering device 130 then sends an ICAP request message 516 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 518. Upon receiving the instruc-tion, the steering device 130 re-writes the response 514 to an HTTP redirect response 520, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 520 directly to the user device 110. In case the flow dose not contain video or image objects, or the network controller 140 determines not to monitor the flow, the steering device 13 0 would forward the response to the user device 110.")<br><br>Swenson at [0069] ("FIG. 6 is a block diagram illustrating an example event trace of "counting" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 612 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 612 to the intended origin server 160 and receives a response 614 back from the origin server 160. The steering device 130 then sends an ICAP request message 616 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 618. Upon receiving the instruc-tion, the steering device 130 re-writes the response 614 to an HTTP redirect response 620, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 620 directly to the user device 110. In case the flow dose not contain video or image objects that need to be redirected, the steering device 130 would forward the response to the user device 110.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 198 of 1100

| No. | '111 Patent Claim 2 | Swenson |
|---|---|---|
| 2[a] | The method according to claim 1, wherein the instruction is 'probe', 'mirror', or 'terminate' instruction, and | Swenson discloses the method according to claim 1, wherein the instruction is 'probe', 'mirror', or 'terminate' instruction.<br><br>For example, Swenson discloses instructions sent by the network controller to the steering device instructions as to whether a packet flow should be ignored, monitored, or optimized. The monitoring and optimizing instructions may comprise forwarding packets/messages user device/origin server and/or the network controller.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0028] ("In contrast to conventional inline TCP throughput monitoring devices that monitor every single data packets transmitted and received, the network controller 140 is an "out-of-band" computer server that interfaces with the steer-ing device 130 to selectively inspect user flows of interest. The network controller 140 may further identify user flows (e.g., among the flows of interest) for optimization. In one embodiment, the network controller 140 may be imple-mented at the steering device 130 to monitor traffic. In other embodiments, the network controller 140 is coupled to and communicates with the steering device 130 for traffic moni-toring and optimization. When queried by the steering device 130, the network controller 140 determines if a given network flow should be ignored, monitored further or optimized. Opti-mization of a flow is often decided at the beginning of the flow because it is rarely possible to switch to optimized content mid-stream once non-optimized content delivery has begun. However, the network controller 140 may determine that existing flows associated with a particular subscriber or other entity should be optimized. In turn, new flows ( e.g., resulting from seek requests in media, new media requests, resume after pause, etc.) determined to be associated with the entity may be |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 199 of 1100

| No. | '111 Patent Claim 2 | Swenson |
|---|---|---|
| | | optimized. The network controller 140 uses the net-work state as well as historical traffic data in its decision for monitoring and optimization. Knowledge on the current net-work state, such as congestion, deems critical when it comes to data optimization.")<br><br>Swenson at [0029] ("As a flow is sent to the network controller 140 for inspection, historical network traffic data stored at the net-work controller 140 may be searched. The historical network traffic data includes information such as subscriber informa-tion, the cell towers to which the user devices attached, rout-ers through which the traffic is passing, geography regions, the backhaul segments, and time-of-day of the flows. For example, in a mobile network, the cell tower to which a user device is attached can be most useful, since it is the location where most congestion occurs due to limited bandwidth and high cost of the radio access network infrastructure. The network controller 140 looks into the historical traffic data for the average of the bandwidth per user at the particular cell tower. The network controller 140 can then estimate the amount of bandwidth or degree of congestion for the new flow based on the historical record.")<br><br>Swenson at [0060] ("If the flow is deemed of interest, the steering device 130 is notified to steer the flow through the network controller 140. This is known as the "continue" working mode for bandwidth monitoring. In the "continue" mode, the network controller 140 interfaces with the steering device 130 to func-tion, on-demand, as a traditional inline network element for flows deemed of interest. Thus, the network controller 140 ingests the network flow for inspection and subsequently forwards the network flow on the network response path. For example, for this particular flow, the origin server 160 responds to the user request by sending video or images over the network link 413 to the steering device 130, which for-wards the video or images to the network controller 140 over a network link 414. After the network controller 140 updates the flow statistics, the video or images are returned to the steering device 130 over a network link 415, which transmits the video or images to the user device 110 over the network link 416.")<br><br>Swenson at [0061] ("Once a flow is reported to the network controller 140, a flow cache entry is created for the flow in the flow cache 322. The flow cache entry keeps track of the flow and its associated bandwidth. For a flow that is marked in "continue" mode, each time the steering device 130 forwards a next portion of the flow payload to the network controller 140, the flow cache 3 22 updates the number of bytes for transmitted in the flow. By |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 200 of 1100

| No. | '111 Patent Claim 2 | Swenson |
|---|---|---|
| | | monitoring the number of bytes per flow over time, the flow analyzer 312 is capable of determining an estimate value of bandwidth associated with flow. Further-more, since the steering device 130 does not have infinite packet buffers, if congestion happens on the network link 416 from the steering device 130 to the user device 110, the TCP congestion control mechanism kicks in at the steering device 130, which may slows down and/or eventually stop receiving data over the network link 413 from origin server 160. During the congestion, the steering device 130 would not forward any data to the network controller 140, since the link 416 is congested and the network controller 140 would not be able to transmit data to the user device 110. Therefore, as an inline element, the network controller 140 can detect network con-gestions and estimate bandwidth associated with any flows of interest selected by the network controller 140. However, in the "continue" mode, the network controller 140 does not modify and transform the HTTP messaged it receives over the ICAP interface. The network controller 140 simply updates the flow statistics and returns the video or images to the steering device 130 for transmission to the user device 110.")<br><br>Swenson at [0064] ("Similar to the "continue" mode, after receiving the initial HTTP messages of a flow and determining to monitor the flow, the network controller 140 notify the steering device 130 to work in a "counting" mode for bandwidth monitoring. In contrast to the "continue" mode, when a matching flow is detected for "counting" mode, the steering device 130 for-wards the HTTP response directly to the user device 110. While at the same time, the steering device 130 send a cus-tomized ICAP message to the network controller 140 over the network link 425. In one embodiment, the customized ICAP message contains the HTTP request and response headers, as well as a count of payload size of the current flow. After updating the flow statistics, the network controller 140 may acknowledge the gateway over the network line 426. In the "counting" mode, the network controller 140 does not join the network response path as an inline network element, but simply listens to the counting of flow size. The benefit of the "counting" mode is to off-load the network controller 140 from ingesting and forwarding the network flow on the net-work response path, while still enabling the detection of con-gestions and estimation of bandwidth associated with the flows of interest.")<br><br>Swenson at [0065] ("FIG. 5 is a block diagram illustrating an example event trace of "continue" working mode between the user device 110, steering device 130, network |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 201 of 1100

| No. | '111 Patent Claim 2 | Swenson |
|---|---|---|
| | | controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 512 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 512 to the intended origin server 160 and receives a response 514 back from the origin server 160. The steering device 130 then sends an ICAP request message 516 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 518. Upon receiving the instruc-tion, the steering device 130 re-writes the response 514 to an HTTP redirect response 520, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 520 directly to the user device 110. In case the flow dose not contain video or image objects, or the network controller 140 determines not to monitor the flow, the steering device 13 0 would forward the response to the user device 110.")<br><br>Swenson at [0069] ("FIG. 6 is a block diagram illustrating an example event trace of "counting" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 612 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 612 to the intended origin server 160 and receives a response 614 back from the origin server 160. The steering device 130 then sends an ICAP request message 616 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 618. Upon receiving the instruc-tion, the steering device 130 re-writes the response 614 to an HTTP redirect response 620, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 620 directly to the user device 110. In case the flow dose not contain video or image objects that need to be redirected, the steering device 130 would forward the response to the user device 110.") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 202 of 1100

| No. | '111 Patent Claim 2 | Swenson |
|---|---|---|
| 2[b] | upon receiving by the network node the 'terminate' instruction, the method further comprising blocking, by the network node, the packet from being sent to the second entity and to the controller. | Swenson discloses upon receiving by the network node the 'terminate' instruction, the method further comprising blocking, by the network node, the packet from being sent to the second entity and to the controller.<br><br>For example, Swenson discloses in response to detected traffic congestion, the steering device stops receiving data from the user device or origin server and does not forward any data to the network controller.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0028] ("In contrast to conventional inline TCP throughput monitoring devices that monitor every single data packets transmitted and received, the network controller 140 is an "out-of-band" computer server that interfaces with the steer-ing device 130 to selectively inspect user flows of interest. The network controller 140 may further identify user flows (e.g., among the flows of interest) for optimization. In one embodiment, the network controller 140 may be imple-mented at the steering device 130 to monitor traffic. In other embodiments, the network controller 140 is coupled to and communicates with the steering device 130 for traffic moni-toring and optimization. When queried by the steering device 130, the network controller 140 determines if a given network flow should be ignored, monitored further or optimized. Opti-mization of a flow is often decided at the beginning of the flow because it is rarely possible to switch to optimized content mid-stream once non-optimized content delivery has begun. However, the network controller 140 may determine that existing flows associated with a particular subscriber or other entity should be optimized. In turn, new flows ( e.g., resulting from seek requests in media, new media requests, resume after pause, etc.) determined to be associated with the entity may be |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 203 of 1100

| No. | '111 Patent Claim 2 | Swenson |
|---|---|---|
| | | optimized. The network controller 140 uses the net-work state as well as historical traffic data in its decision for monitoring and optimization. Knowledge on the current net-work state, such as congestion, deems critical when it comes to data optimization.")<br><br>Swenson at [0061] ("Once a flow is reported to the network controller 140, a flow cache entry is created for the flow in the flow cache 322. The flow cache entry keeps track of the flow and its associated bandwidth. For a flow that is marked in "continue" mode, each time the steering device 130 forwards a next portion of the flow payload to the network controller 140, the flow cache 3 22 updates the number of bytes for transmitted in the flow. By monitoring the number of bytes per flow over time, the flow analyzer 312 is capable of determining an estimate value of bandwidth associated with flow. Further-more, since the steering device 130 does not have infinite packet buffers, if congestion happens on the network link 416 from the steering device 130 to the user device 110, the TCP congestion control mechanism kicks in at the steering device 130, which may slows down and/or eventually stop receiving data over the network link 413 from origin server 160. During the congestion, the steering device 130 would not forward any data to the network controller 140, since the link 416 is congested and the network controller 140 would not be able to transmit data to the user device 110. Therefore, as an inline element, the network controller 140 can detect network con-gestions and estimate bandwidth associated with any flows of interest selected by the network controller 140. However, in the "continue" mode, the network controller 140 does not modify and transform the HTTP messaged it receives over the ICAP interface. The network controller 140 simply updates the flow statistics and returns the video or images to the steering device 130 for transmission to the user device 110.") |

| No. | '111 Patent Claim 3 | Swenson |
|---|---|---|
| 3[a] | The method according to claim 1, wherein the instruction is a 'probe', a 'mirror', or a 'terminate' instruction, and | Swenson discloses the method according to claim 1, wherein the instruction is a 'probe', a 'mirror', or a 'terminate' instruction.<br><br>*See supra* at 2(a). |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 204 of 1100

| No. | '111 Patent Claim 3 | Swenson |
|---|---|---|
| 3[b] | upon receiving by the network node the 'mirror' instruction and responsive to the packet satisfying the criterion, the method further comprising sending the packet, by the network node, to the second entity and to the controller. | Swenson discloses upon receiving by the network node the 'mirror' instruction and responsive to the packet satisfying the criterion, the method further comprising sending the packet, by the network node, to the second entity and to the controller.<br><br>For example, Swenson discloses a counting mode instructed by the network controller to the steering device for monitoring and optimizing, in which the steering device forwards the packet flow to the user device/origin server and at the same time, sending the packet flow to the network controller.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0028] ("In contrast to conventional inline TCP throughput monitoring devices that monitor every single data packets transmitted and received, the network controller 140 is an "out-of-band" computer server that interfaces with the steer-ing device 130 to selectively inspect user flows of interest. The network controller 140 may further identify user flows (e.g., among the flows of interest) for optimization. In one embodiment, the network controller 140 may be imple-mented at the steering device 130 to monitor traffic. In other embodiments, the network controller 140 is coupled to and communicates with the steering device 130 for traffic moni-toring and optimization. When queried by the steering device 130, the network controller 140 determines if a given network flow should be ignored, monitored further or optimized. Opti-mization of a flow is often decided at the beginning of the flow because it is rarely possible to switch to optimized content mid-stream once non-optimized content delivery has begun. However, the network controller 140 may determine that existing flows associated with a particular subscriber or other entity should be optimized. In turn, new flows ( e.g., resulting from seek requests in media, new media |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 205 of 1100

| No. | '111 Patent Claim 3 | Swenson |
|---|---|---|
| | | requests, resume after pause, etc.) determined to be associated with the entity may be optimized. The network controller 140 uses the net-work state as well as historical traffic data in its decision for monitoring and optimization. Knowledge on the current net-work state, such as congestion, deems critical when it comes to data optimization.")<br><br>Swenson at [0059] ("In one embodiment, as the steering device 130 monitors network responses, it is looking for flows that match one or more signatures for video and images. When a match-ing flow is detected, the steering device 130 forwards the HTTP request and a portion of the HTTP response to the network controller 140 over the ICAP client interface 404. After receiving the request and the portion of response at the ICAP server interface 406, the flow analyzer 312 of the net-work controller 140 performs a deep flow inspection to deter-mine if the flow is worth bandwidth monitoring and/or user detection. For example, the flow inspection performed by the flow analyzer 312 may determine if the flow indeed contains large or medium object ( e.g., larger than 50 kB), and/or if the source IP address of the flow is from a user or a group of users that are required to be monitored by policies. The flow ana-lyzer 312 may also determine if the flow needs to be opti-mized based on historical flow statistical data.")<br><br>Swenson at [0064] ("Similar to the "continue" mode, after receiving the initial HTTP messages of a flow and determining to monitor the flow, the network controller 140 notify the steering device 130 to work in a "counting" mode for bandwidth monitoring. In contrast to the "continue" mode, when a matching flow is detected for "counting" mode, the steering device 130 for-wards the HTTP response directly to the user device 110. While at the same time, the steering device 130 send a cus-tomized ICAP message to the network controller 140 over the network link 425. In one embodiment, the customized ICAP message contains the HTTP request and response headers, as well as a count of payload size of the current flow. After updating the flow statistics, the network controller 140 may acknowledge the gateway over the network line 426. In the "counting" mode, the network controller 140 does not join the network response path as an inline network element, but simply listens to the counting of flow size. The benefit of the "counting" mode is to off-load the network controller 140 from ingesting and forwarding the network flow on the net-work response path, while still enabling the detection of con-gestions and estimation of bandwidth associated with the flows of interest.") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 206 of 1100

| No. | '111 Patent Claim 3 | Swenson |
|---|---|---|
| | | Swenson at [0071] ("After receiving the request, the video optimizer 150 forwards the video HTTP GET requests 622 to the origin server 160 and in return, receives a video file 624 from the origin server 160. The video optimizer 150 transcodes the video file to a format usable by the client device 110 based on network bandwidth available to the user device 110. The optimized video 626 is then transmitted from the video opti-mizer 150 to the steering device 130. In one embodiment, the steering device 130 intercepts the optimized video 626. The steering device 130 will then send an ICAP request to the network controller 140 for inspection. The network controller 140 deems this flow to be monitored and sends ICAP response 630. The steering device 130 then allows the flow to go through to the user device 110. The steering device 130 next sends periodic ICAP "counting" updates 632 to the network controller 140 until the flow completes. As such, the client receives the optimized video 626 for substantially real-time playback on an application executing on the user device 110.")<br>Swenson at [0072] ("In one embodiment, if the video optimizer 150 failed to retrieve user requested video file from the origin server 160, the video optimizer 150 appends a "do not transcode" flag to the HTTP redirect request and returned to the user device 110, which re-sends the request out over the network to the origin server 160. The origin server 160 responds appropriately to the request by sending back video 624, which is intercepted by the steering device 130 only. The steering device 130 forwards the video to the user device 110 and at the same time reports the flow size to the network controller 140 for monitoring purpose.") |

| No. | '111 Patent Claim 4 | Swenson |
|---|---|---|
| 4[a] | The method according to claim 1, wherein the instruction is 'probe', 'mirror', or 'terminate' instruction, and | Swenson discloses the method according to claim 1, wherein the instruction is 'probe', 'mirror', or 'terminate' instruction.<br><br>*See supra* at 2(a). |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 207 of 1100

| No. | '111 Patent Claim 4 | Swenson |
|---|---|---|
| 4[b] | upon receiving by the network node the 'probe' instruction and responsive to the packet satisfying the criterion, the method further comprising: sending the packet, by the network node, to the controller; | Swenson discloses upon receiving by the network node the 'probe' instruction and responsive to the packet satisfying the criterion, the method further comprising: sending the packet, by the network node, to the controller.<br><br>For example, Swenson discloses determining by the steering device flows that match one or more signatures or criteria of the packet. Swenson further discloses that when a matching flow is detected the steering device forwards the packet to the network controller.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0028] ("In contrast to conventional inline TCP throughput monitoring devices that monitor every single data packets transmitted and received, the network controller 140 is an "out-of-band" computer server that interfaces with the steer-ing device 130 to selectively inspect user flows of interest. The network controller 140 may further identify user flows (e.g., among the flows of interest) for optimization. In one embodiment, the network controller 140 may be imple-mented at the steering device 130 to monitor traffic. In other embodiments, the network controller 140 is coupled to and communicates with the steering device 130 for traffic moni-toring and optimization. When queried by the steering device 130, the network controller 140 determines if a given network flow should be ignored, monitored further or optimized. Opti-mization of a flow is often decided at the beginning of the flow because it is rarely possible to switch to optimized content mid-stream once non-optimized content delivery has begun. However, the network controller 140 may determine that existing flows associated with a particular subscriber or other entity should be optimized. In turn, new flows ( e.g., resulting from seek requests in media, new media requests, resume after pause, etc.) determined to be associated with the entity may be |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 208 of 1100

| No. | '111 Patent Claim 4 | Swenson |
|---|---|---|
| | | optimized. The network controller 140 uses the net-work state as well as historical traffic data in its decision for monitoring and optimization. Knowledge on the current net-work state, such as congestion, deems critical when it comes to data optimization.")<br><br>Swenson at [0029] ("As a flow is sent to the network controller 140 for inspection, historical network traffic data stored at the net-work controller 140 may be searched. The historical network traffic data includes information such as subscriber informa-tion, the cell towers to which the user devices attached, rout-ers through which the traffic is passing, geography regions, the backhaul segments, and time-of-day of the flows. For example, in a mobile network, the cell tower to which a user device is attached can be most useful, since it is the location where most congestion occurs due to limited bandwidth and high cost of the radio access network infrastructure. The network controller 140 looks into the historical traffic data for the average of the bandwidth per user at the particular cell tower. The network controller 140 can then estimate the amount ofbandwidth or degree of congestion for the new flow based on the historical record.")<br><br>Swenson at [0045] ("The steering device interface 316 interacts with an external routing appliance, such as the steering device 130 to divert portions of the network traffic ( e.g., large object net-work flows). Existing routing appliances in most carrier net-works are designed to handle large amounts of network traf-fic. They are not, however, ideal devices to operate for monitoring and analysis individual flows. Through the steer-ing device interface 316, the network controller 140 may communicate with the external routing appliances, such as the steering device 130, to steer a portion of network traffic to the network controller 140 when certain conditions are met. Generally, network flows of interest to the network controller 140 contain larger media objects, such as videos and images. In one embodiment, the smaller flows, such as web page and text information, are not exchanged over the steering device interface 316.")<br><br>Swenson at [0059] ("In one embodiment, as the steering device 130 monitors network responses, it is looking for flows that match one or more signatures for video and images. When a match-ing flow is detected, the steering device 130 forwards the HTTP request and a portion of the HTTP response to the network controller 140 over the ICAP client interface 404. After receiving the request and the portion of response at the ICAP server interface 406, the flow analyzer 312 of the net-work controller 140 performs a deep flow inspection |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 209 of 1100

| No. | '111 Patent Claim 4 | Swenson |
|---|---|---|
| | | to deter-mine if the flow is worth bandwidth monitoring and/or user detection. For example, the flow inspection performed by the flow analyzer 312 may determine if the flow indeed contains large or medium object ( e.g., larger than 50 kB), and/or if the source IP address of the flow is from a user or a group of users that are required to be monitored by policies. The flow ana-lyzer 312 may also determine if the flow needs to be opti-mized based on historical flow statistical data.")<br><br>Swenson at [0060] ("If the flow is deemed of interest, the steering device 130 is notified to steer the flow through the network controller 140. This is known as the "continue" working mode for bandwidth monitoring. In the "continue" mode, the network controller 140 interfaces with the steering device 130 to func-tion, on-demand, as a traditional inline network element for flows deemed of interest. Thus, the network controller 140 ingests the network flow for inspection and subsequently forwards the network flow on the network response path. For example, for this particular flow, the origin server 160 responds to the user request by sending video or images over the network link 413 to the steering device 130, which for-wards the video or images to the network controller 140 over a network link 414. After the network controller 140 updates the flow statistics, the video or images are returned to the steering device 130 over a network link 415, which transmits the video or images to the user device 110 over the network link 416.")<br><br>Swenson at [0061] ("Once a flow is reported to the network controller 140, a flow cache entry is created for the flow in the flow cache 322. The flow cache entry keeps track of the flow and its associated bandwidth. For a flow that is marked in "continue" mode, each time the steering device 130 forwards a next portion of the flow payload to the network controller 140, the flow cache 3 22 updates the number of bytes for transmitted in the flow. By monitoring the number of bytes per flow over time, the flow analyzer 312 is capable of determining an estimate value of bandwidth associated with flow. Further-more, since the steering device 130 does not have infinite packet buffers, if congestion happens on the network link 416 from the steering device 130 to the user device 110, the TCP congestion control mechanism kicks in at the steering device 130, which may slows down and/or eventually stop receiving data over the network link 413 from origin server 160. During the congestion, the steering device 130 would not forward any data to the network controller 140, since the link 416 is congested and the network controller 140 would not be able to transmit data to the user device 110. Therefore, as an inline element, the network controller |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 210 of 1100

| No. | '111 Patent Claim 4 | Swenson |
|---|---|---|
| | | 140 can detect network con-gestions and estimate bandwidth associated with any flows of interest selected by the network controller 140. However, in the "continue" mode, the network controller 140 does not modify and transform the HTTP messaged it receives over the ICAP interface. The network controller 140 simply updates the flow statistics and returns the video or images to the steering device 130 for transmission to the user device 110.")<br><br>Swenson at [0065] ("FIG. 5 is a block diagram illustrating an example event trace of "continue" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 512 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 512 to the intended origin server 160 and receives a response 514 back from the origin server 160. The steering device 130 then sends an ICAP request message 516 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 518. Upon receiving the instruc-tion, the steering device 130 re-writes the response 514 to an HTTP redirect response 520, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 520 directly to the user device 110. In case the flow dose not contain video or image objects, or the network controller 140 determines not to monitor the flow, the steering device 13 0 would forward the response to the user device 110.")<br><br>Swenson at [0069] ("FIG. 6 is a block diagram illustrating an example event trace of "counting" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 612 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 612 to the intended origin server 160 and receives a response 614 back from the origin server 160. The steering device 130 then sends an ICAP request message 616 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which |

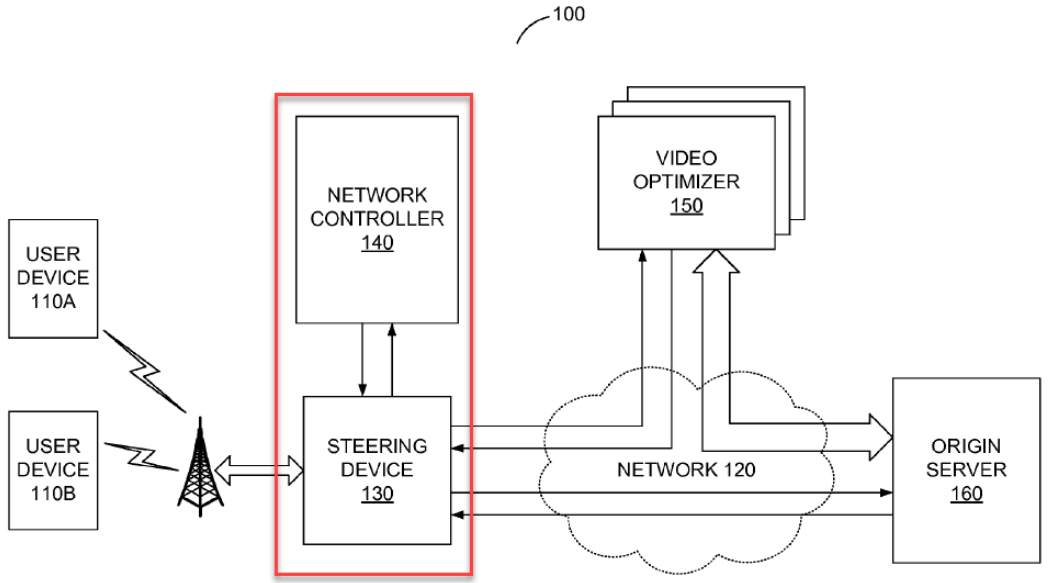


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 211 of 1100

| No. | '111 Patent Claim 4 | Swenson |
|---|---|---|
| | | inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 618. Upon receiving the instruc-tion, the steering device 130 re-writes the response 614 to an HTTP redirect response 620, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 620 directly to the user device 110. In case the flow dose not contain video or image objects that need to be redirected, the steering device 130 would forward the response to the user device 110.")<br><br>Swenson at [0071] ("After receiving the request, the video optimizer 150 forwards the video HTTP GET requests 622 to the origin server 160 and in return, receives a video file 624 from the origin server 160. The video optimizer 150 transcodes the video file to a format usable by the client device 110 based on network bandwidth available to the user device 110. The optimized video 626 is then transmitted from the video opti-mizer 150 to the steering device 130. In one embodiment, the steering device 130 intercepts the optimized video 626. The steering device 130 will then send an ICAP request to the network controller 140 for inspection. The network controller 140 deems this flow to be monitored and sends ICAP response 630. The steering device 130 then allows the flow to go through to the user device 110. The steering device 130 next sends periodic ICAP "counting" updates 632 to the network controller 140 until the flow completes. As such, the client receives the optimized video 626 for substantially real-time playback on an application executing on the user device 110.")<br><br>Swenson at Figure 1 (annotation added) |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 212 of 1100

| No. | '111 Patent Claim 4 | Swenson |
|---|---|---|
| | |  <br>FIG. 1 |
| 4[c] | responsive to receiving the packet, analyzing the packet, by the controller; | Swenson discloses responsive to receiving the packet, analyzing the packet, by the controller.<br><br>For example, Swenson discloses the network controller comprising a flow analyzer for analyzing and inspecting the packet.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 213 of 1100

| No. | '111 Patent Claim 4 | Swenson |
|---|---|---|
| | | interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0028] ("In contrast to conventional inline TCP throughput monitoring devices that monitor every single data packets transmitted and received, the network controller 140 is an "out-of-band" computer server that interfaces with the steer-ing device 130 to selectively inspect user flows of interest. The network controller 140 may further identify user flows (e.g., among the flows of interest) for optimization. In one embodiment, the network controller 140 may be imple-mented at the steering device 130 to monitor traffic. In other embodiments, the network controller 140 is coupled to and communicates with the steering device 130 for traffic moni-toring and optimization. When queried by the steering device 130, the network controller 140 determines if a given network flow should be ignored, monitored further or optimized. Opti-mization of a flow is often decided at the beginning of the flow because it is rarely possible to switch to optimized content mid-stream once non-optimized content delivery has begun. However, the network controller 140 may determine that existing flows associated with a particular subscriber or other entity should be optimized. In turn, new flows ( e.g., resulting from seek requests in media, new media requests, resume after pause, etc.) determined to be associated with the entity may be optimized. The network controller 140 uses the net-work state as well as historical traffic data in its decision for monitoring and optimization. Knowledge on the current net-work state, such as congestion, deems critical when it comes to data optimization.")<br><br>Swenson at [0029] ("As a flow is sent to the network controller 140 for inspection, historical network traffic data stored at the net-work controller 140 may be searched. The historical network traffic data includes information such as subscriber informa-tion, the cell towers to which the user devices attached, rout-ers through which the traffic is passing, geography regions, the backhaul segments, and time-of-day of the flows. For example, in a mobile network, the cell tower to which a user device is attached can be most useful, since it is the location where most congestion occurs due to limited bandwidth and high cost of the radio access network infrastructure. The network controller 140 looks into the historical traffic data for the average of the bandwidth per user at the particular cell tower. The |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 214 of 1100

| No. | '111 Patent Claim 4 | Swenson |
|---|---|---|
| | | network controller 140 can then estimate the amount of bandwidth or degree of congestion for the new flow based on the historical record.")<br><br>Swenson at [0038] ("Turning back to FIG. 1, the network controller 140 allows network operators to apply fine granular optimization policies to ensure high quality of experience (QoE) based on cell tower congestion, device types, subscriber profiles and service plans with lower hardware and software costs. The architecture of the network controller 140 provides an excel-lent fit for the net neutrality guideline of "reasonable network management", and better compliance to the copyright law (DMCA) than solutions that rely on long-term caching. Hav-ing the ability of monitoring network traffic on a per sub-scriber, per flow, or per video file basis, the network controller 140 also selectively monitors and optimizes only a subset of traffic that benefits from optimization the most, thus achiev-ing both scalability and efficiency for optimization at a com-petitive price-point. The core element of the network control-ler 140 lies in its mechanisms for congestion detection and mitigation, which allows optimization resources to be utilized in the most efficient and surgical manner.")<br><br>Swenson at [0039] ("Referring now to FIG. 3, it illustrates one embodi-ment of an example architecture of the network controller 140 for providing selective real-time network monitoring and subscriber identification. The network controller 140 com-prises a flow analyzer 312, a policy engine 314, a steering device interface 316, a video optimizer redirector 318, a flow cache 322, and a subscriber log 324. In other embodiments, the network controller 140 may include additional, fewer, or different components for various applications. Conventional components such as network interfaces, security functions, failover servers, management and network operations con-soles, and the like are not shown so as to not obscure the details of the system architecture.")<br><br>Swenson at [0059] ("In one embodiment, as the steering device 130 monitors network responses, it is looking for flows that match one or more signatures for video and images. When a match-ing flow is detected, the steering device 130 forwards the HTTP request and a portion of the HTTP response to the network controller 140 over the ICAP client interface 404. After receiving the request and the portion of response at the ICAP server interface 406, the flow analyzer 312 of the net-work controller 140 performs a deep flow inspection to deter-mine if the flow is worth bandwidth monitoring and/or user detection. For example, |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 215 of 1100

| No. | '111 Patent Claim 4 | Swenson |
|---|---|---|
| | | the flow inspection performed by the flow analyzer 312 may determine if the flow indeed contains large or medium object ( e.g., larger than 50 kB), and/or if the source IP address of the flow is from a user or a group of users that are required to be monitored by policies. The flow ana-lyzer 312 may also determine if the flow needs to be opti-mized based on historical flow statistical data.")<br><br>Swenson at [0060] ("If the flow is deemed of interest, the steering device 130 is notified to steer the flow through the network controller 140. This is known as the "continue" working mode for bandwidth monitoring. In the "continue" mode, the network controller 140 interfaces with the steering device 130 to func-tion, on-demand, as a traditional inline network element for flows deemed of interest. Thus, the network controller 140 ingests the network flow for inspection and subsequently forwards the network flow on the network response path. For example, for this particular flow, the origin server 160 responds to the user request by sending video or images over the network link 413 to the steering device 130, which for-wards the video or images to the network controller 140 over a network link 414. After the network controller 140 updates the flow statistics, the video or images are returned to the steering device 130 over a network link 415, which transmits the video or images to the user device 110 over the network link 416.")<br><br>Swenson at [0061] ("Once a flow is reported to the network controller 140, a flow cache entry is created for the flow in the flow cache 322. The flow cache entry keeps track of the flow and its associated bandwidth. For a flow that is marked in "continue" mode, each time the steering device 130 forwards a next portion of the flow payload to the network controller 140, the flow cache 3 22 updates the number of bytes for transmitted in the flow. By monitoring the number of bytes per flow over time, the flow analyzer 312 is capable of determining an estimate value of bandwidth associated with flow. Further-more, since the steering device 130 does not have infinite packet buffers, if congestion happens on the network link 416 from the steering device 130 to the user device 110, the TCP congestion control mechanism kicks in at the steering device 130, which may slows down and/or eventually stop receiving data over the network link 413 from origin server 160. During the congestion, the steering device 130 would not forward any data to the network controller 140, since the link 416 is congested and the network controller 140 would not be able to transmit data to the user device 110. Therefore, as an inline element, the network controller 140 can detect network con-gestions and estimate bandwidth associated with any flows of |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 216 of 1100

| No. | '111 Patent Claim 4 | Swenson |
|---|---|---|
| | | interest selected by the network controller 140. However, in the "continue" mode, the network controller 140 does not modify and transform the HTTP messaged it receives over the ICAP interface. The network controller 140 simply updates the flow statistics and returns the video or images to the steering device 130 for transmission to the user device 110.")<br><br>Swenson at [0065] ("FIG. 5 is a block diagram illustrating an example event trace of "continue" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 512 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 512 to the intended origin server 160 and receives a response 514 back from the origin server 160. The steering device 130 then sends an ICAP request message 516 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 518. Upon receiving the instruc-tion, the steering device 130 re-writes the response 514 to an HTTP redirect response 520, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 520 directly to the user device 110. In case the flow dose not contain video or image objects, or the network controller 140 determines not to monitor the flow, the steering device 13 0 would forward the response to the user device 110.")<br><br>Swenson at [0069] ("FIG. 6 is a block diagram illustrating an example event trace of "counting" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 612 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 612 to the intended origin server 160 and receives a response 614 back from the origin server 160. The steering device 130 then sends an ICAP request message 616 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this |

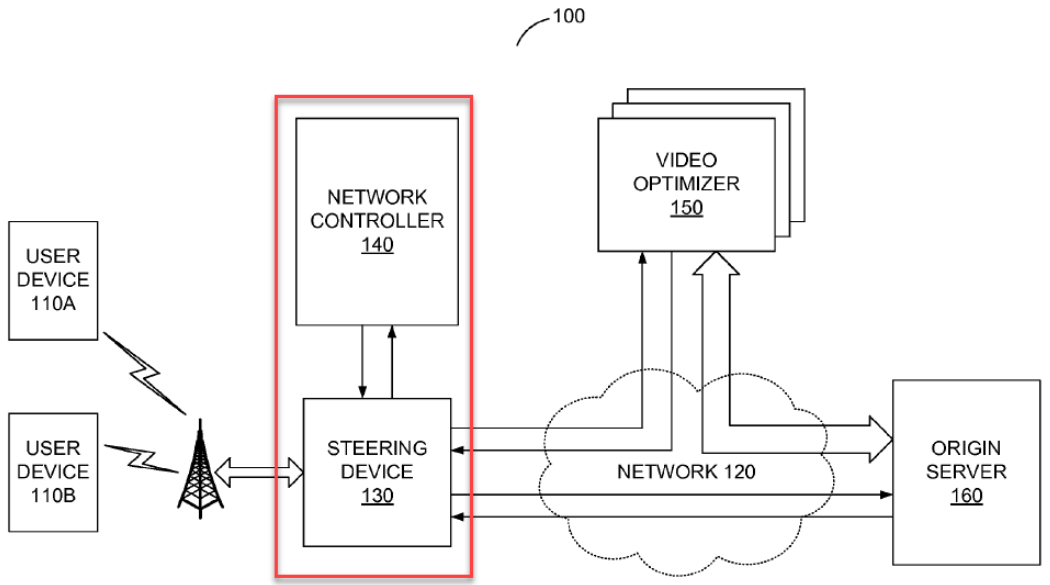


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 217 of 1100

| No. | '111 Patent Claim 4 | Swenson |
|---|---|---|
| | | case, the network controller 140 responds with a redirect to optimize the video in ICAP response 618. Upon receiving the instruc-tion, the steering device 130 re-writes the response 614 to an HTTP redirect response 620, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 620 directly to the user device 110. In case the flow dose not contain video or image objects that need to be redirected, the steering device 130 would forward the response to the user device 110.")<br><br>Swenson at [0071] ("After receiving the request, the video optimizer 150 forwards the video HTTP GET requests 622 to the origin server 160 and in return, receives a video file 624 from the origin server 160. The video optimizer 150 transcodes the video file to a format usable by the client device 110 based on network bandwidth available to the user device 110. The optimized video 626 is then transmitted from the video opti-mizer 150 to the steering device 130. In one embodiment, the steering device 130 intercepts the optimized video 626. The steering device 130 will then send an ICAP request to the network controller 140 for inspection. The network controller 140 deems this flow to be monitored and sends ICAP response 630. The steering device 130 then allows the flow to go through to the user device 110. The steering device 130 next sends periodic ICAP "counting" updates 632 to the network controller 140 until the flow completes. As such, the client receives the optimized video 626 for substantially real-time playback on an application executing on the user device 110.")<br><br>Swenson at Figure 1 (annotation added) |

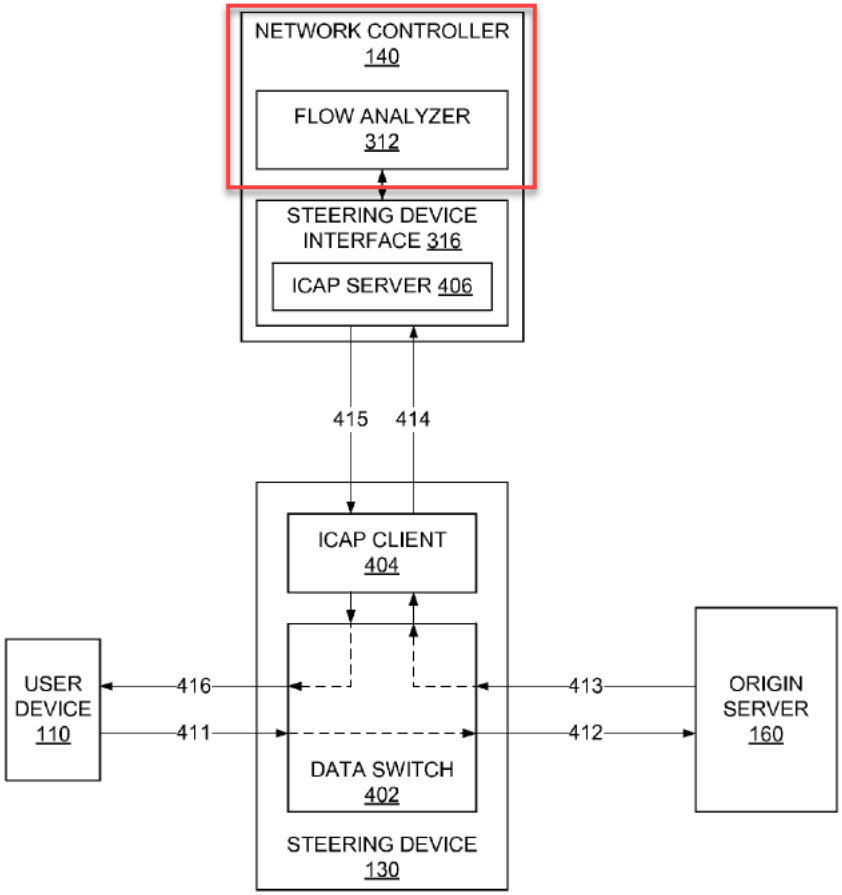


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 218 of 1100

| No. | '111 Patent Claim 4 | Swenson |
|---|---|---|
| | |  <br>FIG. 1<br>Swenson at Figure 4A (annotation added) |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 219 of 1100

| No. | '111 Patent Claim 4 | Swenson |
|---|---|---|
| | |  <br>FIG. 4A |
| 4[d] | sending the packet, by the controller, to the network node; and | Swenson discloses sending the packet, by the controller, to the network node. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 220 of 1100

| No. | '111 Patent Claim 4 | Swenson |
|---|---|---|
| | | For example, Swenson discloses sending the packet, for example a video or image, back to the steering device after the network controller analyzes the packet and updates flow statistics.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0028] ("In contrast to conventional inline TCP throughput monitoring devices that monitor every single data packets transmitted and received, the network controller 140 is an "out-of-band" computer server that interfaces with the steer-ing device 130 to selectively inspect user flows of interest. The network controller 140 may further identify user flows (e.g., among the flows of interest) for optimization. In one embodiment, the network controller 140 may be imple-mented at the steering device 130 to monitor traffic. In other embodiments, the network controller 140 is coupled to and communicates with the steering device 130 for traffic moni-toring and optimization. When queried by the steering device 130, the network controller 140 determines if a given network flow should be ignored, monitored further or optimized. Opti-mization of a flow is often decided at the beginning of the flow because it is rarely possible to switch to optimized content mid-stream once non-optimized content delivery has begun. However, the network controller 140 may determine that existing flows associated with a particular subscriber or other entity should be optimized. In turn, new flows ( e.g., resulting from seek requests in media, new media requests, resume after pause, etc.) determined to be associated with the entity may be optimized. The network controller 140 uses the net-work state as well as historical traffic data in its decision for monitoring and optimization. Knowledge on the current net-work state, such as congestion, deems critical when it comes to data optimization.") |


~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 221 of 1100

| No. | '111 Patent Claim 4 | Swenson |
|---|---|---|
| | | Swenson at [0029] ("As a flow is sent to the network controller 140 for inspection, historical network traffic data stored at the net-work controller 140 may be searched. The historical network traffic data includes information such as subscriber informa-tion, the cell towers to which the user devices attached, rout-ers through which the traffic is passing, geography regions, the backhaul segments, and time-of-day of the flows. For example, in a mobile network, the cell tower to which a user device is attached can be most useful, since it is the location where most congestion occurs due to limited bandwidth and high cost of the radio access network infrastructure. The network controller 140 looks into the historical traffic data for the average of the bandwidth per user at the particular cell tower. The network controller 140 can then estimate the amount ofbandwidth or degree of congestion for the new flow based on the historical record.")<br><br>Swenson at [0057] ("The Internet content adaption protocol is a light-weight protocol aimed at executing a simple remote proce-dure call on HTTP messages. ICAP leverages edge-based devices to help deliver value-added services using transparent HTTP proxy caches. Content adaptation refers to performing the particular value added service, such as content manipula-tion or other processing, for the associated HTTP client request/response. ICAP clients pass HTTP messages to ICAP servers for transformation or other processing. In tum, the ICAP server executes its transformation service on the HTTP messages and sends back responses to the ICAP client. At the core of this process is a cache that can proxy all client trans-actions and process them through ICAP servers, which may focus on specific functions, such as ad insertion, virus scan-ning, content translation, language translation, or content fil-tering. ICAP servers, such as those utilized by the network controller 140, handle these tasks to off-load value-added services from network devices including an ICAP client, such as the steering device 130. By offloading value added services from the steering device 130, processing infrastructure (e.g., optimization services and network controllers) may be scaled independent from the steering devices handling raw HTTP throughput.")<br><br>Swenson at [0059] ("In one embodiment, as the steering device 130 monitors network responses, it is looking for flows that match one or more signatures for video and images. When a match-ing flow is detected, the steering device 130 forwards the HTTP request and a portion of the HTTP response to the network controller 140 over the ICAP client interface 404. After receiving the request and the portion of response at the ICAP server interface |

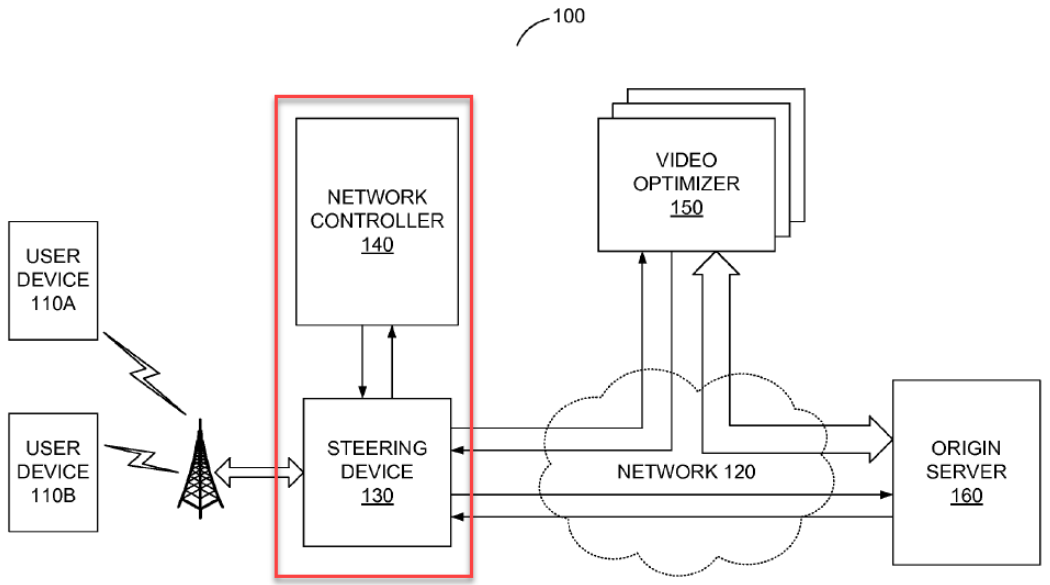


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 222 of 1100

| No. | '111 Patent Claim 4 | Swenson |
|---|---|---|
| | | 406, the flow analyzer 312 of the net-work controller 140 performs a deep flow inspection to deter-mine if the flow is worth bandwidth monitoring and/or user detection. For example, the flow inspection performed by the flow analyzer 312 may determine if the flow indeed contains large or medium object ( e.g., larger than 50 kB), and/or if the source IP address of the flow is from a user or a group of users that are required to be monitored by policies. The flow ana-lyzer 312 may also determine if the flow needs to be opti-mized based on historical flow statistical data.")<br><br>Swenson at [0060] ("If the flow is deemed of interest, the steering device 130 is notified to steer the flow through the network controller 140. This is known as the "continue" working mode for bandwidth monitoring. In the "continue" mode, the network controller 140 interfaces with the steering device 130 to func-tion, on-demand, as a traditional inline network element for flows deemed of interest. Thus, the network controller 140 ingests the network flow for inspection and subsequently forwards the network flow on the network response path. For example, for this particular flow, the origin server 160 responds to the user request by sending video or images over the network link 413 to the steering device 130, which for-wards the video or images to the network controller 140 over a network link 414. After the network controller 140 updates the flow statistics, the video or images are returned to the steering device 130 over a network link 415, which transmits the video or images to the user device 110 over the network link 416.")<br><br>Swenson at [0071] ("After receiving the request, the video optimizer 150 forwards the video HTTP GET requests 622 to the origin server 160 and in return, receives a video file 624 from the origin server 160. The video optimizer 150 transcodes the video file to a format usable by the client device 110 based on network bandwidth available to the user device 110. The optimized video 626 is then transmitted from the video opti-mizer 150 to the steering device 130. In one embodiment, the steering device 130 intercepts the optimized video 626. The steering device 130 will then send an ICAP request to the network controller 140 for inspection. The network controller 140 deems this flow to be monitored and sends ICAP response 630. The steering device 130 then allows the flow to go through to the user device 110. The steering device 130 next sends periodic ICAP "counting" updates 632 to the network controller 140 until the flow completes. As such, the client receives the optimized video 626 for substantially real-time playback on an application executing on the user device 110.") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 223 of 1100

| No. | '111 Patent Claim 4 | Swenson |
|---|---|---|
| | | Swenson at Figure 1 (annotation added)<br>  <br><br>FIG. 1 |
| 4[e] | responsive to receiving the packet, sending the packet, by the network node, to the second entity. | Swenson discloses responsive to receiving the packet, sending the packet, by the network node, to the second entity.<br><br>For example, Swenson discloses sending the packet, for example a video or image, back to the steering device after the network controller analyzes the packet and updates flow statistics. Swenson further discloses the steering device, upon having the packet returned to it, transmitting the packet to the destination entity, for example, the user device or origin server. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 224 of 1100

| No. | '111 Patent Claim 4 | Swenson |
|---|---|---|
| | | Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0028] ("In contrast to conventional inline TCP throughput monitoring devices that monitor every single data packets transmitted and received, the network controller 140 is an "out-of-band" computer server that interfaces with the steer-ing device 130 to selectively inspect user flows of interest. The network controller 140 may further identify user flows (e.g., among the flows of interest) for optimization. In one embodiment, the network controller 140 may be imple-mented at the steering device 130 to monitor traffic. In other embodiments, the network controller 140 is coupled to and communicates with the steering device 130 for traffic moni-toring and optimization. When queried by the steering device 130, the network controller 140 determines if a given network flow should be ignored, monitored further or optimized. Opti-mization of a flow is often decided at the beginning of the flow because it is rarely possible to switch to optimized content mid-stream once non-optimized content delivery has begun. However, the network controller 140 may determine that existing flows associated with a particular subscriber or other entity should be optimized. In turn, new flows ( e.g., resulting from seek requests in media, new media requests, resume after pause, etc.) determined to be associated with the entity may be optimized. The network controller 140 uses the net-work state as well as historical traffic data in its decision for monitoring and optimization. Knowledge on the current net-work state, such as congestion, deems critical when it comes to data optimization.")<br><br>Swenson at [0029] ("As a flow is sent to the network controller 140 for inspection, historical network traffic data stored at the net-work controller 140 may be searched. The historical network traffic data includes information such as subscriber informa-tion, the cell towers to which the user devices attached, rout-ers through which the traffic is passing, |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 225 of 1100

| No. | '111 Patent Claim 4 | Swenson |
|---|---|---|
| | | geography regions, the backhaul segments, and time-of-day of the flows. For example, in a mobile network, the cell tower to which a user device is attached can be most useful, since it is the location where most congestion occurs due to limited bandwidth and high cost of the radio access network infrastructure. The network controller 140 looks into the historical traffic data for the average of the bandwidth per user at the particular cell tower. The network controller 140 can then estimate the amount of bandwidth or degree of congestion for the new flow based on the historical record.")<br><br>Swenson at [0057] ("The Internet content adaption protocol is a light-weight protocol aimed at executing a simple remote proce-dure call on HTTP messages. ICAP leverages edge-based devices to help deliver value-added services using transparent HTTP proxy caches. Content adaptation refers to performing the particular value added service, such as content manipula-tion or other processing, for the associated HTTP client request/response. ICAP clients pass HTTP messages to ICAP servers for transformation or other processing. In tum, the ICAP server executes its transformation service on the HTTP messages and sends back responses to the ICAP client. At the core of this process is a cache that can proxy all client trans-actions and process them through ICAP servers, which may focus on specific functions, such as ad insertion, virus scan-ning, content translation, language translation, or content fil-tering. ICAP servers, such as those utilized by the network controller 140, handle these tasks to off-load value-added services from network devices including an ICAP client, such as the steering device 130. By offloading value added services from the steering device 130, processing infrastructure (e.g., optimization services and network controllers) may be scaled independent from the steering devices handling raw HTTP throughput.")<br><br>Swenson at [0059] ("In one embodiment, as the steering device 130 monitors network responses, it is looking for flows that match one or more signatures for video and images. When a match-ing flow is detected, the steering device 130 forwards the HTTP request and a portion of the HTTP response to the network controller 140 over the ICAP client interface 404. After receiving the request and the portion of response at the ICAP server interface 406, the flow analyzer 312 of the net-work controller 140 performs a deep flow inspection to deter-mine if the flow is worth bandwidth monitoring and/or user detection. For example, the flow inspection performed by the flow analyzer 312 may determine if the flow indeed contains large or medium object ( e.g., larger than 50 kB), and/or if the source IP address of the flow is from a user or a group of users that are required to be monitored by policies. The |

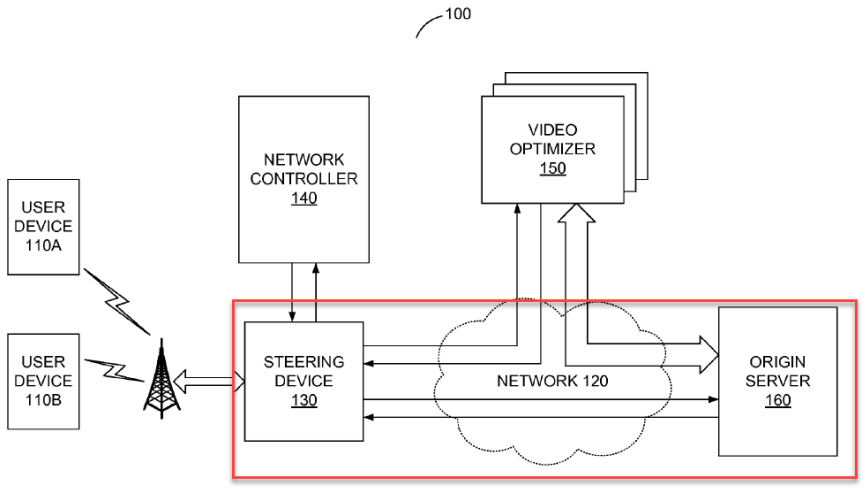


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 226 of 1100

| No. | ’111 Patent Claim 4 | Swenson |
|---|---|---|
| | | flow ana-lyzer 312 may also determine if the flow needs to be opti-mized based on historical flow statistical data.”)<br><br>Swenson at [0060] (“If the flow is deemed of interest, the steering device 130 is notified to steer the flow through the network controller 140. This is known as the "continue" working mode for bandwidth monitoring. In the "continue" mode, the network controller 140 interfaces with the steering device 130 to func-tion, on-demand, as a traditional inline network element for flows deemed of interest. Thus, the network controller 140 ingests the network flow for inspection and subsequently forwards the network flow on the network response path. For example, for this particular flow, the origin server 160 responds to the user request by sending video or images over the network link 413 to the steering device 130, which for-wards the video or images to the network controller 140 over a network link 414. After the network controller 140 updates the flow statistics, the video or images are returned to the steering device 130 over a network link 415, which transmits the video or images to the user device 110 over the network link 416.”)<br><br>Swenson at [0071] (“After receiving the request, the video optimizer 150 forwards the video HTTP GET requests 622 to the origin server 160 and in return, receives a video file 624 from the origin server 160. The video optimizer 150 transcodes the video file to a format usable by the client device 110 based on network bandwidth available to the user device 110. The optimized video 626 is then transmitted from the video opti-mizer 150 to the steering device 130. In one embodiment, the steering device 130 intercepts the optimized video 626. The steering device 130 will then send an ICAP request to the network controller 140 for inspection. The network controller 140 deems this flow to be monitored and sends ICAP response 630. The steering device 130 then allows the flow to go through to the user device 110. The steering device 130 next sends periodic ICAP "counting" updates 632 to the network controller 140 until the flow completes. As such, the client receives the optimized video 626 for substantially real-time playback on an application executing on the user device 110.”)<br><br>Swenson at Figure 1 (annotation added) |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 227 of 1100

| No. | '111 Patent Claim 4 | Swenson |
|---|---|---|
| | |  <br>FIG. 1 |

| No. | '111 Patent Claim 5 | Swenson |
|---|---|---|
| 5 | The method according to claim 1, further comprising responsive to the packet satisfying the criterion and to the instruction, sending the packet or a portion thereof, by the network node, to the controller. | Swenson discloses the method according to claim 1, further comprising responsive to the packet satisfying the criterion and to the instruction, sending the packet or a portion thereof, by the network node, to the controller.<br><br>For example, Swenson discloses determining by the steering device flows that match one or more signatures or criteria of the packet. Swenson further discloses that when a matching flow is detected, the steering device forwards the HTTP request and a portion of the HTTP response to the network controller.<br><br>*See supra* at Claim 1.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 228 of 1100

| No. | '111 Patent Claim 5 | Swenson |
|---|---|---|
| | | user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0028] ("In contrast to conventional inline TCP throughput monitoring devices that monitor every single data packets transmitted and received, the network controller 140 is an "out-of-band" computer server that interfaces with the steer-ing device 130 to selectively inspect user flows of interest. The network controller 140 may further identify user flows (e.g., among the flows of interest) for optimization. In one embodiment, the network controller 140 may be imple-mented at the steering device 130 to monitor traffic. In other embodiments, the network controller 140 is coupled to and communicates with the steering device 130 for traffic moni-toring and optimization. When queried by the steering device 130, the network controller 140 determines if a given network flow should be ignored, monitored further or optimized. Opti-mization of a flow is often decided at the beginning of the flow because it is rarely possible to switch to optimized content mid-stream once non-optimized content delivery has begun. However, the network controller 140 may determine that existing flows associated with a particular subscriber or other entity should be optimized. In turn, new flows ( e.g., resulting from seek requests in media, new media requests, resume after pause, etc.) determined to be associated with the entity may be optimized. The network controller 140 uses the net-work state as well as historical traffic data in its decision for monitoring and optimization. Knowledge on the current net-work state, such as congestion, deems critical when it comes to data optimization.")<br><br>Swenson at [0029] ("As a flow is sent to the network controller 140 for inspection, historical network traffic data stored at the net-work controller 140 may be searched. The historical network traffic data includes information such as subscriber informa-tion, the cell towers to which the user devices attached, rout-ers through which the traffic is passing, geography regions, the backhaul segments, and time-of-day of the flows. For example, in a mobile network, the cell tower to which a user device is attached can be most useful, since it is the location where most congestion occurs due to limited bandwidth and high cost of the |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 229 of 1100

| No. | '111 Patent Claim 5 | Swenson |
|---|---|---|
| | | radio access network infrastructure. The network controller 140 looks into the historical traffic data for the average of the bandwidth per user at the particular cell tower. The network controller 140 can then estimate the amount ofbandwidth or degree of congestion for the new flow based on the historical record.")<br><br>Swenson at [0045] ("The steering device interface 316 interacts with an external routing appliance, such as the steering device 130 to divert portions of the network traffic ( e.g., large object net-work flows). Existing routing appliances in most carrier net-works are designed to handle large amounts of network traf-fic. They are not, however, ideal devices to operate for monitoring and analysis individual flows. Through the steer-ing device interface 316, the network controller 140 may communicate with the external routing appliances, such as the steering device 130, to steer a portion of network traffic to the network controller 140 when certain conditions are met. Generally, network flows of interest to the network controller 140 contain larger media objects, such as videos and images. In one embodiment, the smaller flows, such as web page and text information, are not exchanged over the steering device interface 316.")<br><br>Swenson at [0059] ("In one embodiment, as the steering device 130 monitors network responses, it is looking for flows that match one or more signatures for video and images. When a match-ing flow is detected, the steering device 130 forwards the HTTP request and a portion of the HTTP response to the network controller 140 over the ICAP client interface 404. After receiving the request and the portion of response at the ICAP server interface 406, the flow analyzer 312 of the net-work controller 140 performs a deep flow inspection to deter-mine if the flow is worth bandwidth monitoring and/or user detection. For example, the flow inspection performed by the flow analyzer 312 may determine if the flow indeed contains large or medium object ( e.g., larger than 50 kB), and/or if the source IP address of the flow is from a user or a group of users that are required to be monitored by policies. The flow ana-lyzer 312 may also determine if the flow needs to be opti-mized based on historical flow statistical data.")<br><br>Swenson at [0061] ("Once a flow is reported to the network controller 140, a flow cache entry is created for the flow in the flow cache 322. The flow cache entry keeps track of the flow and its associated bandwidth. For a flow that is marked in "continue" mode, each time the steering device 130 forwards a next portion of the flow payload to the network controller |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 230 of 1100

| No. | '111 Patent Claim 5 | Swenson |
|---|---|---|
| | | 140, the flow cache 3 22 updates the number of bytes for transmitted in the flow. By monitoring the number of bytes per flow over time, the flow analyzer 312 is capable of determining an estimate value of bandwidth associated with flow. Further-more, since the steering device 130 does not have infinite packet buffers, if congestion happens on the network link 416 from the steering device 130 to the user device 110, the TCP congestion control mechanism kicks in at the steering device 130, which may slows down and/or eventually stop receiving data over the network link 413 from origin server 160. During the congestion, the steering device 130 would not forward any data to the network controller 140, since the link 416 is congested and the network controller 140 would not be able to transmit data to the user device 110. Therefore, as an inline element, the network controller 140 can detect network con-gestions and estimate bandwidth associated with any flows of interest selected by the network controller 140. However, in the "continue" mode, the network controller 140 does not modify and transform the HTTP messaged it receives over the ICAP interface. The network controller 140 simply updates the flow statistics and returns the video or images to the steering device 130 for transmission to the user device 110.")<br><br>Swenson at [0065] ("FIG. 5 is a block diagram illustrating an example event trace of "continue" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 512 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 512 to the intended origin server 160 and receives a response 514 back from the origin server 160. The steering device 130 then sends an ICAP request message 516 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 518. Upon receiving the instruc-tion, the steering device 130 re-writes the response 514 to an HTTP redirect response 520, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 520 directly to the user device 110. In case the flow dose not contain video or image objects, or the network controller 140 determines not to monitor the flow, the steering device 13 0 would forward the response to the user device 110.") |

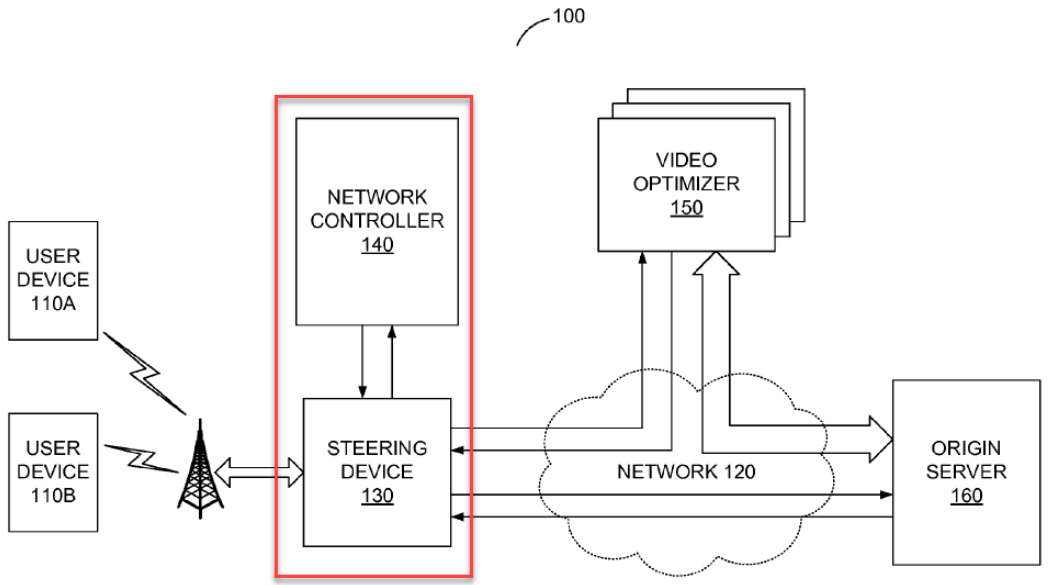


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 231 of 1100

| No. | '111 Patent Claim 5 | Swenson |
|---|---|---|
| | | Swenson at [0069] ("FIG. 6 is a block diagram illustrating an example event trace of "counting" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 612 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 612 to the intended origin server 160 and receives a response 614 back from the origin server 160. The steering device 130 then sends an ICAP request message 616 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 618. Upon receiving the instruc-tion, the steering device 130 re-writes the response 614 to an HTTP redirect response 620, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 620 directly to the user device 110. In case the flow dose not contain video or image objects that need to be redirected, the steering device 130 would forward the response to the user device 110.")<br><br>Swenson at Figure 1 (annotation added) |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 232 of 1100

| No. | '111 Patent Claim 5 | Swenson |
|---|---|---|
| | |  <br>FIG. 1 |

| No. | '111 Patent Claim 6 | Swenson |
|---|---|---|
| 6 | The method according to claim 5, further comprising storing the received packet or a portion thereof, by the controller, in a memory. | Swenson discloses the method according to claim 5, further comprising storing the received packet or a portion thereof, by the controller, in a memory.<br><br>For example, Swenson discloses the network controller storing historical network traffic data based on received packet flows.<br><br>*See supra* at Claim 5.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 233 of 1100

| No. | '111 Patent Claim 6 | Swenson |
|---|---|---|
| | | device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0028] ("In contrast to conventional inline TCP throughput monitoring devices that monitor every single data packets transmitted and received, the network controller 140 is an "out-of-band" computer server that interfaces with the steer-ing device 130 to selectively inspect user flows of interest. The network controller 140 may further identify user flows (e.g., among the flows of interest) for optimization. In one embodiment, the network controller 140 may be imple-mented at the steering device 130 to monitor traffic. In other embodiments, the network controller 140 is coupled to and communicates with the steering device 130 for traffic moni-toring and optimization. When queried by the steering device 130, the network controller 140 determines if a given network flow should be ignored, monitored further or optimized. Opti-mization of a flow is often decided at the beginning of the flow because it is rarely possible to switch to optimized content mid-stream once non-optimized content delivery has begun. However, the network controller 140 may determine that existing flows associated with a particular subscriber or other entity should be optimized. In turn, new flows ( e.g., resulting from seek requests in media, new media requests, resume after pause, etc.) determined to be associated with the entity may be optimized. The network controller 140 uses the net-work state as well as historical traffic data in its decision for monitoring and optimization. Knowledge on the current net-work state, such as congestion, deems critical when it comes to data optimization.")<br><br>Swenson at [0029] ("As a flow is sent to the network controller 140 for inspection, historical network traffic data stored at the net-work controller 140 may be searched. The historical network traffic data includes information such as subscriber informa-tion, the cell towers to which the user devices attached, rout-ers through which the traffic is passing, geography regions, the backhaul segments, and time-of-day of the flows. For example, in a mobile network, the cell tower to which a user device is attached can be most useful, since it |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 234 of 1100

| No. | '111 Patent Claim 6 | Swenson |
|---|---|---|
| | | is the location where most congestion occurs due to limited bandwidth and high cost of the radio access network infrastructure. The network controller 140 looks into the historical traffic data for the average of the bandwidth per user at the particular cell tower. The network controller 140 can then estimate the amount ofbandwidth or degree of congestion for the new flow based on the historical record.")<br><br>Swenson at [0044] ("Additionally, historical flow data over a longer term helps the flow analyzer 312 to determine repeating patterns and heat-maps of certain network sections and to predict when they are under congestion. In this case, the flow statis-tics stored in the flow cache 322 can be mapped against traffic categories for analysis, for example, long-term running aver-ages of video flow bandwidth help determine suitability for optimization. Furthermore, estimated bandwidth per user ( or per cell-ID, per tower, or per router) over time may be metrics calculated by the flow analyzer 312 in order to determine short term needs for optimization. For example, the flow analyzer 312 may determine to being optimizing flows asso-ciated with a particular cell-ID (or those flows for identified high-bandwidth users on the cell-ID) in response to a thresh-old number of high-bandwidth users connecting to a same cell tower corresponding to the cell-ID. The reason why flow analyzer 312 selectively monitors large flows lies in the real-ization that TCP statistics for small objects, which make up most web flows, can be misleading and cause huge errors in throughput estimations.")<br><br>Swenson at [0046] ("The flow cache 322 stores monitored flow informa-tion, which is updated for a flow with each associated trans-action from the steering device 13 0. In one embodiment, data in the flow cache is stored in a map indexed by a hash, which can be up to 64-bit or longer. An entry in the flow cache map may be organized as a linked list to allow hash collisions. Alternatively, fewer bits in the hash index can also be used to speed up binary search in the flow cache map. For example, instead of using 64-bit hash index, which requires at worst 64 steps to find a node, the hash index can be reduced to 16-24 bits. There will be more hash collisions, hence the longer linked list. Other embodiments may use other type of maps or binary trees instead of the linked list to further optimize the hash collision searches.")<br><br>Swenson at [0059] ("In one embodiment, as the steering device 130 monitors network responses, it is looking for flows that match one or more signatures for video and images |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 235 of 1100

| No. | '111 Patent Claim 6 | Swenson |
|---|---|---|
| | | When a match-ing flow is detected, the steering device 130 forwards the HTTP request and a portion of the HTTP response to the network controller 140 over the ICAP client interface 404. After receiving the request and the portion of response at the ICAP server interface 406, the flow analyzer 312 of the net-work controller 140 performs a deep flow inspection to deter-mine if the flow is worth bandwidth monitoring and/or user detection. For example, the flow inspection performed by the flow analyzer 312 may determine if the flow indeed contains large or medium object ( e.g., larger than 50 kB), and/or if the source IP address of the flow is from a user or a group of users that are required to be monitored by policies. The flow ana-lyzer 312 may also determine if the flow needs to be opti-mized based on historical flow statistical data.")<br><br>Swenson at [0061] ("Once a flow is reported to the network controller 140, a flow cache entry is created for the flow in the flow cache 322. The flow cache entry keeps track of the flow and its associated bandwidth. For a flow that is marked in "continue" mode, each time the steering device 130 forwards a next portion of the flow payload to the network controller 140, the flow cache 3 22 updates the number of bytes for transmitted in the flow. By monitoring the number of bytes per flow over time, the flow analyzer 312 is capable of determining an estimate value of bandwidth associated with flow. Further-more, since the steering device 130 does not have infinite packet buffers, if congestion happens on the network link 416 from the steering device 130 to the user device 110, the TCP congestion control mechanism kicks in at the steering device 130, which may slows down and/or eventually stop receiving data over the network link 413 from origin server 160. During the congestion, the steering device 130 would not forward any data to the network controller 140, since the link 416 is congested and the network controller 140 would not be able to transmit data to the user device 110. Therefore, as an inline element, the network controller 140 can detect network con-gestions and estimate bandwidth associated with any flows of interest selected by the network controller 140. However, in the "continue" mode, the network controller 140 does not modify and transform the HTTP messaged it receives over the ICAP interface. The network controller 140 simply updates the flow statistics and returns the video or images to the steering device 130 for transmission to the user device 110.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 236 of 1100

| No. | '111 Patent Claim 7 | Swenson |
|---|---|---|
| 7 | The method according to claim 5, further comprising responsive to the packet satisfying the criterion and to the instruction, sending a portion of the packet, by the network node, to the controller. | Swenson discloses the method according to claim 5, further comprising responsive to the packet satisfying the criterion and to the instruction, sending a portion of the packet, by the network node, to the controller.<br><br>For example, Swenson discloses determining by the steering device flows that match one or more signatures or criteria of the packet. Swenson further discloses that when a matching flow is detected the steering device forwards the HTTP request and a portion of the HTTP response to the network controller.<br><br>*See supra* at Claim 5.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0028] ("In contrast to conventional inline TCP throughput monitoring devices that monitor every single data packets transmitted and received, the network controller 140 is an "out-of-band" computer server that interfaces with the steer-ing device 130 to selectively inspect user flows of interest. The network controller 140 may further identify user flows (e.g., among the flows of interest) for optimization. In one embodiment, the network controller 140 may be imple-mented at the steering device 130 to monitor traffic. In other embodiments, the network controller 140 is coupled to and communicates with the steering device 130 for traffic moni-toring and optimization. When queried by the steering device 130, the network controller 140 determines if a given network flow should be ignored, monitored further or optimized. Opti-mization of a flow is often decided at the beginning of the flow because it is rarely possible to switch to optimized content mid-stream once non-optimized content delivery has begun. However, the network controller 140 may |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 237 of 1100

| No. | '111 Patent Claim 7 | Swenson |
|---|---|---|
| | | determine that existing flows associated with a particular subscriber or other entity should be optimized. In turn, new flows ( e.g., resulting from seek requests in media, new media requests, resume after pause, etc.) determined to be associated with the entity may be optimized. The network controller 140 uses the net-work state as well as historical traffic data in its decision for monitoring and optimization. Knowledge on the current net-work state, such as congestion, deems critical when it comes to data optimization.")<br><br>Swenson at [0045] ("The steering device interface 316 interacts with an external routing appliance, such as the steering device 130 to divert portions of the network traffic ( e.g., large object net-work flows). Existing routing appliances in most carrier net-works are designed to handle large amounts of network traf-fic. They are not, however, ideal devices to operate for monitoring and analysis individual flows. Through the steer-ing device interface 316, the network controller 140 may communicate with the external routing appliances, such as the steering device 130, to steer a portion of network traffic to the network controller 140 when certain conditions are met. Generally, network flows of interest to the network controller 140 contain larger media objects, such as videos and images. In one embodiment, the smaller flows, such as web page and text information, are not exchanged over the steering device interface 316.")<br><br>Swenson at [0059] ("In one embodiment, as the steering device 130 monitors network responses, it is looking for flows that match one or more signatures for video and images. When a match-ing flow is detected, the steering device 130 forwards the HTTP request and a portion of the HTTP response to the network controller 140 over the ICAP client interface 404. After receiving the request and the portion of response at the ICAP server interface 406, the flow analyzer 312 of the net-work controller 140 performs a deep flow inspection to deter-mine if the flow is worth bandwidth monitoring and/or user detection. For example, the flow inspection performed by the flow analyzer 312 may determine if the flow indeed contains large or medium object ( e.g., larger than 50 kB), and/or if the source IP address of the flow is from a user or a group of users that are required to be monitored by policies. The flow ana-lyzer 312 may also determine if the flow needs to be opti-mized based on historical flow statistical data.")<br><br>Swenson at [0061] ("Once a flow is reported to the network controller 140, a flow cache entry is created for the flow in the flow cache 322. The flow cache entry keeps track of the |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 238 of 1100

| No. | ’111 Patent Claim 7 | Swenson |
|---|---|---|
| | | flow and its associated bandwidth. For a flow that is marked in "continue" mode, each time the steering device 130 forwards a next portion of the flow payload to the network controller 140, the flow cache 3 22 updates the number of bytes for transmitted in the flow. By monitoring the number of bytes per flow over time, the flow analyzer 312 is capable of determining an estimate value of bandwidth associated with flow. Further-more, since the steering device 130 does not have infinite packet buffers, if congestion happens on the network link 416 from the steering device 130 to the user device 110, the TCP congestion control mechanism kicks in at the steering device 130, which may slows down and/or eventually stop receiving data over the network link 413 from origin server 160. During the congestion, the steering device 130 would not forward any data to the network controller 140, since the link 416 is congested and the network controller 140 would not be able to transmit data to the user device 110. Therefore, as an inline element, the network controller 140 can detect network con-gestions and estimate bandwidth associated with any flows of interest selected by the network controller 140. However, in the "continue" mode, the network controller 140 does not modify and transform the HTTP messaged it receives over the ICAP interface. The network controller 140 simply updates the flow statistics and returns the video or images to the steering device 130 for transmission to the user device 110.”)<br><br>Swenson at [0065] (“FIG. 5 is a block diagram illustrating an example event trace of "continue" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 512 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 512 to the intended origin server 160 and receives a response 514 back from the origin server 160. The steering device 130 then sends an ICAP request message 516 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 518. Upon receiving the instruc-tion, the steering device 130 re-writes the response 514 to an HTTP redirect response 520, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 520 directly to the user device 110. In case the flow dose not contain |

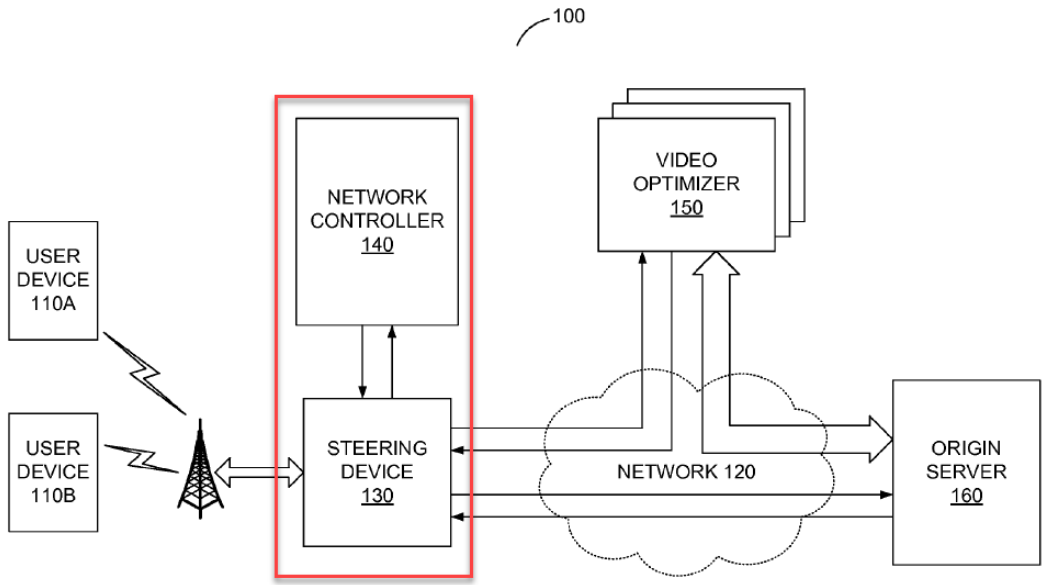


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 239 of 1100

| No. | '111 Patent Claim 7 | Swenson |
|---|---|---|
| | | video or image objects, or the network controller 140 determines not to monitor the flow, the steering device 13 0 would forward the response to the user device 110.")<br><br>Swenson at [0069] ("FIG. 6 is a block diagram illustrating an example event trace of "counting" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 612 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 612 to the intended origin server 160 and receives a response 614 back from the origin server 160. The steering device 130 then sends an ICAP request message 616 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 618. Upon receiving the instruc-tion, the steering device 130 re-writes the response 614 to an HTTP redirect response 620, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 620 directly to the user device 110. In case the flow dose not contain video or image objects that need to be redirected, the steering device 130 would forward the response to the user device 110.")<br><br>Swenson at Figure 1 (annotation added) |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 240 of 1100

| No. | '111 Patent Claim 7 | Swenson |
|---|---|---|
| | |  <br>FIG. 1 |

| No. | '111 Patent Claim 8 | Swenson |
|---|---|---|
| 8[a] | The method according to claim 7, wherein the portion of the packet consists of multiple consecutive bytes, and | Swenson discloses the method according to claim 7, wherein the portion of the packet consists of multiple consecutive bytes.<br><br>For example, Swenson discloses determining by the steering device flows that match one or more signatures or criteria of the packet. Swenson further discloses that when a matching flow is detected the steering device forwards the a next portion of the flow payload to the network controller that consists of a number of bytes.<br><br>*See supra* at Claim 7. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 241 of 1100

| No. | '111 Patent Claim 8 | Swenson |
|---|---|---|
| | | Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0028] ("In contrast to conventional inline TCP throughput monitoring devices that monitor every single data packets transmitted and received, the network controller 140 is an "out-of-band" computer server that interfaces with the steer-ing device 130 to selectively inspect user flows of interest. The network controller 140 may further identify user flows (e.g., among the flows of interest) for optimization. In one embodiment, the network controller 140 may be imple-mented at the steering device 130 to monitor traffic. In other embodiments, the network controller 140 is coupled to and communicates with the steering device 130 for traffic moni-toring and optimization. When queried by the steering device 130, the network controller 140 determines if a given network flow should be ignored, monitored further or optimized. Opti-mization of a flow is often decided at the beginning of the flow because it is rarely possible to switch to optimized content mid-stream once non-optimized content delivery has begun. However, the network controller 140 may determine that existing flows associated with a particular subscriber or other entity should be optimized. In turn, new flows ( e.g., resulting from seek requests in media, new media requests, resume after pause, etc.) determined to be associated with the entity may be optimized. The network controller 140 uses the net-work state as well as historical traffic data in its decision for monitoring and optimization. Knowledge on the current net-work state, such as congestion, deems critical when it comes to data optimization.")<br><br>Swenson at [0061] ("Once a flow is reported to the network controller 140, a flow cache entry is created for the flow in the flow cache 322. The flow cache entry keeps track of the flow and its associated bandwidth. For a flow that is marked in "continue" mode, each time the steering device 130 forwards a next portion of the flow payload to the network controller |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 242 of 1100

| No. | '111 Patent Claim 8 | Swenson |
|---|---|---|
| | | 140, the flow cache 3 22 updates the number of bytes for transmitted in the flow. By monitoring the number of bytes per flow over time, the flow analyzer 312 is capable of determining an estimate value of bandwidth associated with flow. Further-more, since the steering device 130 does not have infinite packet buffers, if congestion happens on the network link 416 from the steering device 130 to the user device 110, the TCP congestion control mechanism kicks in at the steering device 130, which may slows down and/or eventually stop receiving data over the network link 413 from origin server 160. During the congestion, the steering device 130 would not forward any data to the network controller 140, since the link 416 is congested and the network controller 140 would not be able to transmit data to the user device 110. Therefore, as an inline element, the network controller 140 can detect network con-gestions and estimate bandwidth associated with any flows of interest selected by the network controller 140. However, in the "continue" mode, the network controller 140 does not modify and transform the HTTP messaged it receives over the ICAP interface. The network controller 140 simply updates the flow statistics and returns the video or images to the steering device 130 for transmission to the user device 110.")<br><br>Swenson at [0065] ("FIG. 5 is a block diagram illustrating an example event trace of "continue" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 512 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 512 to the intended origin server 160 and receives a response 514 back from the origin server 160. The steering device 130 then sends an ICAP request message 516 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 518. Upon receiving the instruc-tion, the steering device 130 re-writes the response 514 to an HTTP redirect response 520, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 520 directly to the user device 110. In case the flow dose not contain video or image objects, or the network controller 140 determines not to monitor the flow, the steering device 13 0 would forward the response to the user device 110.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 243 of 1100

| No. | '111 Patent Claim 8 | Swenson |
|---|---|---|
| | | Swenson at [0069] ("FIG. 6 is a block diagram illustrating an example event trace of "counting" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 612 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 612 to the intended origin server 160 and receives a response 614 back from the origin server 160. The steering device 130 then sends an ICAP request message 616 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 618. Upon receiving the instruc-tion, the steering device 130 re-writes the response 614 to an HTTP redirect response 620, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 620 directly to the user device 110. In case the flow dose not contain video or image objects that need to be redirected, the steering device 130 would forward the response to the user device 110.") |
| 8[b] | wherein the instruction comprises identification of the consecutive bytes in the packet. | Swenson discloses wherein the instruction comprises identification of the consecutive bytes in the packet.<br><br>For example, Swenson discloses determining by the steering device flows that match one or more signatures or criteria of the packet. Swenson further discloses that when a matching flow is detected the steering device identifies and forwards a next portion of the flow payload to the network controller that consists of a number of bytes.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 244 of 1100

| No. | '111 Patent Claim 8 | Swenson |
|---|---|---|
| | | network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0028] ("In contrast to conventional inline TCP throughput monitoring devices that monitor every single data packets transmitted and received, the network controller 140 is an "out-of-band" computer server that interfaces with the steer-ing device 130 to selectively inspect user flows of interest. The network controller 140 may further identify user flows (e.g., among the flows of interest) for optimization. In one embodiment, the network controller 140 may be imple-mented at the steering device 130 to monitor traffic. In other embodiments, the network controller 140 is coupled to and communicates with the steering device 130 for traffic moni-toring and optimization. When queried by the steering device 130, the network controller 140 determines if a given network flow should be ignored, monitored further or optimized. Opti-mization of a flow is often decided at the beginning of the flow because it is rarely possible to switch to optimized content mid-stream once non-optimized content delivery has begun. However, the network controller 140 may determine that existing flows associated with a particular subscriber or other entity should be optimized. In turn, new flows ( e.g., resulting from seek requests in media, new media requests, resume after pause, etc.) determined to be associated with the entity may be optimized. The network controller 140 uses the net-work state as well as historical traffic data in its decision for monitoring and optimization. Knowledge on the current net-work state, such as congestion, deems critical when it comes to data optimization.")<br><br>Swenson at [0061] ("Once a flow is reported to the network controller 140, a flow cache entry is created for the flow in the flow cache 322. The flow cache entry keeps track of the flow and its associated bandwidth. For a flow that is marked in "continue" mode, each time the steering device 130 forwards a next portion of the flow payload to the network controller 140, the flow cache 3 22 updates the number of bytes for transmitted in the flow. By monitoring the number of bytes per flow over time, the flow analyzer 312 is capable of determining an estimate value of bandwidth associated with flow. Further-more, since the steering device 130 does not have infinite packet buffers, if congestion happens on the network link 416 from the steering device 130 to the user device 110, the TCP congestion control mechanism kicks in at the steering device 130, which may slows down and/or eventually stop receiving data over the network link 413 from origin server 160. During the |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 245 of 1100

| No. | '111 Patent Claim 8 | Swenson |
|---|---|---|
| | | congestion, the steering device 130 would not forward any data to the network controller 140, since the link 416 is congested and the network controller 140 would not be able to transmit data to the user device 110. Therefore, as an inline element, the network controller 140 can detect network con-gestions and estimate bandwidth associated with any flows of interest selected by the network controller 140. However, in the "continue" mode, the network controller 140 does not modify and transform the HTTP messaged it receives over the ICAP interface. The network controller 140 simply updates the flow statistics and returns the video or images to the steering device 130 for transmission to the user device 110.")<br><br>Swenson at [0065] ("FIG. 5 is a block diagram illustrating an example event trace of "continue" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 512 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 512 to the intended origin server 160 and receives a response 514 back from the origin server 160. The steering device 130 then sends an ICAP request message 516 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 518. Upon receiving the instruc-tion, the steering device 130 re-writes the response 514 to an HTTP redirect response 520, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 520 directly to the user device 110. In case the flow dose not contain video or image objects, or the network controller 140 determines not to monitor the flow, the steering device 13 0 would forward the response to the user device 110.")<br><br>Swenson at [0069] ("FIG. 6 is a block diagram illustrating an example event trace of "counting" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 612 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 612 to the |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 246 of 1100

| No. | '111 Patent Claim 8 | Swenson |
|---|---|---|
| | | intended origin server 160 and receives a response 614 back from the origin server 160. The steering device 130 then sends an ICAP request message 616 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 618. Upon receiving the instruc-tion, the steering device 130 re-writes the response 614 to an HTTP redirect response 620, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 620 directly to the user device 110. In case the flow dose not contain video or image objects that need to be redirected, the steering device 130 would forward the response to the user device 110.") |

| No. | '111 Patent Claim 9 | Swenson |
|---|---|---|
| 9 | The method according to claim 5, further comprising responsive to receiving the packet, analyzing the packet, by the controller. | Swenson discloses the method according to claim 5, further comprising responsive to receiving the packet, analyzing the packet, by the controller.<br><br>For example, Swenson discloses the network controller comprising a flow analyzer for analyzing and inspecting the packet.<br><br>*See supra* at Claim 5.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 247 of 1100

| No. | '111 Patent Claim 9 | Swenson |
|---|---|---|
| | | Swenson at [0028] ("In contrast to conventional inline TCP throughput monitoring devices that monitor every single data packets transmitted and received, the network controller 140 is an "out-of-band" computer server that interfaces with the steer-ing device 130 to selectively inspect user flows of interest. The network controller 140 may further identify user flows (e.g., among the flows of interest) for optimization. In one embodiment, the network controller 140 may be imple-mented at the steering device 130 to monitor traffic. In other embodiments, the network controller 140 is coupled to and communicates with the steering device 130 for traffic moni-toring and optimization. When queried by the steering device 130, the network controller 140 determines if a given network flow should be ignored, monitored further or optimized. Opti-mization of a flow is often decided at the beginning of the flow because it is rarely possible to switch to optimized content mid-stream once non-optimized content delivery has begun. However, the network controller 140 may determine that existing flows associated with a particular subscriber or other entity should be optimized. In turn, new flows ( e.g., resulting from seek requests in media, new media requests, resume after pause, etc.) determined to be associated with the entity may be optimized. The network controller 140 uses the net-work state as well as historical traffic data in its decision for monitoring and optimization. Knowledge on the current net-work state, such as congestion, deems critical when it comes to data optimization.")<br><br>Swenson at [0038] ("Turning back to FIG. 1, the network controller 140 allows network operators to apply fine granular optimization policies to ensure high quality of experience (QoE) based on cell tower congestion, device types, subscriber profiles and service plans with lower hardware and software costs. The architecture of the network controller 140 provides an excel-lent fit for the net neutrality guideline of"reasonable network management", and better compliance to the copyright law (DMCA) than solutions that rely on long-term caching. Hav-ing the ability of monitoring network traffic on a per sub-scriber, per flow, or per video file basis, the network controller 140 also selectively monitors and optimizes only a subset of traffic that benefits from optimization the most, thus achiev-ing both scalability and efficiency for optimization at a com-petitive price-point. The core element of the network control-ler 140 lies in its mechanisms for congestion detection and mitigation, which allows optimization resources to be utilized in the most efficient and surgical manner.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 248 of 1100

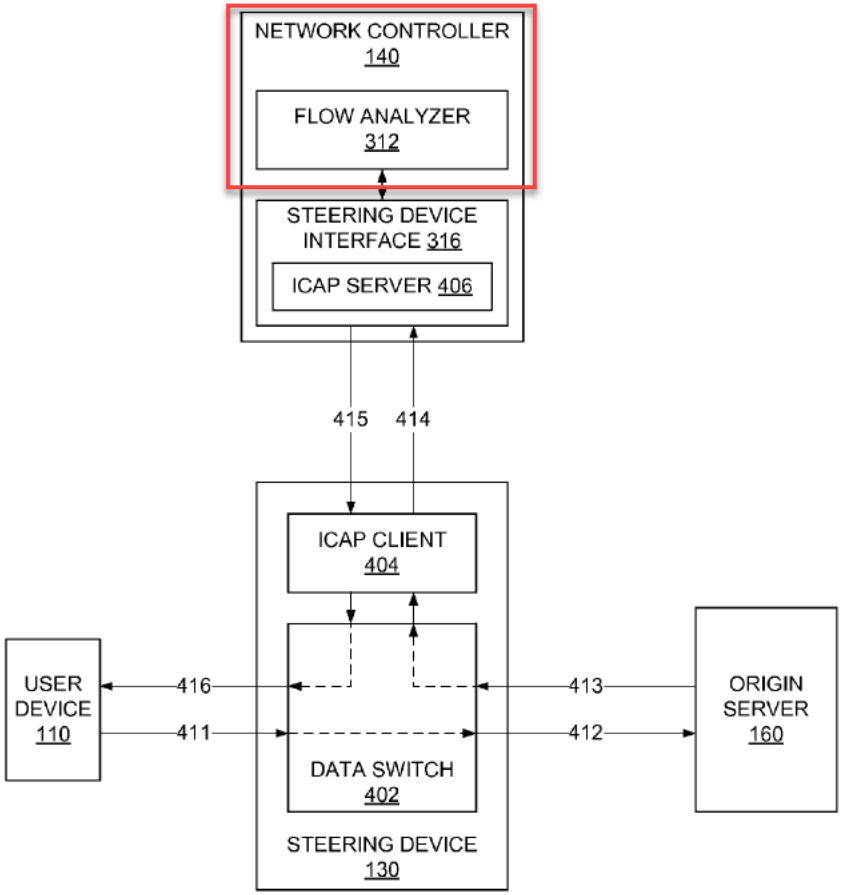
| No. | '111 Patent Claim 9 | Swenson |
|---|---|---|
| | | Swenson at [0039] ("Referring now to FIG. 3, it illustrates one embodi-ment of an example architecture of the network controller 140 for providing selective real-time network monitoring and subscriber identification. The network controller 140 com-prises a flow analyzer 312, a policy engine 314, a steering device interface 316, a video optimizer redirector 318, a flow cache 322, and a subscriber log 324. In other embodiments, the network controller 140 may include additional, fewer, or different components for various applications. Conventional components such as network interfaces, security functions, failover servers, management and network operations con-soles, and the like are not shown so as to not obscure the details of the system architecture.")<br><br>Swenson at [0059] ("In one embodiment, as the steering device 130 monitors network responses, it is looking for flows that match one or more signatures for video and images. When a match-ing flow is detected, the steering device 130 forwards the HTTP request and a portion of the HTTP response to the network controller 140 over the ICAP client interface 404. After receiving the request and the portion of response at the ICAP server interface 406, the flow analyzer 312 of the net-work controller 140 performs a deep flow inspection to deter-mine if the flow is worth bandwidth monitoring and/or user detection. For example, the flow inspection performed by the flow analyzer 312 may determine if the flow indeed contains large or medium object ( e.g., larger than 50 kB), and/or if the source IP address of the flow is from a user or a group of users that are required to be monitored by policies. The flow ana-lyzer 312 may also determine if the flow needs to be opti-mized based on historical flow statistical data.")<br><br>Swenson at [0065] ("FIG. 5 is a block diagram illustrating an example event trace of "continue" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 512 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 512 to the intended origin server 160 and receives a response 514 back from the origin server 160. The steering device 130 then sends an ICAP request message 516 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 249 of 1100

| No. | '111 Patent Claim 9 | Swenson |
|---|---|---|
| | | response 518. Upon receiving the instruc-tion, the steering device 130 re-writes the response 514 to an HTTP redirect response 520, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 520 directly to the user device 110. In case the flow dose not contain video or image objects, or the network controller 140 determines not to monitor the flow, the steering device 13 0 would forward the response to the user device 110.")<br><br>Swenson at [0069] ("FIG. 6 is a block diagram illustrating an example event trace of "counting" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 612 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 612 to the intended origin server 160 and receives a response 614 back from the origin server 160. The steering device 130 then sends an ICAP request message 616 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 618. Upon receiving the instruc-tion, the steering device 130 re-writes the response 614 to an HTTP redirect response 620, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 620 directly to the user device 110. In case the flow dose not contain video or image objects that need to be redirected, the steering device 130 would forward the response to the user device 110.")<br><br>Swenson at Figure 4A (annotation added) |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 250 of 1100

| No. | '111 Patent Claim 9 | Swenson |
|---|---|---|
| | |  <br>FIG. 4A |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 251 of 1100

| No. | '111 Patent Claim 12 | Swenson |
|---|---|---|
| 12 | The method according to claim 9, wherein the analyzing comprises applying security or data analytic application. | Swenson discloses the method according to claim 9, wherein the analyzing comprises applying security or data analytic application.<br><br>For example, Swenson discloses the network controller comprising a flow analyzer for analyzing and inspecting the packet. Swenson further discloses other conventional components such as security functions are included in the network controller.<br><br>*See supra* at Claim 9.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0028] ("In contrast to conventional inline TCP throughput monitoring devices that monitor every single data packets transmitted and received, the network controller 140 is an "out-of-band" computer server that interfaces with the steer-ing device 130 to selectively inspect user flows of interest. The network controller 140 may further identify user flows (e.g., among the flows of interest) for optimization. In one embodiment, the network controller 140 may be imple-mented at the steering device 130 to monitor traffic. In other embodiments, the network controller 140 is coupled to and communicates with the steering device 130 for traffic moni-toring and optimization. When queried by the steering device 130, the network controller 140 determines if a given network flow should be ignored, monitored further or optimized. Opti-mization of a flow is often decided at the beginning of the flow because it is rarely possible to switch to optimized content mid-stream once non-optimized content delivery has begun. However, the network controller 140 may determine that existing flows associated with a particular subscriber or other entity should |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 252 of 1100

| No. | '111 Patent Claim 12 | Swenson |
|---|---|---|
| | | be optimized. In turn, new flows ( e.g., resulting from seek requests in media, new media requests, resume after pause, etc.) determined to be associated with the entity may be optimized. The network controller 140 uses the net-work state as well as historical traffic data in its decision for monitoring and optimization. Knowledge on the current net-work state, such as congestion, deems critical when it comes to data optimization.")<br><br>Swenson at [0038] ("Turning back to FIG. 1, the network controller 140 allows network operators to apply fine granular optimization policies to ensure high quality of experience (QoE) based on cell tower congestion, device types, subscriber profiles and service plans with lower hardware and software costs. The architecture of the network controller 140 provides an excel-lent fit for the net neutrality guideline of"reasonable network management", and better compliance to the copyright law (DMCA) than solutions that rely on long-term caching. Hav-ing the ability of monitoring network traffic on a per sub-scriber, per flow, or per video file basis, the network controller 140 also selectively monitors and optimizes only a subset of traffic that benefits from optimization the most, thus achiev-ing both scalability and efficiency for optimization at a com-petitive price-point. The core element of the network control-ler 140 lies in its mechanisms for congestion detection and mitigation, which allows optimization resources to be utilized in the most efficient and surgical manner.")<br><br>Swenson at [0039] ("Referring now to FIG. 3, it illustrates one embodi-ment of an example architecture of the network controller 140 for providing selective real-time network monitoring and subscriber identification. The network controller 140 com-prises a flow analyzer 312, a policy engine 314, a steering device interface 316, a video optimizer redirector 318, a flow cache 322, and a subscriber log 324. In other embodiments, the network controller 140 may include additional, fewer, or different components for various applications. Conventional components such as network interfaces, security functions, failover servers, management and network operations con-soles, and the like are not shown so as to not obscure the details of the system architecture.")<br><br>Swenson at [0059] ("In one embodiment, as the steering device 130 monitors network responses, it is looking for flows that match one or more signatures for video and images. When a match-ing flow is detected, the steering device 130 forwards the HTTP request and a portion of the HTTP response to the network controller 140 over the ICAP client interface. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 253 of 1100

| No. | '111 Patent Claim 12 | Swenson |
|---|---|---|
| | | 404. After receiving the request and the portion of response at the ICAP server interface 406, the flow analyzer 312 of the net-work controller 140 performs a deep flow inspection to deter-mine if the flow is worth bandwidth monitoring and/or user detection. For example, the flow inspection performed by the flow analyzer 312 may determine if the flow indeed contains large or medium object ( e.g., larger than 50 kB), and/or if the source IP address of the flow is from a user or a group of users that are required to be monitored by policies. The flow ana-lyzer 312 may also determine if the flow needs to be opti-mized based on historical flow statistical data.")<br><br>Swenson at [0065] ("FIG. 5 is a block diagram illustrating an example event trace of "continue" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 512 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 512 to the intended origin server 160 and receives a response 514 back from the origin server 160. The steering device 130 then sends an ICAP request message 516 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 518. Upon receiving the instruc-tion, the steering device 130 re-writes the response 514 to an HTTP redirect response 520, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 520 directly to the user device 110. In case the flow dose not contain video or image objects, or the network controller 140 determines not to monitor the flow, the steering device 13 0 would forward the response to the user device 110.")<br><br>Swenson at [0069] ("FIG. 6 is a block diagram illustrating an example event trace of "counting" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 612 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 612 to the intended origin server 160 and receives a response 614 back from the origin server 160. The |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 254 of 1100

| No. | '111 Patent Claim 12 | Swenson |
|---|---|---|
| | | steering device 130 then sends an ICAP request message 616 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 618. Upon receiving the instruc-tion, the steering device 130 re-writes the response 614 to an HTTP redirect response 620, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 620 directly to the user device 110. In case the flow dose not contain video or image objects that need to be redirected, the steering device 130 would forward the response to the user device 110.")<br><br>Swenson at Figure 4A (annotation added) |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 255 of 1100

| No. | '111 Patent Claim 12 | Swenson |
|---|---|---|

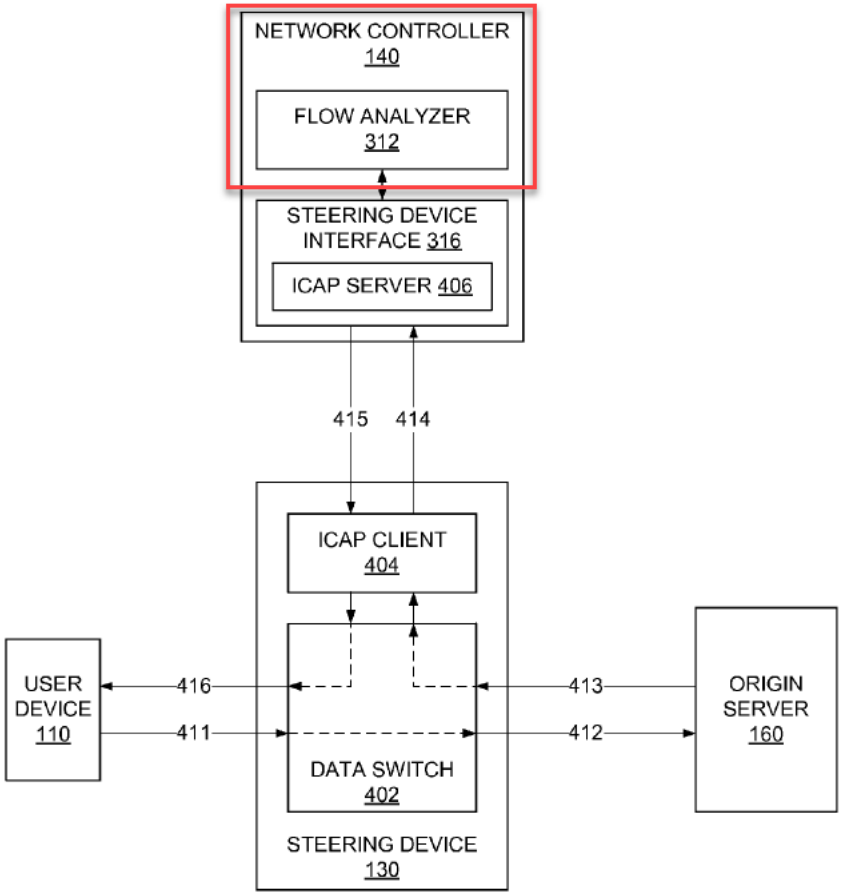



FIG. 4A



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 256 of 1100

| No. | '111 Patent Claim 13 | Swenson |
|---|---|---|
| 13 | The method according to claim 9, wherein the analyzing comprises applying security application that comprises firewall or intrusion detection functionality. | Swenson discloses the method according to claim 9, wherein the analyzing comprises applying security application that comprises firewall or intrusion detection functionality.<br><br>For example, Swenson discloses the network controller comprising a flow analyzer for analyzing and inspecting the packet. Swenson further discloses other conventional component such as security functions, which may be a firewall or intrusion detection engine, are included in the network controller.<br><br>*See supra* at Claim 9.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0028] ("In contrast to conventional inline TCP throughput monitoring devices that monitor every single data packets transmitted and received, the network controller 140 is an "out-of-band" computer server that interfaces with the steer-ing device 130 to selectively inspect user flows of interest. The network controller 140 may further identify user flows (e.g., among the flows of interest) for optimization. In one embodiment, the network controller 140 may be imple-mented at the steering device 130 to monitor traffic. In other embodiments, the network controller 140 is coupled to and communicates with the steering device 130 for traffic moni-toring and optimization. When queried by the steering device 130, the network controller 140 determines if a given network flow should be ignored, monitored further or optimized. Opti-mization of a flow is often decided at the beginning of the flow because it is rarely possible to switch to optimized content mid-stream once non-optimized content delivery has begun. However, the network controller 140 may |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 257 of 1100

| No. | '111 Patent Claim 13 | Swenson |
|---|---|---|
| | | determine that existing flows associated with a particular subscriber or other entity should be optimized. In turn, new flows ( e.g., resulting from seek requests in media, new media requests, resume after pause, etc.) determined to be associated with the entity may be optimized. The network controller 140 uses the net-work state as well as historical traffic data in its decision for monitoring and optimization. Knowledge on the current net-work state, such as congestion, deems critical when it comes to data optimization.")<br><br>Swenson at [0038] ("Turning back to FIG. 1, the network controller 140 allows network operators to apply fine granular optimization policies to ensure high quality of experience (QoE) based on cell tower congestion, device types, subscriber profiles and service plans with lower hardware and software costs. The architecture of the network controller 140 provides an excel-lent fit for the net neutrality guideline of"reasonable network management", and better compliance to the copyright law (DMCA) than solutions that rely on long-term caching. Hav-ing the ability of monitoring network traffic on a per sub-scriber, per flow, or per video file basis, the network controller 140 also selectively monitors and optimizes only a subset of traffic that benefits from optimization the most, thus achiev-ing both scalability and efficiency for optimization at a com-petitive price-point. The core element of the network control-ler 140 lies in its mechanisms for congestion detection and mitigation, which allows optimization resources to be utilized in the most efficient and surgical manner.")<br><br>Swenson at [0039] ("Referring now to FIG. 3, it illustrates one embodi-ment of an example architecture of the network controller 140 for providing selective real-time network monitoring and subscriber identification. The network controller 140 com-prises a flow analyzer 312, a policy engine 314, a steering device interface 316, a video optimizer redirector 318, a flow cache 322, and a subscriber log 324. In other embodiments, the network controller 140 may include additional, fewer, or different components for various applications. Conventional components such as network interfaces, security functions, failover servers, management and network operations con-soles, and the like are not shown so as to not obscure the details of the system architecture.")<br><br>Swenson at [0059] ("In one embodiment, as the steering device 130 monitors network responses, it is looking for flows that match one or more signatures for video and images. When a match-ing flow is detected, the steering device 130 forwards the HTTP request and |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 258 of 1100

| No. | '111 Patent Claim 13 | Swenson |
|---|---|---|
| | | a portion of the HTTP response to the network controller 140 over the ICAP client interface 404. After receiving the request and the portion of response at the ICAP server interface 406, the flow analyzer 312 of the net-work controller 140 performs a deep flow inspection to deter-mine if the flow is worth bandwidth monitoring and/or user detection. For example, the flow inspection performed by the flow analyzer 312 may determine if the flow indeed contains large or medium object ( e.g., larger than 50 kB), and/or if the source IP address of the flow is from a user or a group of users that are required to be monitored by policies. The flow ana-lyzer 312 may also determine if the flow needs to be opti-mized based on historical flow statistical data.")<br><br>Swenson at [0065] ("FIG. 5 is a block diagram illustrating an example event trace of "continue" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 512 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 512 to the intended origin server 160 and receives a response 514 back from the origin server 160. The steering device 130 then sends an ICAP request message 516 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 518. Upon receiving the instruc-tion, the steering device 130 re-writes the response 514 to an HTTP redirect response 520, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 520 directly to the user device 110. In case the flow dose not contain video or image objects, or the network controller 140 determines not to monitor the flow, the steering device 13 0 would forward the response to the user device 110.")<br><br>Swenson at [0069] ("FIG. 6 is a block diagram illustrating an example event trace of "counting" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 612 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 612 to the |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 259 of 1100

| No. | '111 Patent Claim 13 | Swenson |
|---|---|---|
| | | intended origin server 160 and receives a response 614 back from the origin server 160. The steering device 130 then sends an ICAP request message 616 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 618. Upon receiving the instruc-tion, the steering device 130 re-writes the response 614 to an HTTP redirect response 620, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 620 directly to the user device 110. In case the flow dose not contain video or image objects that need to be redirected, the steering device 130 would forward the response to the user device 110.") |

| No. | '111 Patent Claim 14 | Swenson |
|---|---|---|
| 14 | The method according to claim 9, wherein the analyzing comprises performing Deep Packet Inspection (DPI) or using a DPI engine on the packet. | Swenson discloses the method according to claim 9, wherein the analyzing comprises performing Deep Packet Inspection (DPI) or using a DPI engine on the packet.<br><br>For example, Swenson discloses the network controller comprising a flow analyzer performing a deep flow inspection on the packet flow.<br><br>*See supra* at Claim 9.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 260 of 1100

| No. | '111 Patent Claim 14 | Swenson |
|---|---|---|
| | | Swenson at [0028] ("In contrast to conventional inline TCP throughput monitoring devices that monitor every single data packets transmitted and received, the network controller 140 is an "out-of-band" computer server that interfaces with the steer-ing device 130 to selectively inspect user flows of interest. The network controller 140 may further identify user flows (e.g., among the flows of interest) for optimization. In one embodiment, the network controller 140 may be imple-mented at the steering device 130 to monitor traffic. In other embodiments, the network controller 140 is coupled to and communicates with the steering device 130 for traffic moni-toring and optimization. When queried by the steering device 130, the network controller 140 determines if a given network flow should be ignored, monitored further or optimized. Opti-mization of a flow is often decided at the beginning of the flow because it is rarely possible to switch to optimized content mid-stream once non-optimized content delivery has begun. However, the network controller 140 may determine that existing flows associated with a particular subscriber or other entity should be optimized. In turn, new flows ( e.g., resulting from seek requests in media, new media requests, resume after pause, etc.) determined to be associated with the entity may be optimized. The network controller 140 uses the net-work state as well as historical traffic data in its decision for monitoring and optimization. Knowledge on the current net-work state, such as congestion, deems critical when it comes to data optimization.")<br><br>Swenson at [0038] ("Turning back to FIG. 1, the network controller 140 allows network operators to apply fine granular optimization policies to ensure high quality of experience (QoE) based on cell tower congestion, device types, subscriber profiles and service plans with lower hardware and software costs. The architecture of the network controller 140 provides an excel-lent fit for the net neutrality guideline of"reasonable network management", and better compliance to the copyright law (DMCA) than solutions that rely on long-term caching. Hav-ing the ability of monitoring network traffic on a per sub-scriber, per flow, or per video file basis, the network controller 140 also selectively monitors and optimizes only a subset of traffic that benefits from optimization the most, thus achiev-ing both scalability and efficiency for optimization at a com-petitive price-point. The core element of the network control-ler 140 lies in its mechanisms for congestion detection and mitigation, which allows optimization resources to be utilized in the most efficient and surgical manner.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 261 of 1100

| No. | '111 Patent Claim 14 | Swenson |
|---|---|---|
| | | Swenson at [0039] ("Referring now to FIG. 3, it illustrates one embodi-ment of an example architecture of the network controller 140 for providing selective real-time network monitoring and subscriber identification. The network controller 140 com-prises a flow analyzer 312, a policy engine 314, a steering device interface 316, a video optimizer redirector 318, a flow cache 322, and a subscriber log 324. In other embodiments, the network controller 140 may include additional, fewer, or different components for various applications. Conventional components such as network interfaces, security functions, failover servers, management and network operations con-soles, and the like are not shown so as to not obscure the details of the system architecture.")<br><br>Swenson at [0059] ("In one embodiment, as the steering device 130 monitors network responses, it is looking for flows that match one or more signatures for video and images. When a match-ing flow is detected, the steering device 130 forwards the HTTP request and a portion of the HTTP response to the network controller 140 over the ICAP client interface 404. After receiving the request and the portion of response at the ICAP server interface 406, the flow analyzer 312 of the net-work controller 140 performs a deep flow inspection to deter-mine if the flow is worth bandwidth monitoring and/or user detection. For example, the flow inspection performed by the flow analyzer 312 may determine if the flow indeed contains large or medium object ( e.g., larger than 50 kB), and/or if the source IP address of the flow is from a user or a group of users that are required to be monitored by policies. The flow ana-lyzer 312 may also determine if the flow needs to be opti-mized based on historical flow statistical data.")<br><br>Swenson at [0065] ("FIG. 5 is a block diagram illustrating an example event trace of "continue" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 512 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 512 to the intended origin server 160 and receives a response 514 back from the origin server 160. The steering device 130 then sends an ICAP request message 516 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 262 of 1100

| No. | '111 Patent Claim 14 | Swenson |
|---|---|---|
| | | response 518. Upon receiving the instruc-tion, the steering device 130 re-writes the response 514 to an HTTP redirect response 520, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 520 directly to the user device 110. In case the flow dose not contain video or image objects, or the network controller 140 determines not to monitor the flow, the steering device 13 0 would forward the response to the user device 110.")<br>Swenson at [0069] ("FIG. 6 is a block diagram illustrating an example event trace of "counting" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 612 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 612 to the intended origin server 160 and receives a response 614 back from the origin server 160. The steering device 130 then sends an ICAP request message 616 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 618. Upon receiving the instruc-tion, the steering device 130 re-writes the response 614 to an HTTP redirect response 620, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 620 directly to the user device 110. In case the flow dose not contain video or image objects that need to be redirected, the steering device 130 would forward the response to the user device 110.") |

| No. | '111 Patent Claim 15 | Swenson |
|---|---|---|
| 15[a] | The method according to claim 9, wherein the packet comprises distinct header and payload fields, and | Swenson discloses the method according to claim 9, wherein the packet comprises distinct header and payload fields.<br><br>For example, Swenson discloses packet flows with header and payload fields.<br><br>*See supra* at Claim 9. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 263 of 1100

| No. | '111 Patent Claim 15 | Swenson |
|---|---|---|
| | | Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0040] ("The flow analyzer 312 monitors large flows in the network, analyzes collected flow statistics to determine net-work throughput, and accordingly selects flows to be opti-mized. The flow analyzer 312 does not need to see all the flows in order to make an accurate estimate of network con-ditions. The flow analyzer 312 processes the traffic statistics stored in the flow cache 3 22 and user information stored in the subscriber log 324, for example, by associating network flows identified by source IP addresses to a mobile subscriber or user, which is identified by his or her current subscriber ID or device ID. The user flows are also mapped to a congestion level at the current sub-network (e.g., a cell with which the user devices are associated), so that an optimization decision can be made at the beginning of the data transmission.")<br><br>Swenson at [0049] ("The policy engine 314 defines policies for optimiz-ing large flows with media objects to mitigate network con-gestion. Detecting and acting on congestion in the network, the design focus of the network controller 140 is built on this very flexible policy engine. The policy engine 314 is capable of taking virtually any input, either deduced from HTTP headers and payload ( e.g., through RADIUS/Gx interface), or provided by the network infrastructure via API, and making decisions on how to apply optimization based on individual or a combination of these inputs. The optimization policies can be applied to large flows all the time or on a time-of-day basis, a per user basis, and/or depending on the network condition.")<br><br>Swenson at [0061] ("Once a flow is reported to the network controller 140, a flow cache entry is created for the flow in the flow cache 322. The flow cache entry keeps track of the |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 264 of 1100

| No. | ’111 Patent Claim 15 | Swenson |
|---|---|---|
| | | flow and its associated bandwidth. For a flow that is marked in "continue" mode, each time the steering device 130 forwards a next portion of the flow payload to the network controller 140, the flow cache 3 22 updates the number of bytes for transmitted in the flow. By monitoring the number of bytes per flow over time, the flow analyzer 312 is capable of determining an estimate value of bandwidth associated with flow. Further-more, since the steering device 130 does not have infinite packet buffers, if congestion happens on the network link 416 from the steering device 130 to the user device 110, the TCP congestion control mechanism kicks in at the steering device 130, which may slows down and/or eventually stop receiving data over the network link 413 from origin server 160. During the congestion, the steering device 130 would not forward any data to the network controller 140, since the link 416 is congested and the network controller 140 would not be able to transmit data to the user device 110. Therefore, as an inline element, the network controller 140 can detect network con-gestions and estimate bandwidth associated with any flows of interest selected by the network controller 140. However, in the "continue" mode, the network controller 140 does not modify and transform the HTTP messaged it receives over the ICAP interface. The network controller 140 simply updates the flow statistics and returns the video or images to the steering device 130 for transmission to the user device 110.”)<br><br>Swenson at [0064] (Similar to the "continue" mode, after receiving the initial HTTP messages of a flow and determining to monitor the flow, the network controller 140 notify the steering device 130 to work in a "counting" mode for bandwidth monitoring. In contrast to the "continue" mode, when a matching flow is detected for "counting" mode, the steering device 130 for-wards the HTTP response directly to the user device 110. While at the same time, the steering device 130 send a cus-tomized ICAP message to the network controller 140 over the network link 425. In one embodiment, the customized ICAP message contains the HTTP request and response headers, as well as a count of payload size of the current flow. After updating the flow statistics, the network controller 140 may acknowledge the gateway over the network line 426. In the "counting" mode, the network controller 140 does not join the network response path as an inline network element, but simply listens to the counting of flow size. The benefit of the "counting" mode is to off-load the network controller 140 from ingesting and forwarding the network flow on the net- work response path, while still enabling the detection of con-gestions and estimation of bandwidth associated with the flows of interest.”) |


~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 265 of 1100

| No. | '111 Patent Claim 15 | Swenson |
|---|---|---|
| | | Swenson at [0065] ("FIG. 5 is a block diagram illustrating an example event trace of "continue" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 512 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 512 to the intended origin server 160 and receives a response 514 back from the origin server 160. The steering device 130 then sends an ICAP request message 516 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 518. Upon receiving the instruc-tion, the steering device 130 re-writes the response 514 to an HTTP redirect response 520, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 520 directly to the user device 110. In case the flow dose not contain video or image objects, or the network controller 140 determines not to monitor the flow, the steering device 13 0 would forward the response to the user device 110.")<br><br>Swenson at [0069] ("FIG. 6 is a block diagram illustrating an example event trace of "counting" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 612 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 612 to the intended origin server 160 and receives a response 614 back from the origin server 160. The steering device 130 then sends an ICAP request message 616 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 618. Upon receiving the instruc-tion, the steering device 130 re-writes the response 614 to an HTTP redirect response 620, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 620 directly to the user device 110. In case the flow dose not contain |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 266 of 1100

| No. | '111 Patent Claim 15 | Swenson |
|---|---|---|
| | | video or image objects that need to be redirected, the steering device 130 would forward the response to the user device 110.")<br><br>Swenson at [0073] ("FIG. 7 is a block diagram illustrating one embodi-ment of an example of internal components of the flow cache. The flow cache map 700 comprises a plurality of flow cache entries, such as flow cache entries 710 and 712 indexed by a hash. Not shown in the example diagram is a possible linked list behind each flow cache entry which allows chaining of flow cache entries for a given hash index. The hash into the flow cache may be based on source IP address, MAC address, subscriber ID, or other identifier indicative of a given sub-scriber, group of subscribers or subscriber's device.")<br><br>Swenson at [0079] ("In the bandwidth calculation, flows are categorized into buckets based on the size of the objects being transferred. Small objects may not be factored into the bandwidth calcu-lation since they may come and go within a single interval. For example, flows with payload size less than 50 kB may be ignored because a transfer of 50 kB may never reach the full potential throughput of the link. While larger flows may reach the full throughput of the link for a long period of time intervals, they are grouped into 50-75 kB, 75-100 kB and 100 kB+ buckets because the characteristics of these flow sizes can be different, hence the bandwidth for each of the buckets is measured and calculated separately. In other embodiments, the flow size ranges (e.g., 50-75 kB, 75-100 kB and l00kB+) of the buckets may be altered depending on the network traffic and size of objects transmitted. Furthermore, the bucket sizes can also be adjusted based on network topology, such as buffer size, prior to transmission to the client. The calculated bandwidth per bucket is stored in a queue structure that allows for the computing and updating of minimum, maximum, and/or average measurements for each bucket. In one embodiment, the 100 kB+ bucket's current tail entry is checked against the average bandwidth for the 100 kB+ bucket. If the current entry is less than the average multiplied by the number of entries in the queue, the current entry is added to the bandwidth calculation for the current interval. This scheme can filter out large bursts of data from tempo-rarily idle flows. If the bandwidth exceeds the value, a number of bytes (e.g., 125 kB) will be subtracted from the current entry to account for TCP buffers in the network.")<br><br>Swenson at [0083] ("When a new flow is observed, flow cache entries are searched by matching source IP address 722 if the subscriber id or other identifiers of the flow are not |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 267 of 1100

| No. | '111 Patent Claim 15 | Swenson |
|---|---|---|
| | | available. In case of multiple users sharing an IP address, the flow analyzer 312 needs to find patterns or other identifiers in the flows to map them to particular subscribers. Flows without identified sub-scribers are added to the flow cache block under the default user flows 726, which is a default holding place for the new flows. The flow analyzer 312 later will scan through the default user flows that contain cookies or other identifiers that may be used to determine a real user or subscriber associated with the flow. If a flow contains identifiers not associated with an existing real user, a new user or subscriber is created and the user flow block is moved to newly created (or mapped) user or subscriber.") |
| 15[b] | wherein the analyzing comprises checking part of, or whole of, the payload field. | Swenson discloses wherein the analyzing comprises checking part of, or whole of, the payload field.<br><br>For example, Swenson discloses defining optimization policies based on analysis of HTTP payload field of the packet by the flow analyzer of the network controller.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0040] ("The flow analyzer 312 monitors large flows in the network, analyzes collected flow statistics to determine net-work throughput, and accordingly selects flows to be opti-mized. The flow analyzer 312 does not need to see all the flows in order to make an accurate estimate of network con-ditions. The flow analyzer 312 processes the traffic statistics stored in the flow cache 3 22 and user information stored in the subscriber log 324, for example, by associating network flows identified by source IP addresses to a mobile subscriber or user, which is identified by his or her current subscriber ID or device ID. The user flows are also mapped to a congestion level at the current sub-network (e.g., a cell with |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 268 of 1100

| No. | '111 Patent Claim 15 | Swenson |
|---|---|---|
| | | which the user devices are associated), so that an optimization decision can be made at the beginning of the data transmission.")<br><br>Swenson at [0049] ("The policy engine 314 defines policies for optimiz-ing large flows with media objects to mitigate network con-gestion. Detecting and acting on congestion in the network, the design focus of the network controller 140 is built on this very flexible policy engine. The policy engine 314 is capable of taking virtually any input, either deduced from HTTP headers and payload ( e.g., through RADIUS/Gx interface), or provided by the network infrastructure via API, and making decisions on how to apply optimization based on individual or a combination of these inputs. The optimization policies can be applied to large flows all the time or on a time-of-day basis, a per user basis, and/or depending on the network condition.")<br><br>Swenson at [0061] ("Once a flow is reported to the network controller 140, a flow cache entry is created for the flow in the flow cache 322. The flow cache entry keeps track of the flow and its associated bandwidth. For a flow that is marked in "continue" mode, each time the steering device 130 forwards a next portion of the flow payload to the network controller 140, the flow cache 3 22 updates the number of bytes for transmitted in the flow. By monitoring the number of bytes per flow over time, the flow analyzer 312 is capable of determining an estimate value of bandwidth associated with flow. Further-more, since the steering device 130 does not have infinite packet buffers, if congestion happens on the network link 416 from the steering device 130 to the user device 110, the TCP congestion control mechanism kicks in at the steering device 130, which may slows down and/or eventually stop receiving data over the network link 413 from origin server 160. During the congestion, the steering device 130 would not forward any data to the network controller 140, since the link 416 is congested and the network controller 140 would not be able to transmit data to the user device 110. Therefore, as an inline element, the network controller 140 can detect network con-gestions and estimate bandwidth associated with any flows of interest selected by the network controller 140. However, in the "continue" mode, the network controller 140 does not modify and transform the HTTP messaged it receives over the ICAP interface. The network controller 140 simply updates the flow statistics and returns the video or images to the steering device 130 for transmission to the user device 110.") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 269 of 1100

| No. | '111 Patent Claim 15 | Swenson |
|---|---|---|
| | | Swenson at [0065] ("FIG. 5 is a block diagram illustrating an example event trace of "continue" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 512 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 512 to the intended origin server 160 and receives a response 514 back from the origin server 160. The steering device 130 then sends an ICAP request message 516 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 518. Upon receiving the instruc-tion, the steering device 130 re-writes the response 514 to an HTTP redirect response 520, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 520 directly to the user device 110. In case the flow dose not contain video or image objects, or the network controller 140 determines not to monitor the flow, the steering device 13 0 would forward the response to the user device 110.")<br><br>Swenson at [0069] ("FIG. 6 is a block diagram illustrating an example event trace of "counting" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 612 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 612 to the intended origin server 160 and receives a response 614 back from the origin server 160. The steering device 130 then sends an ICAP request message 616 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 618. Upon receiving the instruc-tion, the steering device 130 re-writes the response 614 to an HTTP redirect response 620, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 620 directly to the user device 110. In case the flow dose not contain |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 270 of 1100

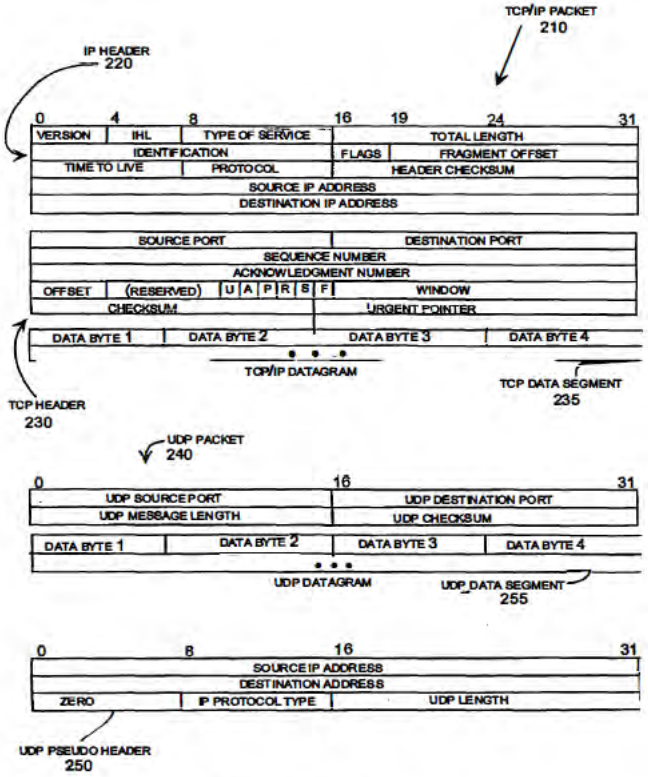
| No. | '111 Patent Claim 15 | Swenson |
|---|---|---|
| | | video or image objects that need to be redirected, the steering device 130 would forward the response to the user device 110.") |

| No. | '111 Patent Claim 16 | Swenson |
|---|---|---|
| 16[a] | The method according to claim 1, wherein the packet comprises distinct header and payload fields, | Swenson discloses the method according to claim 1, wherein the packet comprises distinct header and payload fields.<br><br>*See supra* at Claim 1, 15[a]. |
| 16[b] | the header comprises one or more flag bits, and | Swenson discloses the header comprises one or more flag bits.<br><br>For example, Swenson packet flow header fields in which may include flags, such as a "do not transcode" flag. A person of ordinary skill in the art would understand that a header may be comprised of specific flag bits. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Swenson is found to not meet this limitation the header comprises one or more flag bits would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.") |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 271 of 1100

| No. | '111 Patent Claim 16 | Swenson |
|---|---|---|
| | | Swenson at [0040] ("The flow analyzer 312 monitors large flows in the network, analyzes collected flow statistics to determine net-work throughput, and accordingly selects flows to be opti-mized. The flow analyzer 312 does not need to see all the flows in order to make an accurate estimate of network con-ditions. The flow analyzer 312 processes the traffic statistics stored in the flow cache 3 22 and user information stored in the subscriber log 324, for example, by associating network flows identified by source IP addresses to a mobile subscriber or user, which is identified by his or her current subscriber ID or device ID. The user flows are also mapped to a congestion level at the current sub-network (e.g., a cell with which the user devices are associated), so that an optimization decision can be made at the beginning of the data transmission.")<br><br>Swenson at [0068] ("In one embodiment, responsive to an HTTP get request 522 to an origin server 160, the video optimizer receives a HTTP 404 error from the origin server 160 as opposed to a video file. In such case, the video optimizer 150 appends a "do not transcode" flag to the HTTP redirect request and returned to the user device 110, which re-sends the request out over the network to the origin server 160. The origin server 160 responds appropriately to the request by sending back video 524, which is intercepted by the steering device 130 and the inline on-demand element the network controller 140 for monitoring purpose.")<br><br>Swenson at [0072] ("In one embodiment, if the video optimizer 150 failed to retrieve user requested video file from the origin server 160, the video optimizer 150 appends a "do not transcode" flag to the HTTP redirect request and returned to the user device 110, which re-sends the request out over the network to the origin server 160. The origin server 160 responds appropriately to the request by sending back video 624, which is intercepted by the steering device 130 only. The steering device 130 forwards the video to the user device 110 and at the same time reports the flow size to the network controller 140 for monitoring purpose.")<br><br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Swenson in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 16[b] of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references. |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 272 of 1100

| No. | '111 Patent Claim 16 | Swenson |
|---|---|---|
| | | For example, Copeland discloses packet headers with flag bits.<br><br>Copeland at Figure 2<br><br>  <br><br>PACKET HEADERS<br>FIG. 2<br><br>Copeland at [0076] ("FIG. 2 illustrates an exemplary TCP/IP packet or datagram 210 and an exemplary UDP datagram 240. In a typical TCP/IP packet like 210, each packet typically includes a header portion comprising an IP header 220 and a TCP header 230, followed by a |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 273 of 1100

| No. | '111 Patent Claim 16 | Swenson |
|---|---|---|
| | | data portion that contains the information to be communicated in the packet. The information in the IP header 220 contained in a TCP/IP packet 210, or any other IP packet, contains the IP addresses and assures that the packet is delivered to the right host. The transport layer protocol (TCP) header follows the Internet protocol header and specifies the port numbers for the associated service.")<br><br>Copeland at [0077] ("The header portion in the typical TCP/IP datagram 210 is 40 bytes including 20 bytes of IP header 220 information and 20 bytes of TCP header 230 information. The data portion or segment associated with the packet 210 follows the header information.")<br><br>Copeland at [0078] ("In regards to a typical IP packet 210, the first 4 bits of the IP header 220 identify the Internet protocol (IP) version. The following 4 bits identify the IP header length in 32 bit words. The next 8 bits differentiate the type of service by describing how the packet should be handled in transit. The following 16 bits convey the total packet length.")<br><br>Copeland at [0081] ("In a TCP/IP datagram 210, the initial data of the IP datagram is the TCP header 230 information. The initial TCP header 230 information includes the 16-bit source and 16-bit destination port numbers. A 32-bit sequence number for the data in the packet follows the port numbers. Following the sequence number is a 32-bit acknowledgement number. If an ACK flag (discussed below) is set, this number is the next sequence number the sender of the packet expects to receive. Next is a 4-bit data offset, which is the number of 32-bit words in the TCP header. A 6-bit reserved field follows.")<br><br>Copeland at [0082] ("Following the reserved field, the next 6 bits are a series of one-bit flags, shown in FIG. 2 as flags U, A, P, R, S, F. The first flag is the urgent flag (U). If the U flag is set, it indicates that the urgent pointer is valid and points to urgent data that should be acted upon as soon as possible. The next flag is the A ( or ACK or "acknowledgment") flag. The ACK flag indicates that an acknowledgment number is valid, and acknowledges that data has been received. The next flag, the push (P) flag, tells the receiving end to push all buffered data to the receiving application. The reset (R) flag is the following flag, which terminates both ends of the TCP connection. Next, the S (or SYN for "synchronize") flag is set in the initial packet of a TCP connection where both ends have to synchronize their TCP |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 274 of 1100

| No. | '111 Patent Claim 16 | Swenson |
|---|---|---|
| | | buffers. Following the SYN flag is the F (for FIN or "finish") flag. This flag signifies that the sending end of the communication and the host will not send any more data but still may acknowledge data that is received.")<br><br>Copeland at [0083] ("Following the TCP flag bits is a 16-bit receive window size field that specifies the amount of space avail-able in the receive buffer for the TCP connection. The checksum of the TCP header is a 16-bit field. Following the checksum is a 16 bit urgent pointer that points to the urgent data. The TCP/IP datagram data follows the TCP header.")<br><br>Copeland at [0116] ("These steps generally require manipulations of quantities such as IP addresses, packet length, header length, start times, end times, port numbers, and other packet related information. Usually, though not necessarily, these quanti-ties take the form of electrical, magnetic, or optical signals capable of being stored, transferred, combined, compared, or otherwise manipulated. It is conventional for those skilled in the art to refer to these signals as bits, bytes, words, values, elements, symbols, characters, terms, numbers, points, records, objects, images, files or the like. It should be kept in mind, however, that these and similar terms should be associated with appropriate quantities for computer opera-tions and that these terms are merely conventional labels applied to quantities that exist within and during operation of the computer.")<br><br>As another example, Kempf discloses packet headers with flag bits.<br><br>Kempf at [0081] ("In one embodiment, OpenFlow is modified to pro-vide rules for GTP TEID Routing. FIG. 17 is a diagram of one embodiment of the OpenFlow flow table modification for GTP TEID routing. An OpenFlow switch that supports TEID routing matches on the 2 byte (16 bit) collection of header fields and the 4 byte (32 bit) GTP TEID, in addition to other OpenFlow header fields, in at least one flow table ( e.g., the first flow table). The GTP TEID flag can be wildcarded (i.e. matches are "don't care"). In one embodiment, the EPC pro-tocols do not assign any meaning to TEIDs other than as an endpoint identifier for tunnels, like ports in standard UDP/ TCP transport protocols. In other embodiments, the TEIDs can have a correlated meaning or semantics. The GTP header flags field can also be wildcarded, this can be partially matched by combining the following bitmasks: 0xFF00- Match the Message Type field; 0xe0-Match the Version field; 0xl0- |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 275 of 1100

| No. | '111 Patent Claim 16 | Swenson |
|---|---|---|
| | | Match the PT field; 0x04-Match the E field; 0x02- Match the S field; and 0x0l-Match the PN field.")<br><br>Kempf at [0082] ("In one embodiment, OpenFlow can be modified to support virtual ports for fast path GTP TEID encapsulation and decapsulation. An OpenFlow mobile gateway can be used to support GTP encapsulation and decapsulation with virtual ports. The GTP encapsulation and decapsulation virtual ports can be used for fast encapsulation and decapsulation of user data packets within GTP-U tunnels, and can be designed simply enough that they can be implemented in hardware or firmware. For this reason, GTP virtual ports may have the following restrictions on traffic they will handle: Protocol Type (PT) field= 1, where GTP encapsulation ports only sup-port GTP, not GTP' (PT field=0); Extension Header flag (E)=0, where no extension headers are supported, Sequence Number flag (S)=0, where no sequence numbers are sup-ported; N-PDU flag (PN)=0; and Message type=255, where Only G-PDU messages, i.e. tunneled user data, is supported in the fast path.")<br><br>Kempf at [0083] ("If a packet either needs encapsulation or arrives encapsulated with nonzero header flags, header extensions, and/or the GTP-U packet is not a G-PDU packet (i.e. it is a GTP-U control packet), the processing must proceed via the gateway's slow path (software) control plane. GTP-C and GTP' packets directed to the gateway's IP address are a result of mis-configuration and are in error. They must be sent to the OpenFlow controller, since these packets are handled by the S-GW-C and P-GW-C control plane entities in the cloud computing system or to the billing entity handling GTP' and not the S-GW-D and P-GW-D data plane switches.")<br><br>Kempf at [0088] ("To support slow path encapsulation, the software control plane on the switch maintains a hash table with keys calculated from the GTP-U TEID. The TEID hash keys are calculated using a suitable hash algorithm with low collision frequency, for example SHA-1. The flow table entries contain a record of how the packet header, including the GTP encap-sulation header, should be configured. This includes: the same header fields as for the hardware or firmware encapsu-lation table in FIG.18; values for the GTP header flags (PT, E, S, and PN); the sequence number and/or the N-PDU number if any; if the E flag is 1, then the flow table contains a list of the extension headers, including their types, which the slow path should insert into the GTP header.") |

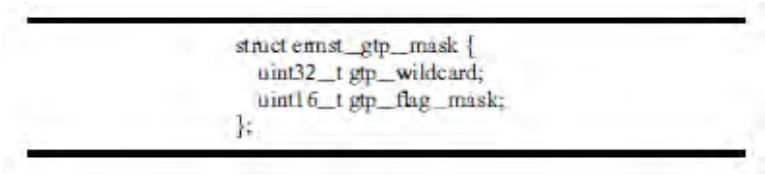


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 276 of 1100

| No. | '111 Patent Claim 16 | Swenson |
|---|---|---|
| | | Kempf at [0092] ("In one embodiment, the system implements a GTP fast path decapsulation virtual port. When requested by the S-GW and P-GW control plane software running in the cloud computing system, the gateway switch installs rules and actions for routing GTP encapsulated packets out of GTP tunnels. The rules match the GTP header flags and the GTP TEID for the packet, in the modified OpenFlow flow table shown in FIG. 17 as follows: the IP destination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); the UDP destination port is the GTP-U destination port (2152); and the header fields and message type field is wildcarded with the flag 0XFFF0 and the upper two bytes of the field match the G-PDU message type (255) while the lower two bytes match 0x30, i.e. the packet is a GTP packet not a GTP' packet and the version number is 1.")<br><br>Kempf at [0094] ("In one embodiment, the system implements han-dling of GTP-U control packets. The OpenFlow controller programs the gateway switch flow tables with 5 rules for each gateway switch IP address used for GTP traffic. These rules contain specified values for the following fields: the IP des-tination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); the UDP destination port is the GTP-U destination port (2152); the GTP header flags and message type field is wildcarded with 0xFFF0; the value of the header flags field is 0x30, i.e. the version number is 1 and the PT field is 1; and the value of the message type field is one of 1 (Echo Request), 2 (Echo Response), 26 (Error Indication), 31 (Support for Extension Headers Notification), or 254 (End Marker).")<br><br>Kempf at [0098] ("The header flags and message type fields for the three rules are wildcarded with the following bitmasks and match as follows: bitmask 0xFFF4 and the upper two bytes match the G-PDU message type (255) while the lower two bytes are Ox34, indicating that the version number is 1, the packet is a GTP packet, and there is an extension header present; bitmask 0xFFFF2 and the upper two bytes match the G-PDU message type (255) while the lower two bytes are 0x32, indicating that the version number is 1, the packet is a GTP packet, and there is a sequence number present; and bitmask 0xFF0l and the upper two bytes match the G-PDU message type (255) while the lower two bytes are 0x31, indicating that the version number is 1, the packet is a GTP packet, and a N-PDU is present.") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 277 of 1100

| No. | '111 Patent Claim 16 | Swenson |
|---|---|---|
| | | Kempf at [0114] ("The gtp_type_n_flags field contains the GTP mes-sage type in the upper 8 bits and the GTP header flags in the lower 8 bits. The gtp_teid field contains the GTP TEID. The gtp_ wildcard field indicates whether the GTP type and flags and TEID should be matched. If the lower four bits are 1, the type and flags field should be ignored, while if the upper four bits are 1, the TEID should be ignored. If the lower bits are 0, the type and fields flag should be matched subject to the flags in the gtp_flag_mask field, while if the upper bits are 0 the TEID should be matched. The mask is combined with the message type and header field of the packet using logical AND; the result becomes the value of the match. Only those parts of the field in which the mask has a 1 value are matched.")<br><br>Kempf at [0117] ("The gtp_type_n_flags field contains the GTP mes-sage type in the upper 8 bits and the GTP header flags in the lower 8 bits. The gtp_<br>teid field contains the GRP TEID. When the value of the oxm_type ( oxm_class+oxm_field is GTP _ MATCH and the HM bit is zero, the flaw's GTP header must match these values exactly. If the HM flag is one, the value contains an ersmt_gtp_match field and an ermst_gtp_mask field, as specified by the OpenFlow 1.2 specification. We define ermst_gtp_mask field for selecting flows based on the settings of flag bits:<br><br>  <br><br><br>Kempf at [0118] ("The gtp_ wildcard field indicates whether the TEID should be matched. If the value is 0xFFFFFFFF, the TEID should be matched and not the flags, if the value is 0x00000000, the flags should be matched and not the TEID. If the gtp_ wildcard indicates the flags should be matched, the gtp_flag_mask is combined with the message type and header field of the packet using logical AND, the result becomes the value of the match. Only those parts of the field in which the mask has a 1 value are matched.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 278 of 1100

| No. | '111 Patent Claim 16 | Swenson |
|---|---|---|
| 16[c] | wherein the packet-applicable criterion is that one or more of the flag bits is set. | Swenson discloses wherein the packet-applicable criterion is that one or more of the flag bits is set.<br><br>For example, Swenson discloses one or more signatures, desired criteria, or conditions of the packet flow used by the steering device to determine the categorization of network traffic which may include one or more flag bits of the header is set. A person of ordinary skill in the art would understand that one or more signatures, desired criteria, or conditions of the packet flow could be that one or more of the flag bits is set. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Swenson is found to not meet this limitation, wherein the packet applicable criterion is that one or more of the flag bits is set would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0040] ("The flow analyzer 312 monitors large flows in the network, analyzes collected flow statistics to determine net-work throughput, and accordingly selects flows to be opti-mized. The flow analyzer 312 does not need to see all the flows in order to make an accurate estimate of network con-ditions. The flow analyzer 312 processes the traffic statistics stored in the flow cache 3 22 and user information stored in the subscriber log 324, for example, by associating network flows identified by source IP addresses to a mobile subscriber or user, which is identified by his or her current subscriber ID or device ID. The user flows are also mapped to a congestion level at the current sub-network (e.g., a cell with |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 279 of 1100

| No. | '111 Patent Claim 16 | Swenson |
|---|---|---|
| | | which the user devices are associated), so that an optimization decision can be made at the beginning of the data transmission.")<br><br>Swenson at [0068] ("In one embodiment, responsive to an HTTP get request 522 to an origin server 160, the video optimizer receives a HTTP 404 error from the origin server 160 as opposed to a video file. In such case, the video optimizer 150 appends a "do not transcode" flag to the HTTP redirect request and returned to the user device 110, which re-sends the request out over the network to the origin server 160. The origin server 160 responds appropriately to the request by sending back video 524, which is intercepted by the steering device 130 and the inline on-demand element the network controller 140 for monitoring purpose.")<br><br>Swenson at [0072] ("In one embodiment, if the video optimizer 150 failed to retrieve user requested video file from the origin server 160, the video optimizer 150 appends a "do not transcode" flag to the HTTP redirect request and returned to the user device 110, which re-sends the request out over the network to the origin server 160. The origin server 160 responds appropriately to the request by sending back video 624, which is intercepted by the steering device 130 only. The steering device 130 forwards the video to the user device 110 and at the same time reports the flow size to the network controller 140 for monitoring purpose.")<br><br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Swenson in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 16[c] of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references.<br><br>For example, Copeland discloses packet specific characteristics including flag bits that are set.<br><br>Copeland at [0081] ("In a TCP/IP datagram 210, the initial data of the IP datagram is the TCP header 230 information. The initial TCP header 230 information includes the 16-bit source and 16-bit destination port numbers. A 32-bit sequence number for the data in the packet follows the port numbers. Following the sequence number is a 32-bit |


~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 280 of 1100

| No. | '111 Patent Claim 16 | Swenson |
|---|---|---|
| | | acknowledgement number. If an ACK flag (discussed below) is set, this number is the next sequence number the sender of the packet expects to receive. Next is a 4-bit data offset, which is the number of 32-bit words in the TCP header. A 6-bit reserved field follows.")<br><br>Copeland at [0082] ("Following the reserved field, the next 6 bits are a series of one-bit flags, shown in FIG. 2 as flags U, A, P, R, S, F. The first flag is the urgent flag (U). If the U flag is set, it indicates that the urgent pointer is valid and points to urgent data that should be acted upon as soon as possible. The next flag is the A ( or ACK or "acknowledgment") flag. The ACK flag indicates that an acknowledgment number is valid, and acknowledges that data has been received. The next flag, the push (P) flag, tells the receiving end to push all buffered data to the receiving application. The reset (R) flag is the following flag, which terminates both ends of the TCP connection. Next, the S (or SYN for "synchronize") flag is set in the initial packet of a TCP connection where both ends have to synchronize their TCP buffers. Following the SYN flag is the F (for FIN or "finish") flag. This flag signifies that the sending end of the communication and the host will not send any more data but still may acknowledge data that is received.")<br><br>Copeland at [0083] ("Following the TCP flag bits is a 16-bit receive window size field that specifies the amount of space avail-able in the receive buffer for the TCP connection. The checksum of the TCP header is a 16-bit field. Following the checksum is a 16 bit urgent pointer that points to the urgent data. The TCP/IP datagram data follows the TCP header.")<br><br>Copeland at [0089] ("FIG. 3 illustrates an exemplary TCP/IP session 300. As discussed in reference to FIG. 2, the SYN flag is set whenever one host initiates a session with another host. In the initial packet, Hostl sends a message with only the SYN flag set. The SYN flag is designed to establish a TCP connection and allow both ends to synchronize their TCP buffers. Hostl provides the sequence of the first data packet it will send.")<br><br>Copeland at [0125] ("For purposes of the description, which follows, the IP address with the lower value, when considered as a 32-bit unsigned integer, is designated ip[0] and the corresponding port number is designated pt[0]. The higher IP address is designated ip[l] and the corresponding TCP or UDP port number is designated pt[l]. At some point, either pt[0] or pt[l] may be designated the "server" port by setting an appropriate bit in a bit map that is part of the flow record (record "state", bit 1 or 2 is set).") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 281 of 1100

| No. | '111 Patent Claim 16 | Swenson |
|---|---|---|
| | | Copeland at [0145] ("A list IP of addresses contacted or probed by each host can be maintained. When this list indicates that more than a threshold number of other hosts (e.g., 8) have been contacted in the same subnet, CI is added to the to the host and a bit in the host record is set to indicate that the host has received CI for "address scanning." Note that the number of hosts to designate a scan is not required to be a fixed value, but could be adjusted based on the sample rate or other means to enhance the accuracy making the number of hosts scanned "statistically significant". These and other values of concern index are shown for non-flow based events in FIG. 7.")<br><br>As another example, Kempf flow table matches in which the flag bits is set.<br><br>Kempf at [0081] ("In one embodiment, OpenFlow is modified to pro-vide rules for GTP TEID Routing. FIG. 17 is a diagram of one embodiment of the OpenFlow flow table modification for GTP TEID routing. An OpenFlow switch that supports TEID routing matches on the 2 byte (16 bit) collection of header fields and the 4 byte (32 bit) GTP TEID, in addition to other OpenFlow header fields, in at least one flow table ( e.g., the first flow table). The GTP TEID flag can be wildcarded (i.e. matches are "don't care"). In one embodiment, the EPC pro-tocols do not assign any meaning to TEIDs other than as an endpoint identifier for tunnels, like ports in standard UDP/ TCP transport protocols. In other embodiments, the TEIDs can have a correlated meaning or semantics. The GTP header flags field can also be wildcarded, this can be partially matched by combining the following bitmasks: 0xFF00- Match the Message Type field; 0xe0-Match the Version field; 0xl0-Match the PT field; 0x04-Match the E field; 0x02- Match the S field; and 0x0l-Match the PN field.")<br><br>Kempf at [0082] ("In one embodiment, OpenFlow can be modified to support virtual ports for fast path GTP TEID encapsulation and decapsulation. An OpenFlow mobile gateway can be used to support GTP encapsulation and decapsulation with virtual ports. The GTP encapsulation and decapsulation virtual ports can be used for fast encapsulation and decapsulation of user data packets within GTP-U tunnels, and can be designed simply enough that they can be implemented in hardware or firmware. For this reason, GTP virtual ports may have the following restrictions on traffic they will handle: Protocol Type (PT) field= 1, where GTP encapsulation ports only sup-port GTP, not GTP' (PT field=0); |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 282 of 1100

| No. | '111 Patent Claim 16 | Swenson |
|---|---|---|
| | | Extension Header flag (E)=0, where no extension headers are supported, Sequence Number flag (S)=0, where no sequence numbers are sup-ported; N-PDU flag (PN)=0; and Message type=255, where Only G-PDU messages, i.e. tunneled user data, is supported in the fast path.")<br><br>Kempf at [0083] ("If a packet either needs encapsulation or arrives encapsulated with nonzero header flags, header extensions, and/or the GTP-U packet is not a G-PDU packet (i.e. it is a GTP-U control packet), the processing must proceed via the gateway's slow path (software) control plane. GTP-C and GTP' packets directed to the gateway's IP address are a result of mis-configuration and are in error. They must be sent to the OpenFlow controller, since these packets are handled by the S-GW-C and P-GW-C control plane entities in the cloud computing system or to the billing entity handling GTP' and not the S-GW-D and P-GW-D data plane switches.")<br><br>Kempf at [0088] ("To support slow path encapsulation, the software control plane on the switch maintains a hash table with keys calculated from the GTP-U TEID. The TEID hash keys are calculated using a suitable hash algorithm with low collision frequency, for example SHA-1. The flow table entries contain a record of how the packet header, including the GTP encap-sulation header, should be configured. This includes: the same header fields as for the hardware or firmware encapsu-lation table in FIG.18; values for the GTP header flags (PT, E, S, and PN); the sequence number and/or the N-PDU number if any; if the E flag is 1, then the flow table contains a list of the extension headers, including their types, which the slow path should insert into the GTP header.")<br><br>Kempf at [0092] ("In one embodiment, the system implements a GTP fast path decapsulation virtual port. When requested by the S-GW and P-GW control plane software running in the cloud computing system, the gateway switch installs rules and actions for routing GTP encapsulated packets out of GTP tunnels. The rules match the GTP header flags and the GTP TEID for the packet, in the modified OpenFlow flow table shown in FIG. 17 as follows: the IP destination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); the UDP destination port is the GTP-U destination port (2152); and the header fields and message type field is wildcarded with the flag 0XFFF0 and the upper two bytes of the field match the G-PDU message type (255) |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 283 of 1100

| No. | '111 Patent Claim 16 | Swenson |
|---|---|---|
| | | while the lower two bytes match 0x30, i.e. the packet is a GTP packet not a GTP' packet and the version number is 1.")<br><br>Kempf at [0094] ("In one embodiment, the system implements han-dling of GTP-U control packets. The OpenFlow controller programs the gateway switch flow tables with 5 rules for each gateway switch IP address used for GTP traffic. These rules contain specified values for the following fields: the IP des-tination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); the UDP destination port is the GTP-U destination port (2152); the GTP header flags and message type field is wildcarded with 0xFFF0; the value of the header flags field is 0x30, i.e. the version number is 1 and the PT field is 1; and the value of the message type field is one of 1 (Echo Request), 2 (Echo Response), 26 (Error Indication), 31 (Support for Extension Headers Notification), or 254 (End Marker).")<br><br>Kempf at [0098] ("The header flags and message type fields for the three rules are wildcarded with the following bitmasks and match as follows: bitmask 0xFFF4 and the upper two bytes match the G-PDU message type (255) while the lower two bytes are Ox34, indicating that the version number is 1, the packet is a GTP packet, and there is an extension header present; bitmask 0xFFFF2 and the upper two bytes match the G-PDU message type (255) while the lower two bytes are 0x32, indicating that the version number is 1, the packet is a GTP packet, and there is a sequence number present; and bitmask 0xFF0l and the upper two bytes match the G-PDU message type (255) while the lower two bytes are 0x31, indicating that the version number is 1, the packet is a GTP packet, and a N-PDU is present.")<br><br>Kempf at [0114] ("The gtp_type_n_flags field contains the GTP mes-sage type in the upper 8 bits and the GTP header flags in the lower 8 bits. The gtp_teid field contains the GTP TEID. The gtp_ wildcard field indicates whether the GTP type and flags and TEID should be matched. If the lower four bits are 1, the type and flags field should be ignored, while if the upper four bits are 1, the TEID should be ignored. If the lower bits are 0, the type and fields flag should be matched subject to the flags in the gtp_flag_mask field, while if the upper bits are 0 the TEID should be matched. The mask is combined with the message type and header field of the packet using logical AND; the result becomes the value of the match. Only those parts of the field in which the mask has a 1 value are matched.") |

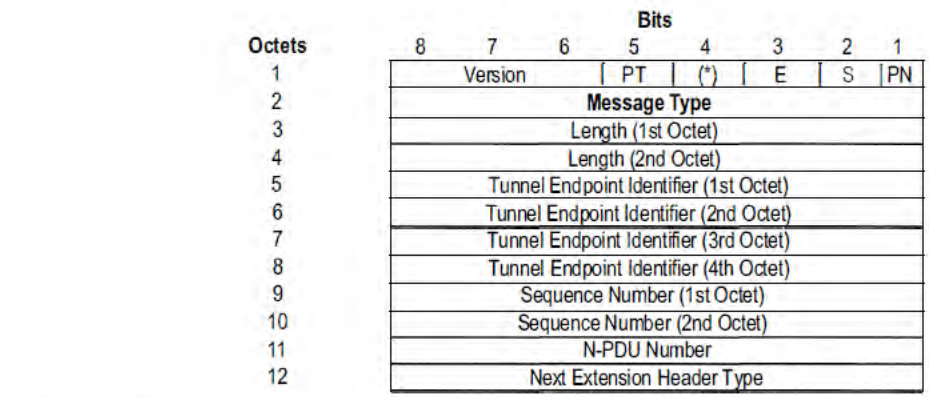


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 284 of 1100

| No. | '111 Patent Claim 16 | Swenson |
|---|---|---|
| | | Kempf at [0117] ("The gtp_type_n_flags field contains the GTP mes-sage type in the upper 8 bits and the GTP header flags in the lower 8 bits. The gtp_teid field contains the GRP TEID. When the value of the oxm_type ( oxm_class+oxm_field is GTP _ MATCH and the HM bit is zero, the flaw's GTP header must match these values exactly. If the HM flag is one, the value contains an ersmt_gtp_match field and an ermst_gtp_mask field, as specified by the OpenF!ow 1.2 specification. We define ermst_gtp_mask field for selecting flows based on the settings of flag bits:<br><br>struct ermst_gtp_mask {<br>uint32_t gtp_wildcard;<br>uint16_t gtp_flag_mask;<br>};<br><br>Kempf at [0118] ("The gtp_ wildcard field indicates whether the TEID should be matched. If the value is 0xFFFFFFFF, the TEID should be matched and not the flags, if the value is 0x00000000, the flags should be matched and not the TEID. If the gtp_ wildcard indicates the flags should be matched, the gtp_flag_mask is combined with the message type and header field of the packet using logical AND, the result becomes the value of the match. Only those parts of the field in which the mask has a 1 value are matched.")<br><br>Kempf at Figure 10 |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 285 of 1100

| No. | '111 Patent Claim 16 | Swenson |
|---|---|---|
| | |  |

| Octets | Bits 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
|---|---|---|---|---|---|---|---|---|
| 1 | Version | | | PT | (*) | E | S | PN |
| 2 | Message Type | | | | | | | |
| 3 | Length (1st Octet) | | | | | | | |
| 4 | Length (2nd Octet) | | | | | | | |
| 5 | Tunnel Endpoint Identifier (1st Octet) | | | | | | | |
| 6 | Tunnel Endpoint Identifier (2nd Octet) | | | | | | | |
| 7 | Tunnel Endpoint Identifier (3rd Octet) | | | | | | | |
| 8 | Tunnel Endpoint Identifier (4th Octet) | | | | | | | |
| 9 | Sequence Number (1st Octet) | | | | | | | |
| 10 | Sequence Number (2nd Octet) | | | | | | | |
| 11 | N-PDU Number | | | | | | | |
| 12 | Next Extension Header Type | | | | | | | |

NOTE 0: (*) This bit is a spare bit. It shall be sent as '0'. The receiver shl not evaluate this bit,
NOTE 1: 1) This field shall only be evaluated when indicated by the S flag set to 1.
NOTE 2: 2) This field shall only be evaluated when indicated by the PN flag set to 1.
NOTE 3: 3) This field shall only be evaluated when indicated by the E flag set to 1.
NOTE 4: 4) This field shall be present if and only if any one or more of the S. PN and E flags are set.

FIG. 10

| No. | '111 Patent Claim 17 | Swenson |
|---|---|---|
| 17[a] | The method according to claim 16, wherein the packet is an Transmission Control Protocol (TCP) packet, and | Swenson discloses the method according to claim 16, wherein the packet is an Transmission Control Protocol (TCP) packet.<br><br>For example, Swenson discloses TCP packet flows.<br><br>*See supra* at Claim 16.<br><br>Swenson at [0019] (" In one embodiment, an on-demand network moni-toring method is adopted to gather data about network flows as they traverse the network. For example, network flows can be monitored selectively or on-demand based on the types of the content carried in the flows. Furthermore, the network monitoring can also be performed selectively |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 286 of 1100

| No. | '111 Patent Claim 17 | Swenson |
|---|---|---|
| | | at inline level, as well as out-of-band to improve efficiency. Both TCP and UDP flows are monitored to gather information about the state of the network, such as the average network throughput for each flow and end-to-end latency between, for example, a client device and an origin server providing multimedia con-tent to the client device. For each TCP or UDP flow, the system tracks the number of bytes sent ( and in some embodi-ments acknowledged). In TCP, the current window size may also be tracked. Records on network flows are stored in a flow statistics database, which can be indexed by subscriber iden-tification (ID), cell tower (base station), and network segment etc. As many flow records accumulate, this database repre-sents both historical and current network condition and capacity for delivering data. Network throughput can be mea-sured by calculating an average number of bytes delivered over a period of time. Steps may be taken to filter out spurious data from small flows with size less than a certain threshold that, when measured, cause very noisy results in measuring bandwidth and/or latency. For example, any flow having delivery time of less than 500 ms can be filtered.")<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0028] ("In contrast to conventional inline TCP throughput monitoring devices that monitor every single data packets transmitted and received, the network controller 140 is an "out-of-band" computer server that interfaces with the steer-ing device 130 to selectively inspect user flows of interest. The network controller 140 may further identify user flows (e.g., among the flows of interest) for optimization. In one embodiment, the network controller 140 may be imple-mented at the steering device 130 to monitor traffic. In other embodiments, the network controller 140 is coupled to and communicates with the steering device 130 for traffic moni-toring and optimization. When queried by the steering |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 287 of 1100

| No. | '111 Patent Claim 17 | Swenson |
|---|---|---|
| | | device 130, the network controller 140 determines if a given network flow should be ignored, monitored further or optimized. Opti-mization of a flow is often decided at the beginning of the flow because it is rarely possible to switch to optimized content mid-stream once non-optimized content delivery has begun. However, the network controller 140 may determine that existing flows associated with a particular subscriber or other entity should be optimized. In turn, new flows ( e.g., resulting from seek requests in media, new media requests, resume after pause, etc.) determined to be associated with the entity may be optimized. The network controller 140 uses the net-work state as well as historical traffic data in its decision for monitoring and optimization. Knowledge on the current net-work state, such as congestion, deems critical when it comes to data optimization.")<br><br>Swenson at [0044] ("Additionally, historical flow data over a longer term helps the flow analyzer 312 to determine repeating patterns and heat-maps of certain network sections and to predict when they are under congestion. In this case, the flow statis-tics stored in the flow cache 322 can be mapped against traffic categories for analysis, for example, long-term running aver-ages of video flow bandwidth help determine suitability for optimization. Furthermore, estimated bandwidth per user ( or per cell-ID, per tower, or per router) over time may be metrics calculated by the flow analyzer 312 in order to determine short term needs for optimization. For example, the flow analyzer 312 may determine to being optimizing flows asso-ciated with a particular cell-ID (or those flows for identified high-bandwidth users on the cell-ID) in response to a thresh-old number of high-bandwidth users connecting to a same cell tower corresponding to the cell-ID. The reason why flow analyzer 312 selectively monitors large flows lies in the real-ization that TCP statistics for small objects, which make up most web flows, can be misleading and cause huge errors in throughput estimations.")<br><br>Swenson at [0061] ("Once a flow is reported to the network controller 140, a flow cache entry is created for the flow in the flow cache 322. The flow cache entry keeps track of the flow and its associated bandwidth. For a flow that is marked in "continue" mode, each time the steering device 130 forwards a next portion of the flow payload to the network controller 140, the flow cache 3 22 updates the number of bytes for transmitted in the flow. By monitoring the number of bytes per flow over time, the flow analyzer 312 is capable of determining an estimate value of bandwidth associated with flow. Further-more, since the steering device 130 does not have infinite packet buffers, if congestion happens on the |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 288 of 1100

| No. | '111 Patent Claim 17 | Swenson |
|---|---|---|
| | | network link 416 from the steering device 130 to the user device 110, the TCP congestion control mechanism kicks in at the steering device 130, which may slows down and/or eventually stop receiving data over the network link 413 from origin server 160. During the congestion, the steering device 130 would not forward any data to the network controller 140, since the link 416 is congested and the network controller 140 would not be able to transmit data to the user device 110. Therefore, as an inline element, the network controller 140 can detect network con-gestions and estimate bandwidth associated with any flows of interest selected by the network controller 140. However, in the "continue" mode, the network controller 140 does not modify and transform the HTTP messaged it receives over the ICAP interface. The network controller 140 simply updates the flow statistics and returns the video or images to the steering device 130 for transmission to the user device 110.")<br><br>Swenson at [0062] (" Based on the flow statistics stored in the flow cache 322, the network controller 140 can also aggregate the flows associated with a user or subscriber in order to estimate the total available bandwidth occupied by the user or subscriber. In one embodiment, the network controller 140 tracks all the flow cache entries looking for flows originated from a com-mon source IP address or a user device identifier. The flow analyzer 312 of the network controller 140 then attempts to group these flows together to form a flow history for the user or subscriber. The network controller further identifies users or subscribers using two data components in the flow cache entry: the TCP source port and HTTP cookies associated with the flow. Together with the flow history, the network control-ler 140 establish pattern, and identify users or subscribers and stores subscriber information in the subscriber log 324. More details of the flow cache and user mapping are described below with reference to FIG. 7.")<br><br>Swenson at [0084] ("The flow analyzer 312 can also map flows to users (subscribers to the mobile or network service) in the flow cache entries by matching cookie hashes, MAC address ( or any unique device identifiers), or TCP source ports. For example, if two flows share the same source port, it is very likely that they belong to the same user because TCP ports are reused often by an individual user, but not often between users. Furthermore, source ports can also be used to map users when network address translation (NAT) is deployed. In a typical network with NAT configuration, each user is allo-cated a block (e.g., 32) of TCP source ports. A random port number within the block is then picked for each |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 289 of 1100

| No. | '111 Patent Claim 17 | Swenson |
|---|---|---|
| | | new user flows initiated. With this knowledge, all source ports within a block can be aggregated under the same user. In some cases, a user with more than one block of port number assigned, the cookie hashes can be used to link the blocks together.") |
| 17[b] | wherein the one or more flag bits comprises comprise a SYN flag bit, an ACK flag bit, a FIN flag bit, a RST flag bit, or any combination thereof. | Swenson discloses wherein the one or more flag bits comprises comprise a SYN flag bit, an ACK flag bit, a FIN flag bit, a RST flag bit, or any combination thereof.<br><br>For example, Swenson discloses headers that may include flag bits. A person of ordinary skill in the art would understand that such flag bits may be a SYN flag bit, an ACK flag bit, a FIN flag bit, a RST flag bit, or any combination thereof. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Swenson is found to not meet this limitation, wherein the one or more flag bits comprises comprise a SYN flag bit, an ACK flag bit, a FIN flag bit, a RST flag bit, or any combination thereof would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Swenson in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 17[b] of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 290 of 1100

| No. | '111 Patent Claim 17 | Swenson |
|---|---|---|
| | | For example, Copeland discloses TCP packets with flag bits including SYN, ACK, FIN, and R flag bits.<br><br>Copeland at [0081] ("In a TCP/IP datagram 210, the initial data of the IP datagram is the TCP header 230 information. The initial TCP header 230 information includes the 16-bit source and 16-bit destination port numbers. A 32-bit sequence number for the data in the packet follows the port numbers. Following the sequence number is a 32-bit acknowledgement number. If an ACK flag (discussed below) is set, this number is the next sequence number the sender of the packet expects to receive. Next is a 4-bit data offset, which is the number of 32-bit words in the TCP header. A 6-bit reserved field follows.")<br><br>Copeland at [0082] ("Following the reserved field, the next 6 bits are a series of one-bit flags, shown in FIG. 2 as flags U, A, P, R, S, F. The first flag is the urgent flag (U). If the U flag is set, it indicates that the urgent pointer is valid and points to urgent data that should be acted upon as soon as possible. The next flag is the A ( or ACK or "acknowledgment") flag. The ACK flag indicates that an acknowledgment number is valid, and acknowledges that data has been received. The next flag, the push (P) flag, tells the receiving end to push all buffered data to the receiving application. The reset (R) flag is the following flag, which terminates both ends of the TCP connection. Next, the S (or SYN for "synchronize") flag is set in the initial packet of a TCP connection where both ends have to synchronize their TCP buffers. Following the SYN flag is the F (for FIN or "finish") flag. This flag signifies that the sending end of the communication and the host will not send any more data but still may acknowledge data that is received.")<br><br>Copeland at [0089] ("FIG. 3 illustrates an exemplary TCP/IP session 300. As discussed in reference to FIG. 2, the SYN flag is set whenever one host initiates a session with another host. In the initial packet, Hostl sends a message with only the SYN flag set. The SYN flag is designed to establish a TCP connection and allow both ends to synchronize their TCP buffers. Hostl provides the sequence of the first data packet it will send.")<br><br>Copeland at [0090] ("Host2 responds with a SYN-ACK packet. In this message, both the SYN flag and the ACK flag are set. Host2 provides the initial sequence number for its data to Hostl. Host2 also sends to Hostl the acknowledgment number that is the next sequence number Host2 expects to receive from host 1. In the SYN-ACK packet sent by Host2, the |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 291 of 1100

| No. | '111 Patent Claim 17 | Swenson |
|---|---|---|
| | | acknowl-edgment number is the initial sequence number of Hostl plus 1, which should be the next sequence number received.")<br><br>Copeland at [0091] ("Hostl responds to the SYN-ACK with a packet with just the ACK flag set. Hostl acknowledges that the next packet of information received from Host2 will be Host2's initial sequence number plus 1. The three-way handshake is complete and data is transferred.")<br><br>Copeland at [0092] ("Host2 responds to ACK packet with its own ACK packet. Host2 acknowledges the data it has received from Hostl by sending an acknowledgment number one greater than its last received data sequence number. Both hosts send packets with the ACK flag set until the session is to end although the P and U flags may also be set, if warranted.")<br><br>Copeland at [0093] ("As illustrated, when Hostl terminates its end of the session, it sends a packet with the FIN and ACK flags set. The FIN flag informs Host2 that Hostl will send no more data. The ACK flag acknowledges the last data received by Hostl by informing Host2 of the next sequence number it expects to receive.")<br><br>Copeland at [0094] ("Host2 acknowledges the FIN packet by sending its own ACK packet. The ACK packet has the acknowledge-ment number one greater than the sequence number of Hostl's FIN-ACK packet. ACK packets are still delivered between the two hosts, except that HOSTl's packets have no data appended to the TCP/IP end of the headers.")<br><br>Copeland at [0095] ("When Host 2 is ready to terminate the session, it sends its own packet with the FIN and ACK flags set. Hostl responds that it has received the final packet with an ACK packet providing to Host2 an acknowledgment number one greater than the sequence number provided in the FIN-ACK packet of Host2.")<br><br>As another example, Uchida discloses the TCP (Transmission Control Protocol) FIN flag, RST flag, and SYN flag.<br><br>Uchida at [0040] ("A flow end can be detected by various methods as below. For example, in one method, a protocol end message is checked. For example, in the TCP (Transmission |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 292 of 1100

| No. | '111 Patent Claim 17 | Swenson |
|---|---|---|
| | | Control Protocol), a FIN flag is checked. In this way, the end of communication, that is, the end of a flow using communica-tion, can be detected. In practice, after a FIN flag, communi-cation with an ACK packet is generated in a reverse-direction flow (a flow in which the source and the destination are reversed). Thus, by detecting the ACK flag in the reverse-direction flow after the FIN packet, a flow end can be deter-mined. Further, since the TCP is used in bidirectional com-munication, the forward- and reverse-direction flows can be used as a pair to determine a flow end. Namely, if the end of a flow is detected, a process rule corresponding to the reverse-direction flow of the flow can also be determined to be unnec-essary. Alternatively, a communication end can also be deter-mined when a predetermined time elapses after reception of a SYN packet and a timeout is determined. Still alternatively, a communication end can be determined by reception of a RST packet. These methods will be described in more detail later as specific examples.")<br><br>Uchida at [0050] ("The flow end check unit can use at least one of a TCP (Transmission Control Protocol) FIN flag, RST flag, and SYN flag extracted by the end determination information extraction unit to determine a flow end.")<br><br>Uchida at [0055] ("In the process rule update method, a flow end can be determined by at least one of a TCP (Transmission Control Protocol) FIN flag, RST flag, and SYN flag.")<br><br>Uchida at [0102] ("Next, specific examples 1 to 3 will be described. In the examples 1 to 3, a flow end is determined by combining features of the above individual exemplary embodiments and using TCP (Transmission Control Protocol) flags.")<br><br>Uchida at [0103] ("FIG. 6 is a state transition diagram of TCP connec-tion. "CLOSED" at the top of FIG. 6 represents the end of TCP communication, and portions connected thereto repre-sent states prior to the end of TCP communication. Approxi-mately 2MSL (MSL: Maximum Segment Lifetime) is the maximum amount of time required to reach the above "CLOSED," that is, if the packet forwarding apparatus stands by for approximately 2MSL after both FINs flow, the above "CLOSED" is reached. Thus, after a FIN is confirmed in either direction, if this 2MSL elapses, basically, a communi-cation end can be determined. Even if the state does not change smoothly because of packet loss or the like (for example, even if an ACK packet does not arrive after "CLOS-ING"), a retransmitted packet is forwarded immediately after this 2MSL. Thus, the end of TCP communication can be |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 293 of 1100

| No. | '111 Patent Claim 17 | Swenson |
|---|---|---|
| | | determined if a new FIN packet is not received within the time corresponding to the 2MSL and a margin (2MSL+a) at long-est.")<br><br>Uchida at [0104] ("Hereinafter, the description will be made, assuming that a packet forwarding apparatus Cl according to the present invention relays TCP communication between a com-puter (client) Dl 0 and a server D20 that use network configu-rations illustrated in FIG. 7. In the example of FIG. 7, the computer Dl0 belongs to a network represented by 192.168. 0./24 and is set by 192.168.0.10. The server D20 belongs to a network represented by 192.168.1./24 and is set by 192.168. 1.10. As in the case of the OpenFlow controller described in Non-Patent Documents 1 and 2, a control apparatus ( control-ler) Dl is connected to the packet forwarding apparatus Cl via a dedicated channel and manages connection between the two networks. In the following description, the control appa-ratus (controller) Dl controls the packet forwarding appara-tus Cl so that connection from other networks appears as communication from network number 1 (192.168.1.1) of the respective networks (see process rule actions in FIG. 19). In addition, in the present specific example, since FIN packets are monitored, the end determination information extraction unit Cl 7 monitors a protocol stack, including: fields in which the TCP is determined; and the FIN flag in the TCP header.")<br><br>Uchida at [0105] ("FIG. 8 is a flow chart of a flow end determination process using FIN flags. In FIG. 8, steps relating to a timeout determination are added to steps Slll to S116 in the flow chart in FIG. 3. Thus, the flow chart in FIG. 8 includes more detailed steps than the flow chart of FIG. 3. Hereinafter, operations will be described with reference to FIGS. 3, 6, and 8 and FIGS. 9 to 13. In practice, prior to TCP/IP communi-cation, ARP (Address Resolution Protocol) communication is executed, and a process rule may be set in that stage. However, for ease of description, description of the ARP communication will be omitted. The following description will be made based on communication at the TCP/IP level.")<br><br>Uchida at [0106] ("First, the computer Dl0 starts communication with the server D20. For an initial establishment of communica-tion, a packet (SYN) is inputted to the packet forwarding apparatus Cl (start of ACTIVE OPEN through SYN forward-ing in FIG. 6). The packet reception unit Cl0 receives and stores this first packet in the packet storage unit Cll (steps SlOl to S102 in FIG. 3).") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 294 of 1100

| No. | '111 Patent Claim 17 | Swenson |
|---|---|---|
| | | Uchida at [0107] ("The packet reception unit C10 notifies the packet process information extraction unit C12 and the end determination information extraction unit C17 of reception of the packet. The packet process information extraction unit C12 refers to the packet storage unit C11 and extracts information such as IP source and destination information that is necessary to search for a process rule (step S103 in FIG. 3). Hereinafter, a process corresponding to steps S103 to S110 in FIG. 3 will be executed.")<br><br>Uchida at [0115] ("Upon receiving a notification that the packet has been received by the packet reception unit Cl 0, the end deter-mination information extraction unit Cl 7 refers to the packet storage unit Cll, monitors a TCP FIN flag, and finds a FIN flag (step S201 in FIG. 8).")<br><br>Uchida at [0116] ("Since a FIN flag is set, the end determination infor-mation extraction unit Cl 7 determines that the packet includes information necessary for determining a flow end. Thus, the end determination information extraction unit Cl 7 extracts information for identifying a process rule to be deleted (the ingress port is 1; the source address is 192.168. 0.10; the destination is 192.168.1.10; and the protocol is TCP (the type is Ox0006)) and stands by until forwarding of the packet. Upon receiving a notification that the packet has been transmitted by the packet forwarding unit C16, the end deter-mination information extraction unit Cl 7 further extracts information for identifying a process rule to be deleted from the packet storage unit Cll. Since the IP address is replaced, the extracted information for identifying a process rule to be deleted represents that the source address is 192.168.1.1; the destination is 192.168.1.1 0; and the protocol is TCP (the type is 0x0006). The information is used for marking of the reverse flow. The end determination information extraction unit Cl 7 notifies the flow end check unit C18 of the notification that the FIN packet has been received and these items of information (step S202 in FIG. 8).")<br><br>Uchida at [0117] ("Upon receiving the above information from the end determination information extraction unit Cl 7, the flow end check unit C18 checks whether or not a FIN flag is set in a predetermined packet header position (step S203). These steps correspond to steps Slll to S114 in FIG. 3.")<br><br>Uchida at [0121] ("Next, after an ACK reply in response to the FIN packet from the computer DlO is forwarded from the server D20 in the same way as the above normal |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 295 of 1100

| No. | '111 Patent Claim 17 | Swenson |
|---|---|---|
| | | packet (start of PASSIVE CLOSE in FIG. 6), the server D20 transmits a FIN packet to the computer DlO. When this FIN packet is inputted to the packet forwarding apparatus Cl, the flow end determi-nation process from steps Slll to S116 is started, as in the case of the above start of ACTIVE CLOSE.")<br><br>Uchida at [0122] ("Upon receiving a notification that the packet has been received from the packet reception unit Cl0, the end determination information extraction unit Cl 7 refers to the packet storage unit Cll, monitors a TCP FIN flag, and finds a FIN packet (step S201 in FIG. 8).")<br><br>Uchida at [0123] ("Since a FIN flag is set, the end determination infor-mation extraction unit Cl 7 determines that the packet includes information necessary for determining a flow end. Thus, the end determination information extraction unit Cl 7 extracts information for identifying a process rule to be deleted (the ingress port is 2; the source address is 192.168. 1.10; the destination is 192.168.1.1; and the protocol is TCP (the type is Ox.0006)) and stands by until the packet is trans-mitted. Upon receiving a notification that the packet has been transmitted from the packet forwarding unit C16, the end determination information extraction unit Cl 7 further extracts information for identifying a modified process rule from the packet storage unit Cll. Since the IP address is replaced, the extracted information for identifying a modified process rule represents that the source address is 192.168.1. 10; the destination is 192.168.0.10; and the protocol is TCP (the type is 0x0006). The information is used for marking of the reverse flow. The end determination information extrac-tion unit Cl 7 notifies the flow end check unit C18 of the notification that the FIN packet has been received and these items of information (step S202 in FIG. 8).")<br><br>Uchida at [0124] ("Upon receiving the above information from the end determination information extraction unit Cl 7, the flow end check unit C18 checks whether or not a FIN flag is set in a predetermined packet header position (step S203 in FIG. 8). These steps correspond to steps Slll to S114 in FIG. 3.")<br><br>Uchida at [0125] ("At this point, since a FIN packet has been transmit-ted, the flow end check unit C18 uses the information for identifying a process rule to be deleted as a key, extracts the process rule (process rule corresponding to ingress port 2 in FIG. 11) from the process rule storage unit C13, and marks a FIN packet reception flag (steps S204 to S205 in |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 296 of 1100

| No. | '111 Patent Claim 17 | Swenson |
|---|---|---|
| | | FIG. 8). This process corresponds to the internal state update process in step S115 in FIG. 3.")<br><br>Uchida at [0134] ("Referring back to the state transition diagram of TCP connection in FIG. 6, there are two cases where "CLOSED" at the top of FIG. 6 is reached without a state transition involving FIN flags. One case arises when the ses-sion is closed from SYN_SENT, which is reached when a SYN packet in which a SYN flag is marked is transmitted. The other case arises when a timeout is generated. In such case, while the packet forwarding apparatus cannot monitor the closed session, the packet forwarding apparatus can con-firm a timeout in the following way. In the present specific example, a flow end is determined by this timeout.")<br><br>Uchida at [0135] ("n the present specific example, if a SYN/ ACK packet does not flow in a direction opposite to the SYN packet flow direction within a predetermined time (from "SYN_ RCVD" to "SYN_SENT" in FIG. 6), a timeout is determined.")<br><br>Uchida at [0136] ("FIG. 14 is a flow chart illustrating a flow end deter-mination process using a SYN flag. Since the basic operations are the same as those of the above specific example 1, the following description will be made with a focus on the dif-ference.")<br><br>Uchida at [0137] ("In FIG. 14, upon receiving a notification that the packet has been received by the packet reception unit ClO, the end determination information extraction unit Cl 7 refers to the packet storage, unit Cll, monitors a TCP SYN flag, and finds a SYN packet (step S301 in FIG. 14).")<br><br>Uchida at [0138] ("Since a SYN flag is set, the end determination infor-mation extraction unit Cl 7 determines that the packet includes information necessary for determining a flow end. Thus, the end determination information extraction unit Cl 7 extracts information for identifying a process rule to be deleted (the ingress port is 2; the source address is 192.168. 1.10; the destination is 192.168.1.1; and the protocol is TCP (the type is Ox.0006)) and stands by until the packet is trans-mitted. Upon receiving a notification that the packet has been transmitted by the packet forwarding unit C16, the end deter-mination information extraction unit Cl 7 further extracts information for identifying a modified process rule from the packet storage unit Cll. Since the IP address is replaced, the extracted information for |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 297 of 1100

| No. | '111 Patent Claim 17 | Swenson |
|---|---|---|
| | | identifying a process rule repre-sents that the source address is 192.168.1.10; the destination is 192.168.0.10; and the protocol is TCP (the type is 0x0006). The information is used for marking of the reverse flow. The end determination information extraction unit Cl 7 notifies the flow end check unit C18 of the notification that the SYN packet has been received and these items of information (step S302 in FIG. 14).")<br><br>Uchida at [0139] ("Upon receiving the above information from the end determination information extraction unit Cl 7, the flow end check unit C18 checks whether a SYN flag is set in a prede-termined packet header position and an ACK flag is not marked (step S303 in FIG. 14). These steps correspond to steps Slll to S114 in FIG. 3.")<br><br>Uchida at [0148] (" Next, a third specific example in which a flow end determination is executed by using a TCP RST (reset) flag will be described.")<br><br>Uchida at [0149] ("Referring back to the state transition diagram of TCP connection in FIG. 6, there is a transition from "SYN_ RCVD," which is a communication establishment standby state, to "LISTEN," which is a communication standby state. A TCP RST (reset) flag signifies release of connection and retry of communication. Namely, since a RST packet in which this RST flag is set signifies invalidation of communi-cation, by detecting this RST flag, a flow end can be deter-mined.")<br><br>Uchida at [0150] ("FIG. 16 is a first flow chart illustrating a flow end determination process using a RST flag. Since the basic operations are the same as those of the above specific example 1, the following description will be made with a focus on the difference.")<br><br>Uchida at [0151] ("In FIG. 16, upon receiving a notification that the packet has been received by the packet reception unit ClO, the end determination information extraction unit Cl 7 refers to the packet storage unit Cll, monitors a TCP RST flag, and finds a RST packet (step S401 in FIG. 16).")<br><br>Uchida at [0152] ("Since a RST flag is set, the end determination infor-mation extraction unit Cl 7 determines that the packet includes information necessary for determining a flow end. Thus, the end determination information extraction unit Cl 7 extracts information for identifying a process rule to be deleted (the ingress port is 2; the source address is 192.168. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 298 of 1100

| No. | '111 Patent Claim 17 | Swenson |
|---|---|---|
| | | 1.10; the destination is 192.168.1.1; and the protocol is TCP (the type is Ox0006)) and stands by until the packet is trans-mitted. Upon receiving a notification that the packet has been transmitted from the packet forwarding unit C16, the end determination information extraction unit Cl 7 notifies the flow end check unit C18 of the notification that the RST packet has been received and these items of information ( step S402 in FIG. 16).")<br><br>Uchida at [0164] ("For example, in a specific example of the present invention, certain TCP flags are monitored. A single packet forwarding apparatus can monitor these flags in a parallel fashion. For example, after a packet that triggers a flow end is detected, the above process may be allowed to branch to the above FIGS. 8, 14, and 16 (17) to realize parallel monitoring.") |

| No. | '111 Patent Claim 18 | Swenson |
|---|---|---|
| 18[a] | The method according to claim 1, wherein the packet comprises distinct header and payload fields, | Swenson discloses the method according to claim 1, wherein the packet comprises distinct header and payload fields.<br><br>*See supra* at Claim 1, 15[a]. |
| 18[b] | the header comprises at least the first and second entities addresses in the packet network, and | Swenson discloses the header comprises at least the first and second entities addresses in the packet network.<br><br>For example, Swenson discloses source and destination addresses included in the packet flow header.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 299 of 1100

| No. | '111 Patent Claim 18 | Swenson |
|---|---|---|
| | | case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0040] ("The flow analyzer 312 monitors large flows in the network, analyzes collected flow statistics to determine net-work throughput, and accordingly selects flows to be opti-mized. The flow analyzer 312 does not need to see all the flows in order to make an accurate estimate of network con-ditions. The flow analyzer 312 processes the traffic statistics stored in the flow cache 3 22 and user information stored in the subscriber log 324, for example, by associating network flows identified by source IP addresses to a mobile subscriber or user, which is identified by his or her current subscriber ID or device ID. The user flows are also mapped to a congestion level at the current sub-network (e.g., a cell with which the user devices are associated), so that an optimization decision can be made at the beginning of the data transmission.")<br><br>Swenson at [0073] ("FIG. 7 is a block diagram illustrating one embodi-ment of an example of internal components of the flow cache. The flow cache map 700 comprises a plurality of flow cache entries, such as flow cache entries 710 and 712 indexed by a hash. Not shown in the example diagram is a possible linked list behind each flow cache entry which allows chaining of flow cache entries for a given hash index. The hash into the flow cache may be based on source IP address, MAC address, subscriber ID, or other identifier indicative of a given sub-scriber, group of subscribers or subscriber's device.")<br><br>Swenson at [0083] ("When a new flow is observed, flow cache entries are searched by matching source IP address 722 if the subscriber id or other identifiers of the flow are not available. In case of multiple users sharing an IP address, the flow analyzer 312 needs to find patterns or other identifiers in the flows to map them to particular subscribers. Flows without identified sub-scribers are added to the flow cache block under the default user flows 726, which is a default holding place for the new flows. The flow analyzer 312 later will scan through the default user flows that contain cookies or other identifiers that may be used to determine a real user or subscriber associated with the flow. If a flow contains identifiers not associated with an existing real user, a new user or subscriber is created and the user flow block is moved to newly created (or mapped) user or subscriber.") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 300 of 1100

| No. | '111 Patent Claim 18 | Swenson |
|---|---|---|
| 18[c] | wherein the packet-applicable criterion is that the first entity address, the second entity address, or both match a predetermined address or addresses. | Swenson discloses wherein the packet-applicable criterion is that the first entity address, the second entity address, or both match a predetermined address or addresses.<br><br>For example, Swenson discloses the one or more signatures, desired criteria, or conditions associated with a packet flow including matching a source and/or destination address.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0040] ("The flow analyzer 312 monitors large flows in the network, analyzes collected flow statistics to determine net-work throughput, and accordingly selects flows to be opti-mized. The flow analyzer 312 does not need to see all the flows in order to make an accurate estimate of network con-ditions. The flow analyzer 312 processes the traffic statistics stored in the flow cache 3 22 and user information stored in the subscriber log 324, for example, by associating network flows identified by source IP addresses to a mobile subscriber or user, which is identified by his or her current subscriber ID or device ID. The user flows are also mapped to a congestion level at the current sub-network (e.g., a cell with which the user devices are associated), so that an optimization decision can be made at the beginning of the data transmission.")<br><br>Swenson at [0073] ("FIG. 7 is a block diagram illustrating one embodi-ment of an example of internal components of the flow cache. The flow cache map 700 comprises a plurality of flow cache entries, such as flow cache entries 710 and 712 indexed by a hash. Not shown in the example diagram is a possible linked list behind each flow cache entry which allows chaining of flow cache entries for a given hash index. The hash into the flow cache may be |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 301 of 1100

| No. | '111 Patent Claim 18 | Swenson |
|---|---|---|
| | | based on source IP address, MAC address, subscriber ID, or other identifier indicative of a given sub-scriber, group of subscribers or subscriber's device.")<br><br>Swenson at [0075] ("Ideally, a flow can be assigned to the mapped user flows block 726 for a user or subscriber by the user's source IP address. However, in some cases, flows associated with an IP address may often be associated with a group of users or subscribers, but there is not enough information to identify a particular user or subscriber. In these cases, a pseudo sub-scriber id can be assigned in the default user flows block 724 until real users or subscribers are identified as more flows are observed.")<br><br>Swenson at [0083] ("When a new flow is observed, flow cache entries are searched by matching source IP address 722 if the subscriber id or other identifiers of the flow are not available. In case of multiple users sharing an IP address, the flow analyzer 312 needs to find patterns or other identifiers in the flows to map them to particular subscribers. Flows without identified sub-scribers are added to the flow cache block under the default user flows 726, which is a default holding place for the new flows. The flow analyzer 312 later will scan through the default user flows that contain cookies or other identifiers that may be used to determine a real user or subscriber associated with the flow. If a flow contains identifiers not associated with an existing real user, a new user or subscriber is created and the user flow block is moved to newly created (or mapped) user or subscriber.") |

| No. | '111 Patent Claim 19 | Swenson |
|---|---|---|
| 19 | The method according to claim 18, wherein the addresses are Internet Protocol (IP) addresses. | Swenson discloses the method according to claim 18, wherein the addresses are Internet Protocol (IP) addresses.<br><br>For example, Swenson discloses IP addresses used to identify network flows.<br><br>*See supra* at Claim 18.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 302 of 1100

| No. | '111 Patent Claim 19 | Swenson |
|---|---|---|
| | | device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0027] ("However, information on the wireless/cellular user devices 110 side is often not available at the steering device 130 that sits between the cellular network and the wired Internet. For example, there is often no information about the identifiers of the towers associated with the mobile devices 110. Tower association information only broadcasted when the mobile devices first attached to the network. In addition, user devices 110 do not usually report any identification information except their IP addresses. Therefore, monitoring of the network traffic and detection of the congestion is auto-mated and managed by the detector 140 so that network can be optimized for end user's experience without the mobile user's knowledge.")<br><br>Swenson at [0040] ("The flow analyzer 312 monitors large flows in the network, analyzes collected flow statistics to determine net-work throughput, and accordingly selects flows to be opti-mized. The flow analyzer 312 does not need to see all the flows in order to make an accurate estimate of network con-ditions. The flow analyzer 312 processes the traffic statistics stored in the flow cache 3 22 and user information stored in the subscriber log 324, for example, by associating network flows identified by source IP addresses to a mobile subscriber or user, which is identified by his or her current subscriber ID or device ID. The user flows are also mapped to a congestion level at the current sub-network (e.g., a cell with which the user devices are associated), so that an optimization decision can be made at the beginning of the data transmission.")<br><br>Swenson at [0047] (" The subscriber log 324 stores user or subscriber information, such as user or subscriber identifications and their device information. In one embodiment, the subscriber and device information is provided to the subscriber log 324 by the administrators or operators of the carrier or service provider networks. In other embodiments, the subscriber or the device information of the carrier networks (e.g., mobile ISPs) is not available to the network controller 140. This makes bandwidth measurement |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 303 of 1100

| No. | '111 Patent Claim 19 | Swenson |
|---|---|---|
| | | more difficult since multiple users' devices may share a single IP address using the net-work address translation (NAT) protocol. Accordingly, algo-rithms that separate multiple users sharing an IP address can be implemented by the flow analyzer 312 to determine the amount of bandwidth available to individual users.")<br><br>Swenson at [0059] ("In one embodiment, as the steering device 130 monitors network responses, it is looking for flows that match one or more signatures for video and images. When a match-ing flow is detected, the steering device 130 forwards the HTTP request and a portion of the HTTP response to the network controller 140 over the ICAP client interface 404. After receiving the request and the portion of response at the ICAP server interface 406, the flow analyzer 312 of the net-work controller 140 performs a deep flow inspection to deter-mine if the flow is worth bandwidth monitoring and/or user detection. For example, the flow inspection performed by the flow analyzer 312 may determine if the flow indeed contains large or medium object ( e.g., larger than 50 kB), and/or if the source IP address of the flow is from a user or a group of users that are required to be monitored by policies. The flow ana-lyzer 312 may also determine if the flow needs to be opti-mized based on historical flow statistical data.")<br><br>Swenson at [0062] ("Based on the flow statistics stored in the flow cache 322, the network controller 140 can also aggregate the flows associated with a user or subscriber in order to estimate the total available bandwidth occupied by the user or subscriber. In one embodiment, the network controller 140 tracks all the flow cache entries looking for flows originated from a com-mon source IP address or a user device identifier. The flow analyzer 312 of the network controller 140 then attempts to group these flows together to form a flow history for the user or subscriber. The network controller further identifies users or subscribers using two data components in the flow cache entry: the TCP source port and HTTP cookies associated with the flow. Together with the flow history, the network control-ler 140 establish pattern, and identify users or subscribers and stores subscriber information in the subscriber log 324. More details of the flow cache and user mapping are described below with reference to FIG. 7.")<br><br>Swenson at [0073] ("FIG. 7 is a block diagram illustrating one embodi-ment of an example of intern al components of the flow cache. The flow cache map 700 comprises a plurality of flow cache entries, such as flow cache entries 710 and 712 indexed by a hash. Not shown in |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 304 of 1100

| No. | '111 Patent Claim 19 | Swenson |
|---|---|---|
| | | the example diagram is a possible linked list behind each flow cache entry which allows chaining of flow cache entries for a given hash index. The hash into the flow cache may be based on source IP address, MAC address, subscriber ID, or other identifier indicative of a given sub-scriber, group of subscribers or subscriber's device.")<br><br>Swenson at [0074] ("A flow cache block 720 pointed to by the flow cache entry 712 is shown to include information on source IP 722, one or more user flow blocks, which represent a logical group of flows associated with a user, a subscriber, or an entity representing a potential subscriber. Examples of these user flow blocks are default user flows block 724 and mapped user flows block 726. The default user flows block 724 store flows that are not yet associated with any particular user or sub-scriber. If the subscriber id or any other identifiers associated with a particular user is known a-priori, all the flows associ-ated with the particular user or subscriber will be assigned to the mapped user flows block 726. The mapped user flows block 726 also include flows that either have been, or are in the process of being mapped to a user or subscriber by the flow analyzer 312. The mapped user flows block 726 can be indexed using subscriber id.")<br><br>Swenson at [0075] ("Ideally, a flow can be assigned to the mapped user flows block 726 for a user or subscriber by the user's source IP address. However, in some cases, flows associated with an IP address may often be associated with a group of users or subscribers, but there is not enough information to identify a particular user or subscriber. In these cases, a pseudo sub-scriber id can be assigned in the default user flows block 724 until real users or subscribers are identified as more flows are observed.")<br><br>Swenson at [0076] ("An example user flow block 730 that can be included in the default user flows block 724 and the mapped user flows block 726 contains data fields like the subscriber id 732 (pseudo or real) estimated bandwidth 734, a list of all flows 736 associated with the subscriber id 732, and a list of cookie hashes 738 among other related flow information. Each entry in the list of cookie hashes 738 contains one unique cookie seen within the flows. The list of flows 736 includes one or more flow statistics block 740. Each flow statistics block 740 contains the IP flow identifier 742 (e.g., srcIP, dstIP, srcPort, dstPort ), current domain and cookie 7 44, total number of bytes seen in each direction 746, the total number of bytes in each direction as of the last update 748. Not shown in the FIG. 7 includes a list of cookie hashes associated with the flow and an expiration time") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 305 of 1100

| No. | '111 Patent Claim 19 | Swenson |
|---|---|---|
| | | Swenson at [0083] ("When a new flow is observed, flow cache entries are searched by matching source IP address 722 if the subscriber id or other identifiers of the flow are not available. In case of multiple users sharing an IP address, the flow analyzer 312 needs to find patterns or other identifiers in the flows to map them to particular subscribers. Flows without identified sub-scribers are added to the flow cache block under the default user flows 726, which is a default holding place for the new flows. The flow analyzer 312 later will scan through the default user flows that contain cookies or other identifiers that may be used to determine a real user or subscriber associated with the flow. If a flow contains identifiers not associated with an existing real user, a new user or subscriber is created and the user flow block is moved to newly created (or mapped) user or subscriber.") |

| No. | '111 Patent Claim 20 | Swenson |
|---|---|---|
| 20[a] | The method according to claim 1, wherein the packet is an Transmission Control Protocol (TCP) packet that comprises source and destination TCP ports, a TCP sequence number, and a TCP sequence mask fields, and | Swenson discloses the method according to claim 1, wherein the packet is an Transmission Control Protocol (TCP) packet that comprises source and destination TCP ports, a TCP sequence number, and a TCP sequence mask fields.<br><br>For example, Swenson discloses TCP packet flows with TCP port identifier information.<br><br>*See supra* at Claim 1, 17[a].<br><br>Swenson at [0019] ("In one embodiment, an on-demand network moni-toring method is adopted to gather data about network flows as they traverse the network. For example, network flows can be monitored selectively or on-demand based on the types of the content carried in the flows. Furthermore, the network monitoring can also be performed selectively at inline level, as well as out-of-band to improve efficiency. Both TCP and UDP flows are monitored to gather information about the state of the network, such as the average network throughput for each flow and end-to-end latency between, for example, a client device and an origin server providing multimedia con-tent to the client device. For each TCP or UDP flow, the system tracks the number of bytes sent ( and in some embodi-ments acknowledged). In TCP, the current window size may also be tracked. Records on network flows are stored in a flow statistics database, which can be indexed by subscriber iden-tification (ID), cell tower (base station), and network segment etc. As many flow |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 306 of 1100

| No. | '111 Patent Claim 20 | Swenson |
|---|---|---|
| | | records accumulate, this database repre-sents both historical and current network condition and capacity for delivering data. Network throughput can be mea-sured by calculating an average number of bytes delivered over a period of time. Steps may be taken to filter out spurious data from small flows with size less than a certain threshold that, when measured, cause very noisy results in measuring bandwidth and/or latency. For example, any flow having delivery time of less than 500 ms can be filtered.")<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0028] ("In contrast to conventional inline TCP throughput monitoring devices that monitor every single data packets transmitted and received, the network controller 140 is an "out-of-band" computer server that interfaces with the steer-ing device 130 to selectively inspect user flows of interest. The network controller 140 may further identify user flows (e.g., among the flows of interest) for optimization. In one embodiment, the network controller 140 may be imple-mented at the steering device 130 to monitor traffic. In other embodiments, the network controller 140 is coupled to and communicates with the steering device 130 for traffic moni-toring and optimization. When queried by the steering device 130, the network controller 140 determines if a given network flow should be ignored, monitored further or optimized. Opti-mization of a flow is often decided at the beginning of the flow because it is rarely possible to switch to optimized content mid-stream once non-optimized content delivery has begun. However, the network controller 140 may determine that existing flows associated with a particular subscriber or other entity should be optimized. In turn, new flows ( e.g., resulting from seek requests in media, new media requests, resume after pause, etc.) determined to be associated with the entity may be optimized. The network controller 140 uses the net-work state as well as historical traffic |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 307 of 1100

| No. | '111 Patent Claim 20 | Swenson |
|---|---|---|
| | | data in its decision for monitoring and optimization. Knowledge on the current net-work state, such as congestion, deems critical when it comes to data optimization.")<br><br>Swenson at [0044] ("Additionally, historical flow data over a longer term helps the flow analyzer 312 to determine repeating patterns and heat-maps of certain network sections and to predict when they are under congestion. In this case, the flow statis-tics stored in the flow cache 322 can be mapped against traffic categories for analysis, for example, long-term running aver-ages of video flow bandwidth help determine suitability for optimization. Furthermore, estimated bandwidth per user ( or per cell-ID, per tower, or per router) over time may be metrics calculated by the flow analyzer 312 in order to determine short term needs for optimization. For example, the flow analyzer 312 may determine to being optimizing flows asso-ciated with a particular cell-ID (or those flows for identified high-bandwidth users on the cell-ID) in response to a thresh-old number of high-bandwidth users connecting to a same cell tower corresponding to the cell-ID. The reason why flow analyzer 312 selectively monitors large flows lies in the real-ization that TCP statistics for small objects, which make up most web flows, can be misleading and cause huge errors in throughput estimations.")<br><br>Swenson at [0061] ("Once a flow is reported to the network controller 140, a flow cache entry is created for the flow in the flow cache 322. The flow cache entry keeps track of the flow and its associated bandwidth. For a flow that is marked in "continue" mode, each time the steering device 130 forwards a next portion of the flow payload to the network controller 140, the flow cache 3 22 updates the number of bytes for transmitted in the flow. By monitoring the number of bytes per flow over time, the flow analyzer 312 is capable of determining an estimate value of bandwidth associated with flow. Further-more, since the steering device 130 does not have infinite packet buffers, if congestion happens on the network link 416 from the steering device 130 to the user device 110, the TCP congestion control mechanism kicks in at the steering device 130, which may slows down and/or eventually stop receiving data over the network link 413 from origin server 160. During the congestion, the steering device 130 would not forward any data to the network controller 140, since the link 416 is congested and the network controller 140 would not be able to transmit data to the user device 110. Therefore, as an inline element, the network controller 140 can detect network con-gestions and estimate bandwidth associated with any flows of interest selected by the network controller 140. However, in the "continue" mode, the |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 308 of 1100

| No. | '111 Patent Claim 20 | Swenson |
|---|---|---|
| | | network controller 140 does not modify and transform the HTTP messaged it receives over the ICAP interface. The network controller 140 simply updates the flow statistics and returns the video or images to the steering device 130 for transmission to the user device 110.")<br><br>Swenson at [0062] (" Based on the flow statistics stored in the flow cache 322, the network controller 140 can also aggregate the flows associated with a user or subscriber in order to estimate the total available bandwidth occupied by the user or subscriber. In one embodiment, the network controller 140 tracks all the flow cache entries looking for flows originated from a com-mon source IP address or a user device identifier. The flow analyzer 312 of the network controller 140 then attempts to group these flows together to form a flow history for the user or subscriber. The network controller further identifies users or subscribers using two data components in the flow cache entry: the TCP source port and HTTP cookies associated with the flow. Together with the flow history, the network control-ler 140 establish pattern, and identify users or subscribers and stores subscriber information in the subscriber log 324. More details of the flow cache and user mapping are described below with reference to FIG. 7.")<br><br>Swenson at [0073] ("FIG. 7 is a block diagram illustrating one embodi-ment of an example of internal components of the flow cache. The flow cache map 700 comprises a plurality of flow cache entries, such as flow cache entries 710 and 712 indexed by a hash. Not shown in the example diagram is a possible linked list behind each flow cache entry which allows chaining of flow cache entries for a given hash index. The hash into the flow cache may be based on source IP address, MAC address, subscriber ID, or other identifier indicative of a given sub-scriber, group of subscribers or subscriber's device.")<br><br>Swenson at [0084] ("The flow analyzer 312 can also map flows to users (subscribers to the mobile or network service) in the flow cache entries by matching cookie hashes, MAC address ( or any unique device identifiers), or TCP source ports. For example, if two flows share the same source port, it is very likely that they belong to the same user because TCP ports are reused often by an individual user, but not often between users. Furthermore, source ports can also be used to map users when network address translation (NAT) is deployed. In a typical network with NAT configuration, each user is allo-cated a block (e.g., 32) of TCP source ports. A random port number within the block is then picked for each |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 309 of 1100

| No. | '111 Patent Claim 20 | Swenson |
|---|---|---|
| | | new user flows initiated. With this knowledge, all source ports within a block can be aggregated under the same user. In some cases, a user with more than one block of port number assigned, the cookie hashes can be used to link the blocks together.") |
| 20[b] | wherein the packet-applicable criterion is that the source TCP port, the destination TCP port, the TCP sequence number, the TCP sequence mask, or any combination thereof, matches a predetermined value or values. | Swenson discloses wherein the packet-applicable criterion is that the source TCP port, the destination TCP port, the TCP sequence number, the TCP sequence mask, or any combination thereof, matches a predetermined value or values.<br><br>For example, Swenson discloses TCP packet flows with TCP port identifier information that can be used as one of more signatures, desired criteria, or conditions of the packet flow to determine if the flow matches.<br><br>Swenson at [0019] ("In one embodiment, an on-demand network moni-toring method is adopted to gather data about network flows as they traverse the network. For example, network flows can be monitored selectively or on-demand based on the types of the content carried in the flows. Furthermore, the network monitoring can also be performed selectively at inline level, as well as out-of-band to improve efficiency. Both TCP and UDP flows are monitored to gather information about the state of the network, such as the average network throughput for each flow and end-to-end latency between, for example, a client device and an origin server providing multimedia con-tent to the client device. For each TCP or UDP flow, the system tracks the number of bytes sent ( and in some embodi-ments acknowledged). In TCP, the current window size may also be tracked. Records on network flows are stored in a flow statistics database, which can be indexed by subscriber iden-tification (ID), cell tower (base station), and network segment etc. As many flow records accumulate, this database repre-sents both historical and current network condition and capacity for delivering data. Network throughput can be mea-sured by calculating an average number of bytes delivered over a period of time. Steps may be taken to filter out spurious data from small flows with size less than a certain threshold that, when measured, cause very noisy results in measuring bandwidth and/or latency. For example, any flow having delivery time of less than 500 ms can be filtered.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 310 of 1100

| No. | '111 Patent Claim 20 | Swenson |
|---|---|---|
| | | Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0028] ("In contrast to conventional inline TCP throughput monitoring devices that monitor every single data packets transmitted and received, the network controller 140 is an "out-of-band" computer server that interfaces with the steer-ing device 130 to selectively inspect user flows of interest. The network controller 140 may further identify user flows (e.g., among the flows of interest) for optimization. In one embodiment, the network controller 140 may be imple-mented at the steering device 130 to monitor traffic. In other embodiments, the network controller 140 is coupled to and communicates with the steering device 130 for traffic moni-toring and optimization. When queried by the steering device 130, the network controller 140 determines if a given network flow should be ignored, monitored further or optimized. Opti-mization of a flow is often decided at the beginning of the flow because it is rarely possible to switch to optimized content mid-stream once non-optimized content delivery has begun. However, the network controller 140 may determine that existing flows associated with a particular subscriber or other entity should be optimized. In turn, new flows ( e.g., resulting from seek requests in media, new media requests, resume after pause, etc.) determined to be associated with the entity may be optimized. The network controller 140 uses the net-work state as well as historical traffic data in its decision for monitoring and optimization. Knowledge on the current net-work state, such as congestion, deems critical when it comes to data optimization.")<br><br>Swenson at [0044] ("Additionally, historical flow data over a longer term helps the flow analyzer 312 to determine repeating patterns and heat-maps of certain network sections and to predict when they are under congestion. In this case, the flow statis-tics stored in the flow cache 322 can be mapped against traffic categories for analysis, for example, long-term |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 311 of 1100

| No. | '111 Patent Claim 20 | Swenson |
|---|---|---|
| | | running aver-ages of video flow bandwidth help determine suitability for optimization. Furthermore, estimated bandwidth per user ( or per cell-ID, per tower, or per router) over time may be metrics calculated by the flow analyzer 312 in order to determine short term needs for optimization. For example, the flow analyzer 312 may determine to being optimizing flows asso-ciated with a particular cell-ID (or those flows for identified high-bandwidth users on the cell-ID) in response to a thresh-old number of high-bandwidth users connecting to a same cell tower corresponding to the cell-ID. The reason why flow analyzer 312 selectively monitors large flows lies in the real-ization that TCP statistics for small objects, which make up most web flows, can be misleading and cause huge errors in throughput estimations.")<br><br>Swenson at [0061] ("Once a flow is reported to the network controller 140, a flow cache entry is created for the flow in the flow cache 322. The flow cache entry keeps track of the flow and its associated bandwidth. For a flow that is marked in "continue" mode, each time the steering device 130 forwards a next portion of the flow payload to the network controller 140, the flow cache 3 22 updates the number of bytes for transmitted in the flow. By monitoring the number of bytes per flow over time, the flow analyzer 312 is capable of determining an estimate value of bandwidth associated with flow. Further-more, since the steering device 130 does not have infinite packet buffers, if congestion happens on the network link 416 from the steering device 130 to the user device 110, the TCP congestion control mechanism kicks in at the steering device 130, which may slows down and/or eventually stop receiving data over the network link 413 from origin server 160. During the congestion, the steering device 130 would not forward any data to the network controller 140, since the link 416 is congested and the network controller 140 would not be able to transmit data to the user device 110. Therefore, as an inline element, the network controller 140 can detect network con-gestions and estimate bandwidth associated with any flows of interest selected by the network controller 140. However, in the "continue" mode, the network controller 140 does not modify and transform the HTTP messaged it receives over the ICAP interface. The network controller 140 simply updates the flow statistics and returns the video or images to the steering device 130 for transmission to the user device 110.")<br><br>Swenson at [0062] (" Based on the flow statistics stored in the flow cache 322, the network controller 140 can also aggregate the flows associated with a user or subscriber in order to |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 312 of 1100

| No. | '111 Patent Claim 20 | Swenson |
|---|---|---|
| | | estimate the total available bandwidth occupied by the user or subscriber. In one embodiment, the network controller 140 tracks all the flow cache entries looking for flows originated from a com-mon source IP address or a user device identifier. The flow analyzer 312 of the network controller 140 then attempts to group these flows together to form a flow history for the user or subscriber. The network controller further identifies users or subscribers using two data components in the flow cache entry: the TCP source port and HTTP cookies associated with the flow. Together with the flow history, the network control-ler 140 establish pattern, and identify users or subscribers and stores subscriber information in the subscriber log 324. More details of the flow cache and user mapping are described below with reference to FIG. 7.")<br><br>Swenson at [0073] ("FIG. 7 is a block diagram illustrating one embodi-ment of an example of internal components of the flow cache. The flow cache map 700 comprises a plurality of flow cache entries, such as flow cache entries 710 and 712 indexed by a hash. Not shown in the example diagram is a possible linked list behind each flow cache entry which allows chaining of flow cache entries for a given hash index. The hash into the flow cache may be based on source IP address, MAC address, subscriber ID, or other identifier indicative of a given sub-scriber, group of subscribers or subscriber's device.")<br><br>Swenson at [0084] ("The flow analyzer 312 can also map flows to users (subscribers to the mobile or network service) in the flow cache entries by matching cookie hashes, MAC address ( or any unique device identifiers), or TCP source ports. For example, if two flows share the same source port, it is very likely that they belong to the same user because TCP ports are reused often by an individual user, but not often between users. Furthermore, source ports can also be used to map users when network address translation (NAT) is deployed. In a typical network with NAT configuration, each user is allo-cated a block (e.g., 32) of TCP source ports. A random port number within the block is then picked for each new user flows initiated. With this knowledge, all source ports within a block can be aggregated under the same user. In some cases, a user with more than one block of port number assigned, the cookie hashes can be used to link the blocks together.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 313 of 1100

| No. | '111 Patent Claim 21 | Swenson |
|---|---|---|
| 21 | The method according to claim 1, wherein the packet network comprises a Wide Area Network (WAN), Local Area Network (LAN), the Internet, Metropolitan Area Network (MAN), Internet Service Provider (ISP) backbone datacenter network, or inter - datacenter network. | Swenson discloses the method according to claim 1, wherein the packet network comprises a Wide Area Network (WAN), Local Area Network (LAN), the Internet, Metropolitan Area Network (MAN), Internet Service Provider (ISP) backbone datacenter network, or inter - datacenter network.<br><br>For example, Swenson discloses a service provider network comprised of an Internet service provider.<br><br>*See supra* at Claim 1.<br><br>Swenson at Abstract ("A system and a method are disclosed for selectively monitor-ing traffic in a service provider network. The system receives a notice for a beginning of a network data flow, which responds to a request from a user device for content at an origin server. The system then determines whether to monitor the data flow from the origin server to the user device. If so determined, the system collects statistic information of the data flow and stores the statistic information to a flow record in a database. The system also maps the flow record to a subscriber of the service provider network by analyzing the statistic information of the data flow and estimates bandwidth provided to the data flow by the service provider's network based on the analysis of the statistic information of the data flow.")<br><br>Swenson at [0005] ("Mobile devices, such as smart phones and tablets, have become prevalent in recent years. Given the fast advance in mobile computing power and far-reaching wireless Inter-net access, more and more users view streamed videos on their mobile devices. The detection of network congestion has become increasingly important for network operators attempting to maximize user experience on the network. Even as network operators are ever increasing the capacity of their networks, the demand for bandwidth is growing at an even faster pace. Managing network growth and dealing with con-gestion in the infrastructure is particularly important in the mobile space because of the high cost of radio spectrum and radio access network (RAN) equipment utilized by wireless mobile networks. These high costs prevent mobile service providers from engineering excess capacity into each net-work access point through the purchase of additional RAN infrastructure. The same situation can, however, also happens to other types of network infrastructure.") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 314 of 1100

| No. | '111 Patent Claim 21 | Swenson |
|---|---|---|
| | | Swenson at [0006] ("Existing network elements can give operators a view into the current state of traffic in their network, but they do not provide a measure of "goodness," i.e., how much elasticity is left or how much more data can the network handle. This measure is important for multimedia content delivery since a good user experience usually depends on the network's ability to deliver data in a reliable and sustainable fashion. A minimum data rate is required to prevent stalling and re-buffering during the streaming of multimedia content, hence ensuring sufficient bandwidth is important to quality of experience. Typically, multimedia content providers are suf-ficiently equipped to deliver multimedia content at levels far beyond the capabilities of wireless infrastructure. Hence, the burden falls on wireless service providers to implement net-work data optimization to ease the traffic burden and maxi-mize the experience of each and every user on the network. Currently, however, mobile service providers are often forced to use very coarse tools that have little visibility into which network segments are congested and tend to apply optimiza-tion to flows that may not need any optimization.")<br><br>Swenson at [0007] ("Typically, mobile service providers use inline net-work appliances that monitor every bit of subscriber traffic in order to make estimates of network throughput. This puts a huge burden on the system since it must scale to handle hundreds of thousands to millions of network requests per second through a single network access point. Furthermore, network service providers often must utilize these monitoring techniques on a micro-scale ( e.g., per RAN equipment instal-lation) in order to react to the condition of the network, which results in increased cost. In addition, a large portion of web traffic consists of small object requests, which can obscure network monitoring at any level due to their short lifetime and bursty characteristics.")<br><br>Swenson at [0024] ("A network efficiency strategy that aspires to keep capital expenditure from outpacing revenues has to be bal-anced with demands from consumers for better user experi-ences that rely increasingly on higher data usage. Today, mobile operators are employing a variety of tools to manage capacity including data usage caps, Wi-Fi offload and intel-ligent optimization. The environment 100 demonstrates such a solution that provides a unified foundation with deep ses-sion intelligence, integrated services management, and dynamic adaptability to fit any service offering. Together, the network controller 140 and the video optimizer 150 deliver a world-class media optimization |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 315 of 1100

| No. | '111 Patent Claim 21 | Swenson |
|---|---|---|
| | | solution that brings a surgical capacity advantage to wireless operators as well as Internet service providers with better peak capacity savings than alter-native solutions.")<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0047] ("The subscriber log 324 stores user or subscriber information, such as user or subscriber identifications and their device information. In one embodiment, the subscriber and device information is provided to the subscriber log 324 by the administrators or operators of the carrier or service provider networks. In other embodiments, the subscriber or the device information of the carrier networks (e.g., mobile ISPs) is not available to the network controller 140. This makes bandwidth measurement more difficult since multiple users' devices may share a single IP address using the net-work address translation (NAT) protocol. Accordingly, algo-rithms that separate multiple users sharing an IP address can be implemented by the flow analyzer 312 to determine the amount of bandwidth available to individual users.") |

| No. | '111 Patent Claim 22 | Swenson |
|---|---|---|
| 22 | The method according to claim 1, wherein the first entity is a server device and the second entity is a client device, or wherein the first entity is a client | Swenson discloses the method according to claim 1, wherein the first entity is a server device and the second entity is a client device, or wherein the first entity is a client device and the second entity is a server device.<br><br>For example, Swenson discloses a user device and an origin server.<br><br>*See supra* at Claim 1. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 316 of 1100

| No. | '111 Patent Claim 22 | Swenson |
|---|---|---|
| | device and the second entity is a server device.. | Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0030] ("The video optimizer 150 is a computer server that provides video and image optimization and delivers opti-mized video and image content to the user devices 110 via the network 120. The video and image optimization is an on-demand service provided through the transcoding of the video and image content. For example, when a user device attempts to retrieve video from the origin server 160, the network controller 140 may decide that the flow meets certain criteria for content optimization. The network controller 140 then redirected the user devices 110 to the video optimizer 150 to retrieve the optimized content. The video optimizer 150 receives information in the redirect request from the user devices 110 or from the network controller 140 about the video or image content to be optimized and retrieve the video or image content from the corresponding origin server 160 for optimization and subsequent delivery to the user devices 110.")<br><br>Swenson at [0032] ("The video optimizer 150 and the origin server 160 are typically formed of one or more computers. While only one server of each video optimizer 150 and origin server 160 is shown in the environment 100 of FIG. 1, different embodi-ments may include multiple web servers and video servers operated by a single entity or multiple entities. In other embodiments, a single server may also provide different func-tionalities, such as delivering web content as a web server, as well as serving optimized video content.")<br><br>Swenson at [0034] ("The machine may be a server computer, a client computer, a personal computer (PC), a tablet PC, a set-top box (STB), a personal digital assistant (PDA), a cellular tele-phone, a smart phone, a web appliance, a network router, switch or bridge, or |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 317 of 1100

| No. | '111 Patent Claim 22 | Swenson |
|---|---|---|
| | | any machine capable of executing instructions 224 ( sequential or otherwise) that specify actions to be taken by that machine. Further, while only a single machine is illustrated, the term "machine" shall also be taken to include any collection of machines that individually or jointly execute instructions 224 to perform any one or more of the methodologies discussed herein.")<br><br>Swenson at [0055] ("The video optimizer redirector 318 generates a redi-rect request to a URL pointing to the video optimizer 150 if the video is deemed to be transcoded. In one embodiment, the URL may contain at least one of a video resolution, a video bit rate, a video frame rate divisor, an audio sample rate and number of channels, an audio bit rate, a source URL, a user agent of a client, a source domain cookie and any other authentication data by the video optimizer 150. The video optimizer redirector 318 rewrites the original response with the HTTP redirect and sets the location header to the new URL. This causes the user devices 110 to issue a new request to the video optimizer 150. The video optimizer redirector 318 also has the logic to look for incoming URLs generated by itself so that they are not intercepted again.")<br><br>Swenson at [0058] ("Referring now to FIG. 4A, network traffic flows from the user device 110 through the steering device 130 and arrive at the origin server 160 over the network request path. For example, a browser on the user device 110 may request web content from the origin server 160. A HTTP request message initiated at the user device 110 is forwarded to the steering device 130 over the network link 411. A data switch 402 inside the steering device 130 then relays the request message to the origin server 160 over the network link 412. On the opposite direction, network traffic originated from the origin server 160 flows through the steering device 130 back to the user device 110 over the network response path. For example, the origin server 160 responds to the user request by sending web content over the network link 413 to the steering device 130, which forwards the web content to the user device 110 over the network link 416. Note that the network links 411 and 416 are two opposite directions on the same physical link, so are the network link pair 414 and 415. On the other hand, the network link pair 412 and 413 may or may not share the same network path because traffic between the steering device 130 and origin server 160 on opposite directions may be routed differently over one or more routers.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 318 of 1100

| No. | '111 Patent Claim 22 | Swenson |
|---|---|---|
| | | Swenson at [0070] ("Once the user device 110 receives the HTTP redirect request 620, the user device 110 sends the request over the network to the video optimizer 150. In one embodiment, the network controller 140 monitors the traffic and/or requests from the client device 110 as the HTTP redirect request 620 is routed to the video optimizer 150. In such a configuration, the video optimizer 150 only sees requests for video files that need to be transcoded (i.e., optimized) and are associated with a HTTP redirect request 620. As such, the video optimizer 150 is not burdened with all the requests generated by a user device 110.")<br><br>Swenson at [0095] ("Certain embodiments are described herein as including logic or a number of components, modules, or mechanisms. Modules may constitute either software mod-ules ( e.g., code embodied on a machine-readable medium or in a transmission signal) or hardware modules. A hardware module is tangible unit capable of performing certain opera-tions and may be configured or arranged in a certain manner. In example embodiments, one or more computer systems ( e.g., a standalone, client or server computer system) or one or more hardware modules of a computer system (e.g., a pro-cessor or a group of processors 102) may be configured by software (e.g., an application or application portion) as a hardware module that operates to perform certain operations as described herein.") |

| No. | '111 Patent Claim 23 | Swenson |
|---|---|---|
| 23[a] | The method according to claim 22, wherein the server device comprises a web server, and | Swenson discloses the method according to claim 22, wherein the server device comprises a web server.<br><br>For example, Swenson discloses servers that's provide different functionalities, such as delivering web content as a web server.<br><br>*See supra* at Claim 22.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 319 of 1100

| No. | '111 Patent Claim 23 | Swenson |
|---|---|---|
| | | inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0030] ("The video optimizer 150 is a computer server that provides video and image optimization and delivers opti-mized video and image content to the user devices 110 via the network 120. The video and image optimization is an on-demand service provided through the transcoding of the video and image content. For example, when a user device attempts to retrieve video from the origin server 160, the network controller 140 may decide that the flow meets certain criteria for content optimization. The network controller 140 then redirected the user devices 110 to the video optimizer 150 to retrieve the optimized content. The video optimizer 150 receives information in the redirect request from the user devices 110 or from the network controller 140 about the video or image content to be optimized and retrieve the video or image content from the corresponding origin server 160 for optimization and subsequent delivery to the user devices 110.")<br><br>Swenson at [0032] ("The video optimizer 150 and the origin server 160 are typically formed of one or more computers. While only one server of each video optimizer 150 and origin server 160 is shown in the environment 100 of FIG. 1, different embodi-ments may include multiple web servers and video servers operated by a single entity or multiple entities. In other embodiments, a single server may also provide different func-tionalities, such as delivering web content as a web server, as well as serving optimized video content.")<br><br>Swenson at [0034] ("The machine may be a server computer, a client computer, a personal computer (PC), a tablet PC, a set-top box (STB), a personal digital assistant (PDA), a cellular tele-phone, a smart phone, a web appliance, a network router, switch or bridge, or any machine capable of executing instructions 224 ( sequential or otherwise) that specify actions to be taken by that machine. Further, while only a single machine is illustrated, the term "machine" shall also be taken to include any collection of machines that individually or jointly execute instructions 224 to perform any one or more of the methodologies discussed herein.") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 320 of 1100

| No. | '111 Patent Claim 23 | Swenson |
|---|---|---|
| | | Swenson at [0058] ("Referring now to FIG. 4A, network traffic flows from the user device 110 through the steering device 130 and arrive at the origin server 160 over the network request path. For example, a browser on the user device 110 may request web content from the origin server 160. A HTTP request message initiated at the user device 110 is forwarded to the steering device 130 over the network link 411. A data switch 402 inside the steering device 130 then relays the request message to the origin server 160 over the network link 412. On the opposite direction, network traffic originated from the origin server 160 flows through the steering device 130 back to the user device 110 over the network response path. For example, the origin server 160 responds to the user request by sending web content over the network link 413 to the steering device 130, which forwards the web content to the user device 110 over the network link 416. Note that the network links 411 and 416 are two opposite directions on the same physical link, so are the network link pair 414 and 415. On the other hand, the network link pair 412 and 413 may or may not share the same network path because traffic between the steering device 130 and origin server 160 on opposite directions may be routed differently over one or more routers.") |
| 23[b] | wherein the client device comprises a smartphone, a tablet computer, a personal computer, a laptop computer, or a wearable computing device. | Swenson discloses wherein the client device comprises a smartphone, a tablet computer, a personal computer, a laptop computer, or a wearable computing device.<br><br>For example, Swenson discloses a user device that may be a client computer, a personal computer (PC), a tablet PC, a set-top box (STB), a personal digital assistant (PDA), a cellular tele-phone, a smart phone, a web appliance, etc.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 321 of 1100

| No. | '111 Patent Claim 23 | Swenson |
|---|---|---|
| | | Swenson at [0034] ("The machine may be a server computer, a client computer, a personal computer (PC), a tablet PC, a set-top box (STB), a personal digital assistant (PDA), a cellular tele-phone, a smart phone, a web appliance, a network router, switch or bridge, or any machine capable of executing instructions 224 ( sequential or otherwise) that specify actions to be taken by that machine. Further, while only a single machine is illustrated, the term "machine" shall also be taken to include any collection of machines that individually or jointly execute instructions 224 to perform any one or more of the methodologies discussed herein.")<br><br>Swenson at [0058] ("Referring now to FIG. 4A, network traffic flows from the user device 110 through the steering device 130 and arrive at the origin server 160 over the network request path. For example, a browser on the user device 110 may request web content from the origin server 160. A HTTP request message initiated at the user device 110 is forwarded to the steering device 130 over the network link 411. A data switch 402 inside the steering device 130 then relays the request message to the origin server 160 over the network link 412. On the opposite direction, network traffic originated from the origin server 160 flows through the steering device 130 back to the user device 110 over the network response path. For example, the origin server 160 responds to the user request by sending web content over the network link 413 to the steering device 130, which forwards the web content to the user device 110 over the network link 416. Note that the network links 411 and 416 are two opposite directions on the same physical link, so are the network link pair 414 and 415. On the other hand, the network link pair 412 and 413 may or may not share the same network path because traffic between the steering device 130 and origin server 160 on opposite directions may be routed differently over one or more routers.") |

| No. | '111 Patent Claim 24 | Swenson |
|---|---|---|
| 24 | The method according to claim 22, wherein the communication between the network node and the controller is based on, or uses, a standard protocol. | Swenson discloses the method according to claim 22, wherein the communication between the network node and the controller is based on, or uses, a standard protocol.<br><br>For example, Swenson discloses communication between the network controller and steering device using the Internet content adaption protocol (ICAP). Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Swenson is found to not meet this limitation, wherein the communication between the network node and the controller is based on, or uses, a standard |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 322 of 1100

| No. | '111 Patent Claim 24 | Swenson |
|---|---|---|
| | | protocol would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>*See supra* at Claim 22.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0056] ("FIGS. 4A and 4B each illustrates one embodiment of an example working mode of the network controller for providing selective on-demand real-time network monitoring and subscriber identification. Shown with the network con-troller 140 are the user device 110, the steering device 130, and the origin server 160. The network controller 140 is coupled to the steering device 130 through the steering device interface 316. In one embodiment, the network controller 140 and the steering device 130 communicate with each other using the Internet content adaption protocol (ICAP). The steering device interface 316 executes an ICAP server 406, which interacts with an ICAP client 404 running on the steer-ing device 130. Similar or different protocols may be used for communication between the network controller 140 and the steering device 130 in other embodiments.")<br><br>Swenson at [0057] ("The Internet content adaption protocol is a light-weight protocol aimed at executing a simple remote proce-dure call on HTTP messages. ICAP leverages edge-based devices to help deliver value-added services using transparent HTTP proxy caches. Content adaptation refers to performing the particular value added service, such as content manipula-tion or other processing, for the associated HTTP client request/response. ICAP clients pass HTTP messages to ICAP servers for transformation or other processing. In turn, |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 323 of 1100

| No. | '111 Patent Claim 24 | Swenson |
|---|---|---|
| | | the ICAP server executes its transformation service on the HTTP messages and sends back responses to the ICAP client. At the core of this process is a cache that can proxy all client trans-actions and process them through ICAP servers, which may focus on specific functions, such as ad insertion, virus scan-ning, content translation, language translation, or content fil-tering. ICAP servers, such as those utilized by the network controller 140, handle these tasks to off-load value-added services from network devices including an ICAP client, such as the steering device 130. By offloading value added services from the steering device 130, processing infrastructure (e.g., optimization services and network controllers) may be scaled independent from the steering devices handling raw HTTP throughput.")<br><br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Swenson in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 24 of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references.<br><br>For example, Kempf discloses a packet network using the OpenFlow protocol, which is used in Software Defined Networks for communication between network device and a controller.<br><br>Kempf at [0004] ("The GPRS tunneling protocol (GTP) is an important communication protocol utilized within the GPRS core net-work. GTP enables end user devices ( e.g., cellular phones) in a GSM network to move from place to place while continuing to connect to the Internet. The end user devices are connected to other devices through a gateway GPRS support node (GGSN). The GGSN tracks the end user device's data from the end user device's serving GPRS support node (GGSN) that is handling the session originating from the end user device.")<br><br>Kempf at [0006] ("A method implements a control plane of an evolved packet core (EPC) of a third generation partnership project (3GPP) long term evolution (LTE) network in a cloud com-puting system. The cloud computing system includes a cloud manager and a controller. The controller executes a plurality of control plane modules. The control plane communicates with the data plane of the EPC implemented in a plurality of network elements of the 3GPP LTE network through a control protocol. The EPC with the control |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 324 of 1100

| No. | '111 Patent Claim 24 | Swenson |
|---|---|---|
| | | plane implemented in the cloud computing system utilizes resources more efficiently than an architecture with the control plane implemented in the plurality of network elements of the 3GPP LTE network. The method comprises the steps of initializing the plurality of control plane modules of the EPC within the controller. Each control plane module in the plurality of control plane modules is initialized as a separate virtual machine by the cloud man-ager. Each control plane module provides a set of control plane functions for managing the data plane. The cloud man-ager monitors resource utilization of each control plane mod-ule and the control plane traffic handled by each control plane module. The cloud manager detects a threshold level of resource utilization or traffic load for one of the plurality of control plane modules of the EPC. A new control plane mod-ule is initialized as a separate virtual machine by the cloud manager in response to detecting the threshold level. The new control plane module shares the load of the one of the plural-ity of control plane modules and signals the plurality of net-work elements in the data plane to establish flow rules and actions to establish differential routing of flows in the data plane using the control protocol, wherein the control protocol is an OpenFlow protocol, and wherein flow matches are encoded using an extensible match structure in which the flow match is encoded as a type-length-value (TLV).")<br><br>Kempf at [0007] ("A cloud computer system implements a control plane of an evolved packet core (EPC) of a third generation partnership project (3GPP) long term evolution (LTE) net-work. The control plane communicates with the data plane of the EPC that is implemented in a plurality of network ele-ments of the 3GPP LTE network through a control protocol. The EPC with the control plane implemented in the cloud computing system utilizes resources more efficiently than an architecture with the control plane implemented in the plu-rality of network elements of the 3GPP LTE network. The cloud computing system, comprises a controller configured to execute a plurality of control plane modules of the EPC, each control plane module configured to provide a set of control plane functions for managing the data plane and to signal the plurality of network elements in the data plane to establish flow rules and actions to establish differential rout-ing of flows in the data plane using the control protocol, wherein the control protocol is an OpenFlow protocol, and wherein flow matches are encoded using an extensible match structure in which the flow match is encoded as a type-length-value (TLV) and a cloud manager communicatively coupled to the controller. The cloud manager is configured to initialize each of the plurality of control plane modules within the controller as a separate virtual machine, monitor |

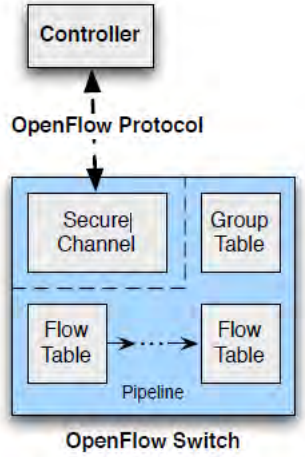


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 325 of 1100

| No. | '111 Patent Claim 24 | Swenson |
|---|---|---|
| | | resource utilization of each control plane module and the control plane traffic handled by each control plane module, detect whether a threshold level ofresource utilization or traffic load has been reached by any of the plurality of control plane modules of the EPC, and initialize a new control plane module as a separate virtual machine in response to detecting the threshold level, the new control plane module to share the load of the one of the plurality of control plane modules that exceeded the threshold level.")<br><br>Kempf at [0038] ("Implementing the control plane of an EPC in a cloud computing facility and the data plane of the EPC using a set of OpenFlow switches, as well as managing communication between the control plane and the dataplane using the Open-Flow protocol (e.g., OpenFlow 1.1), creates a problem that the OpenFlow protocol does not support GTP or GTP tunnel endpoint identifier (TEID) routing, which is necessary for implementing the dataplane of the EPC")<br><br>Kempf at [0039] ("The embodiments of the invention overcome these disadvantages of the prior art. The disadvantages of the prior art are avoided by splitting the control plane and the data plane for the EPC architecture and to implement the control plane by deploying the EPC control plane entities in a cloud computing facility, while the data plane is implemented by a distributed collection of OpenFlow switches. The OpenFlow protocol is used to connect the two, with enhancements to support GTP routing. While the EPC architecture already has a split between the control plane and the data plane, in the sense that the serving gateway (S-GW) and the PDN gateway (P-GW) are data plane entities while the MME, PCRF, and home subscriber server (HSS) are control plane entities, this split was made at the level of the mobility management pro-tocol, GTP.")<br><br>Kempf at [0040] ("The standard EPC architecture assumes a standard routed IP network for transport on top of which the mobile network entities and protocols are implemented. The enhanced EPC architecture described herein is instead at the level ofIP routing and media access control (MAC) switch-ing. Instead of using L2 routing and L3 internal gateway protocols to distribute IP routing and managing Ethernet and IP routing as a collection of distributed control entities, L2 and L3 routing management is centralized in a cloud facility and the routing is controlled from the cloud facility using the OpenFlow protocol. As used herein, the "OpenFlow proto-col" refers to the OpenFlow network protocol and switching specification defined in the OpenFlow Switch Specification at www.openflowswitch.org a |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 326 of 1100

| No. | '111 Patent Claim 24 | Swenson |
|---|---|---|
| | | web site hosted by Stanford Uni-versity. As used herein, an "OpenFlow switch" refers to a network element implementing the OpenFlow protocol.)<br><br>Kempf at [0044] ("FIG. 1 is a diagram of one embodiment of an example network with an OpenFlow switch, conforming to the OpenFlow 1.0 specification. The OpenFlow 1.0 protocol enables a controller 101 to connect to an OpenFlow 1.0 enabled switch 109 using a secure channel 103 and control a single forwarding table 107 in the switch 109. The controller 101 is an external software component executed by a remote computing device that enables a user to configure the Open-Flow 1.0 switch 109. The secure channel 103 can be provided by any type of network including a local area network (LAN) or a wide area network (WAN), such as the Internet.")<br><br>As another example, OpenFlow is a standard protocol used in SDNs to communicate between an OpenFlow switch and controller.<br><br>OpenFlow at 6-7<br>  <br><br>Figure 1: Main components of an OpenFlow switch. |

| No. | '111 Patent Claim 24 | Swenson |
|---|---|---|
| | | 2 Switch Components<br><br>An OpenFlow Switch consists of one or more *flow tables* and a *group table*, which perform packet lookups and forwarding, and an *OpenFlow channel* to an external controller (Figure 1). The switch communicates with the controller and the controller manages the switch via the OpenFlow protocol.<br><br>Using the OpenFlow protocol, the controller can add, update, and delete *flow entries* in flow tables, both reactively (in response to packets) and proactively. Each flow table in the switch contains a set of flow entries; each flow entry consists of *match fields*, *counters*, and a set of *instructions* to apply to matching packets (see 5.2).<br><br>Matching starts at the first flow table and may continue to additional flow tables (see 5.1). Flow entries match packets in priority order, with the first matching entry in each table being used (see 5.3). If a matching entry is found, the instructions associated with the specific flow entry are executed. If no match is found in a flow table, the outcome depends on configuration of the table-miss flow entry: for example, the packet may be forwarded to the controller over the OpenFlow channel, dropped, or may continue to the next flow table (see 5.4).<br><br>Instructions associated with each flow entry either contain actions or modify pipeline processing (see 5.9). Actions included in instructions describe packet forwarding, packet modification and group table |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 328 of 1100

| No. | '111 Patent Claim 24 | Swenson |
|---|---|---|
| | | processing. Pipeline processing instructions allow packets to be sent to subsequent tables for further processing and allow information, in the form of metadata, to be communicated between tables. Table pipeline processing stops when the instruction set associated with a matching flow entry does not specify a next table; at this point the packet is usually modified and forwarded (see 5.10).<br><br>Flow entries may forward to a *port*. This is usually a physical port, but it may also be a logical port defined by the switch or a reserved port defined by this specification (see 4.1). Reserved ports may specify generic forwarding actions such as sending to the controller, flooding, or forwarding using non-OpenFlow methods, such as "normal" switch processing (see 4.5), while switch-defined logical ports may specify link aggregation groups, tunnels or loopback interfaces (see 4.4).<br><br>Actions associated with flow entries may also direct packets to a group, which specifies additional processing (see 5.6). Groups represent sets of actions for flooding, as well as more complex forwarding semantics (e.g. multipath, fast reroute, and link aggregation). As a general layer of indirection, groups also enable multiple flow entries to forward to a single identifier (e.g. IP forwarding to a common next hop). This abstraction allows common output actions across flow entries to be changed efficiently.<br><br>The group table contains group entries; each group entry contains a list of *action buckets* with specific semantics dependent on group type (see 5.6.1). The actions in one or more action buckets are applied to packets sent to the group.<br><br>Switch designers are free to implement the internals in any way convenient, provided that correct match and instruction semantics are preserved. For example, while a flow entry may use an all group to forward to multiple ports, a switch designer may choose to implement this as a single bitmask within the hardware forwarding table. Another example is matching; the pipeline exposed by an OpenFlow switch may be physically implemented with a different number of hardware tables. |

| No. | '111 Patent Claim 27 | Swenson |
|---|---|---|
| 27 | The method according to claim 1, wherein the network node comprises a router, a switch, or a bridge. | Swenson discloses the method according to claim 1, wherein the network node comprises a router, a switch, or a bridge.<br><br>For example, Swenson discloses a steering device that may be a router.<br><br>*See supra* at Claim 1.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 329 of 1100

| No. | '111 Patent Claim 27 | Swenson |
|---|---|---|
| | | device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0029] ("As a flow is sent to the network controller 140 for inspection, historical network traffic data stored at the net-work controller 140 may be searched. The historical network traffic data includes information such as subscriber informa-tion, the cell towers to which the user devices attached, rout-ers through which the traffic is passing, geography regions, the backhaul segments, and time-of-day of the flows. For example, in a mobile network, the cell tower to which a user device is attached can be most useful, since it is the location where most congestion occurs due to limited bandwidth and high cost of the radio access network infrastructure. The network controller 140 looks into the historical traffic data for the average of the bandwidth per user at the particular cell tower. The network controller 140 can then estimate the amount of bandwidth or degree of congestion for the new flow based on the historical record.") |

| No. | '111 Patent Claim 28 | Swenson |
|---|---|---|
| 28 | The method according to claim 1, wherein the packet network is an Internet Protocol (IP) network, and the packet is an IP packet. | Swenson discloses the method according to claim 1, wherein the packet network is an Internet Protocol (IP) network, and the packet is an IP packet.<br><br>For example, Swenson discloses packet flows in a network with IP addresses.<br><br>*See supra* at Claim 1.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 330 of 1100

| No. | '111 Patent Claim 28 | Swenson |
|---|---|---|
| | | interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0027] ("However, information on the wireless/cellular user devices 110 side is often not available at the steering device 130 that sits between the cellular network and the wired Internet. For example, there is often no information about the identifiers of the towers associated with the mobile devices 110. Tower association information only broadcasted when the mobile devices first attached to the network. In addition, user devices 110 do not usually report any identification information except their IP addresses. Therefore, monitoring of the network traffic and detection of the congestion is auto-mated and managed by the detector 140 so that network can be optimized for end user's experience without the mobile user's knowledge.")<br><br>Swenson at [0040] ("The flow analyzer 312 monitors large flows in the network, analyzes collected flow statistics to determine net-work throughput, and accordingly selects flows to be opti-mized. The flow analyzer 312 does not need to see all the flows in order to make an accurate estimate of network con-ditions. The flow analyzer 312 processes the traffic statistics stored in the flow cache 3 22 and user information stored in the subscriber log 324, for example, by associating network flows identified by source IP addresses to a mobile subscriber or user, which is identified by his or her current subscriber ID or device ID. The user flows are also mapped to a congestion level at the current sub-network (e.g., a cell with which the user devices are associated), so that an optimization decision can be made at the beginning of the data transmission.")<br><br>Swenson at [0047] (" The subscriber log 324 stores user or subscriber information, such as user or subscriber identifications and their device information. In one embodiment, the subscriber and device information is provided to the subscriber log 324 by the administrators or operators of the carrier or service provider networks. In other embodiments, the subscriber or the device information of the carrier networks (e.g., mobile ISPs) is not available to the network controller 140. This makes bandwidth measurement more difficult since multiple users' devices may share a single IP address using the net-work address translation (NAT) protocol. Accordingly, algo-rithms that separate multiple users |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 331 of 1100

| No. | '111 Patent Claim 28 | Swenson |
|---|---|---|
| | | sharing an IP address can be implemented by the flow analyzer 312 to determine the amount of bandwidth available to individual users.")<br><br>Swenson at [0059] ("In one embodiment, as the steering device 130 monitors network responses, it is looking for flows that match one or more signatures for video and images. When a match-ing flow is detected, the steering device 130 forwards the HTTP request and a portion of the HTTP response to the network controller 140 over the ICAP client interface 404. After receiving the request and the portion of response at the ICAP server interface 406, the flow analyzer 312 of the net-work controller 140 performs a deep flow inspection to deter-mine if the flow is worth bandwidth monitoring and/or user detection. For example, the flow inspection performed by the flow analyzer 312 may determine if the flow indeed contains large or medium object ( e.g., larger than 50 kB), and/or if the source IP address of the flow is from a user or a group of users that are required to be monitored by policies. The flow ana-lyzer 312 may also determine if the flow needs to be opti-mized based on historical flow statistical data.")<br><br>Swenson at [0062] ("Based on the flow statistics stored in the flow cache 322, the network controller 140 can also aggregate the flows associated with a user or subscriber in order to estimate the total available bandwidth occupied by the user or subscriber. In one embodiment, the network controller 140 tracks all the flow cache entries looking for flows originated from a com-mon source IP address or a user device identifier. The flow analyzer 312 of the network controller 140 then attempts to group these flows together to form a flow history for the user or subscriber. The network controller further identifies users or subscribers using two data components in the flow cache entry: the TCP source port and HTTP cookies associated with the flow. Together with the flow history, the network control-ler 140 establish pattern, and identify users or subscribers and stores subscriber information in the subscriber log 324. More details of the flow cache and user mapping are described below with reference to FIG. 7.")<br><br>Swenson at [0073] ("FIG. 7 is a block diagram illustrating one embodi-ment of an example of intern al components of the flow cache. The flow cache map 700 comprises a plurality of flow cache entries, such as flow cache entries 710 and 712 indexed by a hash. Not shown in the example diagram is a possible linked list behind each flow cache entry which allows chaining of flow cache entries for a given hash index. The hash into the flow cache may be |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 332 of 1100

| No. | '111 Patent Claim 28 | Swenson |
|---|---|---|
| | | based on source IP address, MAC address, subscriber ID, or other identifier indicative of a given sub-scriber, group of subscribers or subscriber's device.")<br><br>Swenson at [0074] ("A flow cache block 720 pointed to by the flow cache entry 712 is shown to include information on source IP 722, one or more user flow blocks, which represent a logical group of flows associated with a user, a subscriber, or an entity representing a potential subscriber. Examples of these user flow blocks are default user flows block 724 and mapped user flows block 726. The default user flows block 724 store flows that are not yet associated with any particular user or sub-scriber. If the subscriber id or any other identifiers associated with a particular user is known a-priori, all the flows associ-ated with the particular user or subscriber will be assigned to the mapped user flows block 726. The mapped user flows block 726 also include flows that either have been, or are in the process of being mapped to a user or subscriber by the flow analyzer 312. The mapped user flows block 726 can be indexed using subscriber id.")<br><br>Swenson at [0075] ("Ideally, a flow can be assigned to the mapped user flows block 726 for a user or subscriber by the user's source IP address. However, in some cases, flows associated with an IP address may often be associated with a group of users or subscribers, but there is not enough information to identify a particular user or subscriber. In these cases, a pseudo sub-scriber id can be assigned in the default user flows block 724 until real users or subscribers are identified as more flows are observed.")<br><br>Swenson at [0076] ("An example user flow block 730 that can be included in the default user flows block 724 and the mapped user flows block 726 contains data fields like the subscriber id 732 (pseudo or real) estimated bandwidth 734, a list of all flows 736 associated with the subscriber id 732, and a list of cookie hashes 738 among other related flow information. Each entry in the list of cookie hashes 738 contains one unique cookie seen within the flows. The list of flows 736 includes one or more flow statistics block 740. Each flow statistics block 740 contains the IP flow identifier 742 (e.g., srcIP, dstIP, srcPort, dstPort ), current domain and cookie 7 44, total number of bytes seen in each direction 746, the total number of bytes in each direction as of the last update 748. Not shown in the FIG. 7 includes a list of cookie hashes associated with the flow and an expiration time") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 333 of 1100

| No. | '111 Patent Claim 28 | Swenson |
|---|---|---|
| | | Swenson at [0083] ("When a new flow is observed, flow cache entries are searched by matching source IP address 722 if the subscriber id or other identifiers of the flow are not available. In case of multiple users sharing an IP address, the flow analyzer 312 needs to find patterns or other identifiers in the flows to map them to particular subscribers. Flows without identified sub-scribers are added to the flow cache block under the default user flows 726, which is a default holding place for the new flows. The flow analyzer 312 later will scan through the default user flows that contain cookies or other identifiers that may be used to determine a real user or subscriber associated with the flow. If a flow contains identifiers not associated with an existing real user, a new user or subscriber is created and the user flow block is moved to newly created (or mapped) user or subscriber.") |

| No. | '111 Patent Claim 29 | Swenson |
|---|---|---|
| 29 | The method according to claim 28, wherein the packet network is an Transmission Control Protocol (TCP) network, and the packet is an TCP packet. | Swenson discloses the method according to claim 28, wherein the packet network is an Transmission Control Protocol (TCP) network, and the packet is an TCP packet.<br><br>For example, Swenson discloses TCP traffic flows of TCP packets.<br><br>*See supra* at Claim 28.<br><br>Swenson at [0019] ("In one embodiment, an on-demand network moni-toring method is adopted to gather data about network flows as they traverse the network. For example, network flows can be monitored selectively or on-demand based on the types of the content carried in the flows. Furthermore, the network monitoring can also be performed selectively at inline level, as well as out-of-band to improve efficiency. Both TCP and UDP flows are monitored to gather information about the state of the network, such as the average network throughput for each flow and end-to-end latency between, for example, a client device and an origin server providing multimedia con-tent to the client device. For each TCP or UDP flow, the system tracks the number of bytes sent ( and in some embodi-ments acknowledged). In TCP, the current window size may also be tracked. Records on network flows are stored in a flow statistics database, which can be indexed by subscriber iden-tification (ID), cell tower (base station), and network segment etc. As many flow records accumulate, this database repre-sents both historical and current network condition and capacity for delivering data. Network throughput can be mea-sured by calculating an average number of bytes delivered over a period of time. Steps may be taken to filter out |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 334 of 1100

| No. | '111 Patent Claim 29 | Swenson |
|---|---|---|
| | | spurious data from small flows with size less than a certain threshold that, when measured, cause very noisy results in measuring bandwidth and/or latency. For example, any flow having delivery time of less than 500 ms can be filtered.")<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0028] ("In contrast to conventional inline TCP throughput monitoring devices that monitor every single data packets transmitted and received, the network controller 140 is an "out-of-band" computer server that interfaces with the steer-ing device 130 to selectively inspect user flows of interest. The network controller 140 may further identify user flows (e.g., among the flows of interest) for optimization. In one embodiment, the network controller 140 may be imple-mented at the steering device 130 to monitor traffic. In other embodiments, the network controller 140 is coupled to and communicates with the steering device 130 for traffic moni-toring and optimization. When queried by the steering device 130, the network controller 140 determines if a given network flow should be ignored, monitored further or optimized. Opti-mization of a flow is often decided at the beginning of the flow because it is rarely possible to switch to optimized content mid-stream once non-optimized content delivery has begun. However, the network controller 140 may determine that existing flows associated with a particular subscriber or other entity should be optimized. In turn, new flows ( e.g., resulting from seek requests in media, new media requests, resume after pause, etc.) determined to be associated with the entity may be optimized. The network controller 140 uses the net-work state as well as historical traffic data in its decision for monitoring and optimization. Knowledge on the current net-work state, such as congestion, deems critical when it comes to data optimization.") |


~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 335 of 1100

| No. | '111 Patent Claim 29 | Swenson |
|---|---|---|
| | | Swenson at [0044] ("Additionally, historical flow data over a longer term helps the flow analyzer 312 to determine repeating patterns and heat-maps of certain network sections and to predict when they are under congestion. In this case, the flow statis-tics stored in the flow cache 322 can be mapped against traffic categories for analysis, for example, long-term running aver-ages of video flow bandwidth help determine suitability for optimization. Furthermore, estimated bandwidth per user ( or per cell-ID, per tower, or per router) over time may be metrics calculated by the flow analyzer 312 in order to determine short term needs for optimization. For example, the flow analyzer 312 may determine to being optimizing flows asso-ciated with a particular cell-ID (or those flows for identified high-bandwidth users on the cell-ID) in response to a thresh-old number of high-bandwidth users connecting to a same cell tower corresponding to the cell-ID. The reason why flow analyzer 312 selectively monitors large flows lies in the real-ization that TCP statistics for small objects, which make up most web flows, can be misleading and cause huge errors in throughput estimations.")<br><br>Swenson at [0061] ("Once a flow is reported to the network controller 140, a flow cache entry is created for the flow in the flow cache 322. The flow cache entry keeps track of the flow and its associated bandwidth. For a flow that is marked in "continue" mode, each time the steering device 130 forwards a next portion of the flow payload to the network controller 140, the flow cache 3 22 updates the number of bytes for transmitted in the flow. By monitoring the number of bytes per flow over time, the flow analyzer 312 is capable of determining an estimate value of bandwidth associated with flow. Further-more, since the steering device 130 does not have infinite packet buffers, if congestion happens on the network link 416 from the steering device 130 to the user device 110, the TCP congestion control mechanism kicks in at the steering device 130, which may slows down and/or eventually stop receiving data over the network link 413 from origin server 160. During the congestion, the steering device 130 would not forward any data to the network controller 140, since the link 416 is congested and the network controller 140 would not be able to transmit data to the user device 110. Therefore, as an inline element, the network controller 140 can detect network con-gestions and estimate bandwidth associated with any flows of interest selected by the network controller 140. However, in the "continue" mode, the network controller 140 does not modify and transform the HTTP messaged it receives over the ICAP interface. The network controller 140 simply updates the flow statistics and |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 336 of 1100

| No. | '111 Patent Claim 29 | Swenson |
|---|---|---|
| | | returns the video or images to the steering device 130 for transmission to the user device 110.")<br><br>Swenson at [0062] (" Based on the flow statistics stored in the flow cache 322, the network controller 140 can also aggregate the flows associated with a user or subscriber in order to estimate the total available bandwidth occupied by the user or subscriber. In one embodiment, the network controller 140 tracks all the flow cache entries looking for flows originated from a com-mon source IP address or a user device identifier. The flow analyzer 312 of the network controller 140 then attempts to group these flows together to form a flow history for the user or subscriber. The network controller further identifies users or subscribers using two data components in the flow cache entry: the TCP source port and HTTP cookies associated with the flow. Together with the flow history, the network control-ler 140 establish pattern, and identify users or subscribers and stores subscriber information in the subscriber log 324. More details of the flow cache and user mapping are described below with reference to FIG. 7.")<br><br>Swenson at [0073] ("FIG. 7 is a block diagram illustrating one embodi-ment of an example of internal components of the flow cache. The flow cache map 700 comprises a plurality of flow cache entries, such as flow cache entries 710 and 712 indexed by a hash. Not shown in the example diagram is a possible linked list behind each flow cache entry which allows chaining of flow cache entries for a given hash index. The hash into the flow cache may be based on source IP address, MAC address, subscriber ID, or other identifier indicative of a given sub-scriber, group of subscribers or subscriber's device.")<br><br>Swenson at [0084] ("The flow analyzer 312 can also map flows to users (subscribers to the mobile or network service) in the flow cache entries by matching cookie hashes, MAC address ( or any unique device identifiers), or TCP source ports. For example, if two flows share the same source port, it is very likely that they belong to the same user because TCP ports are reused often by an individual user, but not often between users. Furthermore, source ports can also be used to map users when network address translation (NAT) is deployed. In a typical network with NAT configuration, each user is allo-cated a block (e.g., 32) of TCP source ports. A random port number within the block is then picked for each new user flows initiated. With this knowledge, all source ports within a block can be |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 337 of 1100

| No. | '111 Patent Claim 29 | Swenson |
|---|---|---|
| | | aggregated under the same user. In some cases, a user with more than one block of port number assigned, the cookie hashes can be used to link the blocks together.") |

| No. | '111 Patent Claim 30 | Swenson |
|---|---|---|
| 30[a] | The method according to claim 1, further comprising: receiving, by the network node from the first entity over the packet network, one or more additional packets; | Swenson discloses the method according to claim 1, further comprising: receiving, by the network node from the first entity over the packet network, one or more additional packets.<br><br>For example, Swenson discloses sending by a user device or origin server multiple packets in a flow as well as packet flow requests and responses.<br><br>*See supra at* Claim 1, 1[c].<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0030] ("The video optimizer 150 is a computer server that provides video and image optimization and delivers opti-mized video and image content to the user devices 110 via the network 120. The video and image optimization is an on-demand service provided through the transcoding of the video and image content. For example, when a user device attempts to retrieve video from the origin server 160, the network controller 140 may decide that the flow meets certain criteria for content optimization. The network controller 140 then redirected the user devices 110 to the video optimizer 150 to retrieve the optimized content. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 338 of 1100

| No. | '111 Patent Claim 30 | Swenson |
|---|---|---|
| | | The video optimizer 150 receives information in the redirect request from the user devices 110 or from the network controller 140 about the video or image content to be optimized and retrieve the video or image content from the corresponding origin server 160 for optimization and subsequent delivery to the user devices 110.")<br><br>Swenson at [0063] ("FIG. 4B illustrates one embodiment of a second example working mode of the network controller 140 for providing selective on-demand network monitoring. In FIG. 4B, the network request path consists of a network link 421 from the user device 110 to the steering device 130, and a network link 422 from the steering device 130 to the origin server 160. On the opposite direction, the network response path consists of a network link 423 from the origin server 160 to the steering device 130, and a network link 424 from the steering device 13 0 back to the user device 110. Note that the network link pair 421 and 424 share the same physical link, so are network link pair 425 and 426.") |
| 30[b] | checking, by the network node, if any one of the one or more additional packets satisfies the criterion; | Swenson discloses checking, by the network node, if any one of the one or more additional packets satisfies the criterion.<br><br>*See supra at* Claim 1[d], 30[a]. |
| 30[c] | responsive to an additional packet not satisfying the criterion, sending, by the network node over the packet network, the additional packet to the second entity; and | Swenson discloses responsive to an additional packet not satisfying the criterion, sending, by the network node over the packet network, the additional packet to the second entity.<br><br>*See supra at* Claim 1[e], 30[a]. |
| 30[d] | responsive to the additional packet satisfying the criterion, sending the additional packet, by the network node over the packet | Swenson discloses responsive to the additional packet satisfying the criterion, sending the additional packet, by the network node over the packet network, in response to the instruction.<br><br>*See supra at* Claim 1[f], 30[a]. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 339 of 1100

| No. | '111 Patent Claim 30 | Swenson |
|---|---|---|
| | network, in response to the instruction. | |

| No. | '111 Patent Claim 31 | Swenson |
|---|---|---|
| 31[a] | The method according to claim 1, wherein the packet network is a Software Defined Network (SDN), | Swenson discloses the method according to claim 1, wherein the packet network is a Software Defined Network (SDN).<br><br>For example, Swenson discloses logically coupling a steering device to a network controller via an ICAP interface. A person of ordinary skill in the art would understand that the communication between the steering device and network controller constitutes an SDN. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Swenson is found to not meet this limitation, wherein the packet network is a Software Defined Network (SDN) would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>*See supra* at Claim 1.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 340 of 1100

| No. | '111 Patent Claim 31 | Swenson |
|---|---|---|
| | | Swenson at [0056] ("FIGS. 4A and 4B each illustrates one embodiment of an example working mode of the network controller for providing selective on-demand real-time network monitoring and subscriber identification. Shown with the network con-troller 140 are the user device 110, the steering device 130, and the origin server 160. The network controller 140 is coupled to the steering device 130 through the steering device interface 316. In one embodiment, the network controller 140 and the steering device 130 communicate with each other using the Internet content adaption protocol (ICAP). The steering device interface 316 executes an ICAP server 406, which interacts with an ICAP client 404 running on the steer-ing device 130. Similar or different protocols may be used for communication between the network controller 140 and the steering device 130 in other embodiments.")<br><br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Swenson in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 31[a] of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references.<br><br>For example, Kempf discloses a packet network using the OpenFlow protocol, i.e., wherein the packet network is a Software Defined Network (SDN). A person of ordinary skill in the art would understand that the OpenFlow protocol is used in Software Defined Networks.<br><br>Kempf at [0004] ("The GPRS tunneling protocol (GTP) is an important communication protocol utilized within the GPRS core net-work. GTP enables end user devices ( e.g., cellular phones) in a GSM network to move from place to place while continuing to connect to the Internet. The end user devices are connected to other devices through a gateway GPRS support node (GGSN). The GGSN tracks the end user device's data from the end user device's serving GPRS support node (GGSN) that is handling the session originating from the end user device.")<br><br>Kempf at [0006] ("A method implements a control plane of an evolved packet core (EPC) of a third generation partnership project (3GPP) long term evolution (LTE) network in a cloud com-puting system. The cloud computing system includes a cloud manager and a controller. The controller executes a plurality of control plane modules. The control plane |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 341 of 1100

| No. | '111 Patent Claim 31 | Swenson |
|---|---|---|
| | | communicates with the data plane of the EPC implemented in a plurality of network elements of the 3GPP LTE network through a control protocol. The EPC with the control plane implemented in the cloud computing system utilizes resources more efficiently than an architecture with the control plane implemented in the plurality of network elements of the 3GPP LTE network. The method comprises the steps of initializing the plurality of control plane modules of the EPC within the controller. Each control plane module in the plurality of control plane modules is initialized as a separate virtual machine by the cloud man-ager. Each control plane module provides a set of control plane functions for managing the data plane. The cloud man-ager monitors resource utilization of each control plane mod-ule and the control plane traffic handled by each control plane module. The cloud manager detects a threshold level of resource utilization or traffic load for one of the plurality of control plane modules of the EPC. A new control plane mod-ule is initialized as a separate virtual machine by the cloud manager in response to detecting the threshold level. The new control plane module shares the load of the one of the plural-ity of control plane modules and signals the plurality of net-work elements in the data plane to establish flow rules and actions to establish differential routing of flows in the data plane using the control protocol, wherein the control protocol is an OpenFlow protocol, and wherein flow matches are encoded using an extensible match structure in which the flow match is encoded as a type-length-value (TLV).")<br><br>Kempf at [0007] ("A cloud computer system implements a control plane of an evolved packet core (EPC) of a third generation partnership project (3GPP) long term evolution (LTE) net-work. The control plane communicates with the data plane of the EPC that is implemented in a plurality of network ele-ments of the 3GPP LTE network through a control protocol. The EPC with the control plane implemented in the cloud computing system utilizes resources more efficiently than an architecture with the control plane implemented in the plu-rality of network elements of the 3GPP LTE network. The cloud computing system, comprises a controller configured to execute a plurality of control plane modules of the EPC, each control plane module configured to provide a set of control plane functions for managing the data plane and to signal the plurality of network elements in the data plane to establish flow rules and actions to establish differential rout-ing of flows in the data plane using the control protocol, wherein the control protocol is an OpenFlow protocol, and wherein flow matches are encoded using an extensible match structure in which the flow match is encoded as a type-length-value (TLV) and a cloud manager communicatively |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 342 of 1100

| No. | '111 Patent Claim 31 | Swenson |
|---|---|---|
| | | coupled to the controller. The cloud manager is configured to initialize each of the plurality of control plane modules within the controller as a separate virtual machine, monitor resource utilization of each control plane module and the control plane traffic handled by each control plane module, detect whether a threshold level ofresource utilization or traffic load has been reached by any of the plurality of control plane modules of the EPC, and initialize a new control plane module as a separate virtual machine in response to detecting the threshold level, the new control plane module to share the load of the one of the plurality of control plane modules that exceeded the threshold level.")<br><br>Kempf at [0038] ("Implementing the control plane of an EPC in a cloud computing facility and the data plane of the EPC using a set of OpenFlow switches, as well as managing communication between the control plane and the dataplane using the Open-Flow protocol (e.g., OpenFlow 1.1), creates a problem that the OpenFlow protocol does not support GTP or GTP tunnel endpoint identifier (TEID) routing, which is necessary for implementing the dataplane of the EPC")<br><br>Kempf at [0039] ("The embodiments of the invention overcome these disadvantages of the prior art. The disadvantages of the prior art are avoided by splitting the control plane and the data plane for the EPC architecture and to implement the control plane by deploying the EPC control plane entities in a cloud computing facility, while the data plane is implemented by a distributed collection of OpenFlow switches. The OpenFlow protocol is used to connect the two, with enhancements to support GTP routing. While the EPC architecture already has a split between the control plane and the data plane, in the sense that the serving gateway (S-GW) and the PDN gateway (P-GW) are data plane entities while the MME, PCRF, and home subscriber server (HSS) are control plane entities, this split was made at the level of the mobility management pro-tocol, GTP.")<br><br>Kempf at [0040] ("The standard EPC architecture assumes a standard routed IP network for transport on top of which the mobile network entities and protocols are implemented. The enhanced EPC architecture described herein is instead at the level ofIP routing and media access control (MAC) switch-ing. Instead of using L2 routing and L3 internal gateway protocols to distribute IP routing and managing Ethernet and IP routing as a collection of distributed control entities, L2 and L3 routing management is centralized in a cloud facility and the routing is controlled from the cloud facility using the OpenFlow protocol. As used |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 343 of 1100

| No. | '111 Patent Claim 31 | Swenson |
|---|---|---|
| | | herein, the "OpenFlow proto-col" refers to the OpenFlow network protocol and switching specification defined in the OpenFlow Switch Specification at www.openflowswitch.org a web site hosted by Stanford Uni-versity. As used herein, an "OpenFlow switch" refers to a network element implementing the OpenFlow protocol.)<br><br>Kempf at [0044] ("FIG. 1 is a diagram of one embodiment of an example network with an OpenFlow switch, conforming to the OpenFlow 1.0 specification. The OpenFlow 1.0 protocol enables a controller 101 to connect to an OpenFlow 1.0 enabled switch 109 using a secure channel 103 and control a single forwarding table 107 in the switch 109. The controller 101 is an external software component executed by a remote computing device that enables a user to configure the Open-Flow 1.0 switch 109. The secure channel 103 can be provided by any type of network including a local area network (LAN) or a wide area network (WAN), such as the Internet.")<br><br>As another example, Chua discloses techniques and methods related to software defined networks (SDNs).<br><br>Chua at 1:45-55 ("In general, this disclosure describes techniques related to controlling software defined networks (SDNs). A software defined network is generally a network of interconnected computing devices having forwarding planes or data planes that can be programmed remotely by one or more controller devices. In this manner, the control plane can be physically separate from the data plane ( or forwarding plane) for an SDN. These computing devices can have either physical instantiation or virtual (software-only) instantiation without the presence of a hardware appliance. This disclosure describes various techniques related to controlling SDNs.")<br><br>Chua at 1:56-63 ("In one example, a method includes determining, by a con-troller device for a software defined network, connections between network devices in the software defined network, determining, by the controller device, one or more paths for network traffic between the network devices based on the determination of the connections, and programming, by the controller device, the network devices to direct network traf-fic along the one or more paths.")<br><br>Chua at 2:14-20 ("In another example, a method includes programming, by a controller device for a software defined network (SDN), a first network device of the SDN to send |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 344 of 1100

| No. | '111 Patent Claim 31 | Swenson |
|---|---|---|
| | | packets of a packet flow to a service device, and programming, by the controller device, one or more network devices of the SDN to perform a programmed action on packets of the packet flow based on data received from the service device for the packet flow.")<br><br>Chua at 2:38-48 ("In another example, a method includes programming, by a controller device for a software defined network (SDN), a set of network devices of the SDN to form a path through the SDN and to send data representative of packets sent along the path to the controller device, sending, by the controller device, packets of a packet flow corresponding to the path to one of the set of network devices, determining, by the controller device, whether the set of network devices is properly forwarding the packets of the packet flow along the path based on data received from the set of network devices, and present-ing a report representative of the determination.")<br><br>Chua at 5:50-6:5 ("SDN 106 generally serves to interconnect various endpoint devices, such as client device 102 and server device 104. In addition, SDN 106 may provide services to network traffic flowing between client device 102 and server device 104. Alternatively, SDN 106 may provide services to client device 102, without further directing traffic to server device 106. For example, administrator 114 may use SDN controller 112 to program network devices of SDN 106 to direct network traffic for client device 102 to one or more of service devices 116. Service devices 116 may include, for example, intrusion detection service (IDS) devices, intrusion prevention system (IPS) devices, web proxies, web servers, web-application firewalls and the like. In other examples, service devices 116 may, additionally or alternatively, include devices for provid-ing services such as, for example, denial of service (DoS) protection, distributed denial of service (DDoS) protection, traffic filtering, wide area network (WAN) acceleration, or other such services. Service devices 116 may also addition-ally or alternatively include malware detection devices, net-work anti-virus devices, network packet capture and analysis devices, honeypot devices, reflector net devices, tar pit devices, domain name service (DNS) and global DNS server devices, mail proxies, and anti-spam devices.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 345 of 1100

| No. | '111 Patent Claim 31 | Swenson |
|---|---|---|
| 31[b] | the packet is routed as part of a data plane and | Swenson discloses the packet is routed as part of a data plane.<br><br>For example, Swenson discloses routing traffic flows between a user device and server device via a steering device in a network. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Swenson is found to not meet this limitation the packet is routed as part of a data plane would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Swenson in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 31[b] of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references.<br><br>For example, Kempf discloses routing packets on a data plane using a control protocol.<br><br>Kempf at [0006] ("A method implements a control plane of an evolved packet core (EPC) of a third generation partnership project (3GPP) long term evolution (LTE) network in a cloud com-puting system. The cloud computing system includes a cloud manager and a controller. The controller executes a plurality of control plane modules. The control plane communicates with the data plane of the EPC implemented in a plurality of network elements of the 3GPP LTE network through a control protocol. The EPC with the control plane implemented in the cloud computing system utilizes resources more efficiently than |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 346 of 1100

| No. | '111 Patent Claim 31 | Swenson |
|---|---|---|
| | | an architecture with the control plane implemented in the plurality of network elements of the 3GPP LTE network. The method comprises the steps of initializing the plurality of control plane modules of the EPC within the controller. Each control plane module in the plurality of control plane modules is initialized as a separate virtual machine by the cloud man-ager. Each control plane module provides a set of control plane functions for managing the data plane. The cloud man-ager monitors resource utilization of each control plane mod-ule and the control plane traffic handled by each control plane module. The cloud manager detects a threshold level of resource utilization or traffic load for one of the plurality of control plane modules of the EPC. A new control plane mod-ule is initialized as a separate virtual machine by the cloud manager in response to detecting the threshold level. The new control plane module shares the load of the one of the plural-ity of control plane modules and signals the plurality of net-work elements in the data plane to establish flow rules and actions to establish differential routing of flows in the data plane using the control protocol, wherein the control protocol is an OpenFlow protocol, and wherein flow matches are encoded using an extensible match structure in which the flow match is encoded as a type-length-value (TLV).")<br><br>Kempf at [0007] ("A cloud computer system implements a control plane of an evolved packet core (EPC) of a third generation partnership project (3GPP) long term evolution (LTE) net-work. The control plane communicates with the data plane of the EPC that is implemented in a plurality of network ele-ments of the 3GPP LTE network through a control protocol. The EPC with the control plane implemented in the cloud computing system utilizes resources more efficiently than an architecture with the control plane implemented in the plu-rality of network elements of the 3GPP LTE network. The cloud computing system, comprises a controller configured to execute a plurality of control plane modules of the EPC, each control plane module configured to provide a set of control plane functions for managing the data plane and to signal the plurality of network elements in the data plane to establish flow rules and actions to establish differential rout-ing of flows in the data plane using the control protocol, wherein the control protocol is an OpenFlow protocol, and wherein flow matches are encoded using an extensible match structure in which the flow match is encoded as a type-length-value (TLV) and a cloud manager communicatively coupled to the controller. The cloud manager is configured to initialize each of the plurality of control plane modules within the controller as a separate virtual machine, monitor resource utilization of each control plane module and the control plane traffic handled by |



Orckit-Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 347 of 1100

| No. | '111 Patent Claim 31 | Swenson |
|---|---|---|
| | | each control plane module, detect whether a threshold level ofresource utilization or traffic load has been reached by any of the plurality of control plane modules of the EPC, and initialize a new control plane module as a separate virtual machine in response to detecting the threshold level, the new control plane module to share the load of the one of the plurality of control plane modules that exceeded the threshold level.")<br><br>Kempf at [0038] ("Implementing the control plane of an EPC in a cloud computing facility and the data plane of the EPC using a set of OpenFlow switches, as well as managing communication between the control plane and the dataplane using the Open-Flow protocol (e.g., OpenFlow 1.1), creates a problem that the OpenFlow protocol does not support GTP or GTP tunnel endpoint identifier (TEID) routing, which is necessary for implementing the dataplane of the EPC")<br><br>Kempf at [0039] ("The embodiments of the invention overcome these disadvantages of the prior art. The disadvantages of the prior art are avoided by splitting the control plane and the data plane for the EPC architecture and to implement the control plane by deploying the EPC control plane entities in a cloud computing facility, while the data plane is implemented by a distributed collection of OpenFlow switches. The OpenFlow protocol is used to connect the two, with enhancements to support GTP routing. While the EPC architecture already has a split between the control plane and the data plane, in the sense that the serving gateway (S-GW) and the PDN gateway (P-GW) are data plane entities while the MME, PCRF, and home subscriber server (HSS) are control plane entities, this split was made at the level of the mobility management pro-tocol, GTP.")<br><br>Kempf at [0040] ("The standard EPC architecture assumes a standard routed IP network for transport on top of which the mobile network entities and protocols are implemented. The enhanced EPC architecture described herein is instead at the level ofIP routing and media access control (MAC) switch-ing. Instead of using L2 routing and L3 internal gateway protocols to distribute IP routing and managing Ethernet and IP routing as a collection of distributed control entities, L2 and L3 routing management is centralized in a cloud facility and the routing is controlled from the cloud facility using the OpenFlow protocol. As used herein, the "OpenFlow proto-col" refers to the OpenFlow network protocol and switching specification defined in the OpenFlow Switch Specification at www.openflowswitch.org a |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 348 of 1100

| No. | '111 Patent Claim 31 | Swenson |
|---|---|---|
| | | web site hosted by Stanford Uni-versity. As used herein, an "OpenFlow switch" refers to a network element implementing the OpenFlow protocol.)<br><br>Kempf at [0041] ("The standard EPC control plane entities-the MME, PCRF, and HSS-are likewise deployed in the cloud, along with the control plane parts of the S-GW and P-GW, namely, the S-GW-C and the P-GW-C. The data plane con-sists of standard OpenFlow switches with enhancements as needed for routing GTP packets, rather than IP routers and Ethernet switches. At a minimum, the data plane parts of the S-GW and P-GW, namely, the S-GW-Dand the P-GW-D, and the packet routing part of the E-NodeB in the E-UTRAN require OpenFlow enhancements for GTP routing. Addi-tional enhancements for GTP routing may be needed on other switches within the EPC architecture depending on how much fine grained control over the routing an operator requires.")<br><br>Kempf at [0078] ("FIG. 15 is a diagram of one embodiment of how the EPC in the cloud computing system enables a managed ser-vices company to manage multiple operator networks out of a single data center. The managed services cloud computing facility 1501 runs separate instances of the EPC control plane for every mobile operator with which the managed services company has a contract. Each EPC instance is in a VPC 1503A,B that isolates the mobile operator's traffic from other tenants in the cloud computing facility 1501 of the data cen-ter. The EPC control plane instance for a mobile operator is connected to the mobile operator's geographically distributed EPC OpenFlow data plane switching fabric 1507 A,B and the mobile operator's base stations through a virtual edge router 1509A,B. The virtual edge router 1509A,B routes traffic from the data center to and from the appropriate mobile operator EPC data plane switching fabric 1507 A,B. In some cases, the mobile operators may even share base stations and EPC switching fabrics, though the example embodiment in FIG. 15 shows a case where the two mobile operators have separate switching fabrics.")<br><br>Kempf at [0087] ("In one embodiment, slow path support for GTP is implemented with an OpenFlow gateway switch. An Open-Flow mobile gateway switch also contains support on the software control plane for slow path packet processing. This path is taken by G-PDU (message type 255) packets with nonzero header fields or extension headers, and user data plane packets requiring encapsulation with such fields or addition of extension headers, and by G TP-U control packets. For this purpose, the switch supports three local ports in the |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 349 of 1100

| No. | '111 Patent Claim 31 | Swenson |
|---|---|---|
| | | software control plane: LOCAL_GTP _CONTROL-the switch fast path forwards GTP encapsulated packets directed to the gateway IP address that contain GTP-U control mes-sages and the local switch software control plane initiates local control plane actions depending on the GTP-U control message; LOCAL_GTP _U_DECAP-the switch fast path forwards G-PDU packets to this port that have nonzero header fields or extension headers (i.e. E!=0, S!=0, or PN!=0). These packets require specialized handling. The local switch software slow path processes the packets and performs the specialized handling; and LOCAL_GTP _U_ENCAP-the switch fast path forwards user data plane packets to this port that require encapsulation in a GTP tunnel with nonzero header fields or extension headers (i.e. E!=0, S!=0, or PN!=0). These packets require specialized handling. The local switch software slow path encapsulates the packets and performs the specialized handling. In addition to forwarding the packet, the switch fast path makes the OpenFlow metadata field avail-able to the slow path software.")<br><br>Kempf at [0093] ("The virtual port simply removes the GTP tunnel header and forwards the enclosed user data plane packet out the bound physical port.")<br><br>Kempf at [0101] ("In one embodiment, the system implements han-dling of user data plane packets requiring GTP-U encapsula-tion with extension headers, sequence numbers, and N-PDU numbers. User data plane packets that require extension head-ers, sequence numbers, or N-PDU numbers during GTP encapsulation require special handling by the software slow path. For these packets, the OpenFlow controller programs a rule matching the 4 tuple: IP source address; IP destination address; UDP/TCP/SCTP source port; and UDP/TCP/SCTP destination port. The instructions for matching packets are:<br><br>Write-Metadata ( GTP-TEID, 0x FFFFFFFF)<br>Apply-Actions (Set-Output-Port LOCAL_GTP _U_ENCAP )")<br><br>Kempf at [0145] ("In other embodiments, other control protocols can be utilized in place of OpenFlow as described herein. The use of OpenFlow is presented by way of example and not limita-tion. Other control protocols can also be utilized to manage the communication between the control plane and data plane and configuration of the data plane of the split EPC architec-ture. An example of such a protocol is FORCES, an IETF standard protocol for splitting the control plane and forward-ing plane in networks. The FORCES protocol |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 350 of 1100

| No. | '111 Patent Claim 31 | Swenson |
|---|---|---|
| | | specification is described in RFC 5810. RFC 5812 describes the architecture of a FORCES forwarding element, the equivalent of an Open-Flow switch. The FORCES protocol itself does not directly support programming routes into the forwarding element, it is, instead, a framework for handling the interaction between the FORCES controller and a FORCES forwarding element. The forwarding element architecture describes how to design the protocol that actually allows a FORCES controller to program a FORCES forwarding element. One skilled in the art would understand that a FORCES based system could include features described herein above in relation to the OpenFlow embodiment, such as the GTP OpenFlow exten-sion, to allow the controller to program the switches for GTP TEID routing.")<br><br>As another example, Chua discloses forwarding packets over a data plane to various network destinations.<br><br>Chua at 1:45-55 ("In general, this disclosure describes techniques related to controlling software defined networks (SDNs). A software defined network is generally a network of interconnected computing devices having forwarding planes or data planes that can be programmed remotely by one or more controller devices. In this manner, the control plane can be physically separate from the data plane ( or forwarding plane) for an SDN. These computing devices can have either physical instantiation or virtual (software-only) instantiation without the presence of a hardware appliance. This disclosure describes various techniques related to controlling SDNs.")<br><br>Chua at 1:56-63 ("In one example, a method includes determining, by a con-troller device for a software defined network, connections between network devices in the software defined network, determining, by the controller device, one or more paths for network traffic between the network devices based on the determination of the connections, and programming, by the controller device, the network devices to direct network traf-fic along the one or more paths.")<br><br>Chua at 23:22-34 ("FIG. 4 illustrates various devices and services organized according to the "control plane" and the "data plane." In general, devices and services of the control plane manage devices of the data plane to cause the devices of the data plane to forward data traffic between various network destinations. In conventional routers, each router includes functionality for both the control plane and the data plane, and the same is true for |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 351 of 1100

| No. | '111 Patent Claim 31 | Swenson |
|---|---|---|
| | | conventional switches. However, in accordance with the techniques of this disclosure, the control plane can be entirely separated from the data plane, such that an SDN controller, such as SDN controller 112, can program devices of the data plane, such as network switches, to perform the techniques of this disclosure.")<br><br>Chua at 23:35:45 ("FIG. 4 is a conceptual diagram illustrating an example flow management system 250 including various components that may operate in accordance with the techniques of this disclo-sure. Flow management system 250 (also referred to as "sys-tem 250") includes control plane 252 and data plane 280. In general, control plane 252 includes components that relate to control information, e.g., routing information relating to packet flows and paths through an SDN. Data plane 280 generally includes components that send, forward, and/or receive data in accordance with control information from components of control plane 252.")<br><br>Chua at 24:20-36 ("In accordance with the techniques of this disclosure, flow management server 256 programs network switches 282, based on connections between network switches 282, to form paths through an SDN. For example, flow management server 256 may program network switches 282 to establish a path between TCP client 284 and server 288, and/or a path between TCP client 284 and multicast source 286. In some examples, flow management server 256 may program net-work switches 282 to define multiple paths, e.g., a primary path and one or more backup paths, as discussed above. Likewise, flow management server 256 may send test traffic through network switches 282 to test one or more of the paths. Data plane 280 may include one or more service devices (such as web proxy devices, IDS devices, and/or web serv-ers), to which network switches 282 may direct network packets. Server 288 may represent a service device of an SDN controlled by control plane 252, in some examples.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 352 of 1100

| No. | '111 Patent Claim 31 | Swenson |
|---|---|---|
| 31[c] | the network node communication with the controller serves as a control plane. | Swenson discloses the network node communication with the controller serves as a control plane.<br><br>For example, Swenson discloses communication between the steering device and controller. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Swenson is found to not meet this limitation, the network node communication with the controller serves as a control plane would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Swenson in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 31[c] of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references.<br><br>For example, Kempf discloses communication between network elements and an OpenFlow controller over a control plane.<br><br>Kempf at [0006] ("A method implements a control plane of an evolved packet core (EPC) of a third generation partnership project (3GPP) long term evolution (LTE) network in a cloud com-puting system. The cloud computing system includes a cloud manager and a controller. The controller executes a plurality of control plane modules. The control plane communicates with the data plane of the EPC implemented in a plurality of network |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 353 of 1100

| No. | '111 Patent Claim 31 | Swenson |
|---|---|---|
| | | elements of the 3GPP LTE network through a control protocol. The EPC with the control plane implemented in the cloud computing system utilizes resources more efficiently than an architecture with the control plane implemented in the plurality of network elements of the 3GPP LTE network. The method comprises the steps of initializing the plurality of control plane modules of the EPC within the controller. Each control plane module in the plurality of control plane modules is initialized as a separate virtual machine by the cloud man-ager. Each control plane module provides a set of control plane functions for managing the data plane. The cloud man-ager monitors resource utilization of each control plane mod-ule and the control plane traffic handled by each control plane module. The cloud manager detects a threshold level of resource utilization or traffic load for one of the plurality of control plane modules of the EPC. A new control plane mod-ule is initialized as a separate virtual machine by the cloud manager in response to detecting the threshold level. The new control plane module shares the load of the one of the plural-ity of control plane modules and signals the plurality of net-work elements in the data plane to establish flow rules and actions to establish differential routing of flows in the data plane using the control protocol, wherein the control protocol is an OpenFlow protocol, and wherein flow matches are encoded using an extensible match structure in which the flow match is encoded as a type-length-value (TLV).")<br><br>Kempf at [0007] ("A cloud computer system implements a control plane of an evolved packet core (EPC) of a third generation partnership project (3GPP) long term evolution (LTE) net-work. The control plane communicates with the data plane of the EPC that is implemented in a plurality of network ele-ments of the 3GPP LTE network through a control protocol. The EPC with the control plane implemented in the cloud computing system utilizes resources more efficiently than an architecture with the control plane implemented in the plu-rality of network elements of the 3GPP LTE network. The cloud computing system, comprises a controller configured to execute a plurality of control plane modules of the EPC, each control plane module configured to provide a set of control plane functions for managing the data plane and to signal the plurality of network elements in the data plane to establish flow rules and actions to establish differential rout-ing of flows in the data plane using the control protocol, wherein the control protocol is an OpenFlow protocol, and wherein flow matches are encoded using an extensible match structure in which the flow match is encoded as a type-length-value (TLV) and a cloud manager communicatively coupled to the controller. The cloud manager is configured to initialize each of the plurality |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 354 of 1100

| No. | '111 Patent Claim 31 | Swenson |
|---|---|---|
| | | of control plane modules within the controller as a separate virtual machine, monitor resource utilization of each control plane module and the control plane traffic handled by each control plane module, detect whether a threshold level ofresource utilization or traffic load has been reached by any of the plurality of control plane modules of the EPC, and initialize a new control plane module as a separate virtual machine in response to detecting the threshold level, the new control plane module to share the load of the one of the plurality of control plane modules that exceeded the threshold level.")<br><br>Kempf at [0038] ("Implementing the control plane of an EPC in a cloud computing facility and the data plane of the EPC using a set of OpenFlow switches, as well as managing communication between the control plane and the dataplane using the Open-Flow protocol (e.g., OpenFlow 1.1), creates a problem that the OpenFlow protocol does not support GTP or GTP tunnel endpoint identifier (TEID) routing, which is necessary for implementing the dataplane of the EPC")<br><br>Kempf at [0039] ("The embodiments of the invention overcome these disadvantages of the prior art. The disadvantages of the prior art are avoided by splitting the control plane and the data plane for the EPC architecture and to implement the control plane by deploying the EPC control plane entities in a cloud computing facility, while the data plane is implemented by a distributed collection of OpenFlow switches. The OpenFlow protocol is used to connect the two, with enhancements to support GTP routing. While the EPC architecture already has a split between the control plane and the data plane, in the sense that the serving gateway (S-GW) and the PDN gateway (P-GW) are data plane entities while the MME, PCRF, and home subscriber server (HSS) are control plane entities, this split was made at the level of the mobility management pro-tocol, GTP.")<br><br>Kempf at [0040] ("The standard EPC architecture assumes a standard routed IP network for transport on top of which the mobile network entities and protocols are implemented. The enhanced EPC architecture described herein is instead at the level ofIP routing and media access control (MAC) switch-ing. Instead of using L2 routing and L3 internal gateway protocols to distribute IP routing and managing Ethernet and IP routing as a collection of distributed control entities, L2 and L3 routing management is centralized in a cloud facility and the routing is controlled from the cloud facility using the OpenFlow protocol. As used herein, the "OpenFlow proto-col" refers to the OpenFlow network protocol and switching |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 355 of 1100

| No. | '111 Patent Claim 31 | Swenson |
|---|---|---|
| | | specification defined in the OpenFlow Switch Specification at www.openflowswitch.org a web site hosted by Stanford Uni-versity. As used herein, an "OpenFlow switch" refers to a network element implementing the OpenFlow protocol.)<br><br>Kempf at [0041] ("The standard EPC control plane entities-the MME, PCRF, and HSS-are likewise deployed in the cloud, along with the control plane parts of the S-GW and P-GW, namely, the S-GW-C and the P-GW-C. The data plane con-sists of standard OpenFlow switches with enhancements as needed for routing GTP packets, rather than IP routers and Ethernet switches. At a minimum, the data plane parts of the S-GW and P-GW, namely, the S-GW-Dand the P-GW-D, and the packet routing part of the E-NodeB in the E-UTRAN require OpenFlow enhancements for GTP routing. Addi-tional enhancements for GTP routing may be needed on other switches within the EPC architecture depending on how much fine grained control over the routing an operator requires.")<br><br>Kempf at [0078] ("FIG. 15 is a diagram of one embodiment of how the EPC in the cloud computing system enables a managed ser-vices company to manage multiple operator networks out of a single data center. The managed services cloud computing facility 1501 runs separate instances of the EPC control plane for every mobile operator with which the managed services company has a contract. Each EPC instance is in a VPC 1503A,B that isolates the mobile operator's traffic from other tenants in the cloud computing facility 1501 of the data cen-ter. The EPC control plane instance for a mobile operator is connected to the mobile operator's geographically distributed EPC OpenFlow data plane switching fabric 1507 A,B and the mobile operator's base stations through a virtual edge router 1509A,B. The virtual edge router 1509A,B routes traffic from the data center to and from the appropriate mobile operator EPC data plane switching fabric 1507 A,B. In some cases, the mobile operators may even share base stations and EPC switching fabrics, though the example embodiment in FIG. 15 shows a case where the two mobile operators have separate switching fabrics.")<br><br>Kempf at [0087] ("In one embodiment, slow path support for GTP is implemented with an OpenFlow gateway switch. An Open-Flow mobile gateway switch also contains support on the software control plane for slow path packet processing. This path is taken by G-PDU (message type 255) packets with nonzero header fields or extension headers, and user data plane packets requiring encapsulation with such fields or addition of extension headers, and |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 356 of 1100

| No. | '111 Patent Claim 31 | Swenson |
|---|---|---|
| | | by G TP-U control packets. For this purpose, the switch supports three local ports in the software control plane: LOCAL_GTP _CONTROL-the switch fast path forwards GTP encapsulated packets directed to the gateway IP address that contain GTP-U control mes-sages and the local switch software control plane initiates local control plane actions depending on the GTP-U control message; LOCAL_GTP _U_DECAP-the switch fast path forwards G-PDU packets to this port that have nonzero header fields or extension headers (i.e. E!=0, S!=0, or PN!=0). These packets require specialized handling. The local switch software slow path processes the packets and performs the specialized handling; and LOCAL_GTP _U_ENCAP-the switch fast path forwards user data plane packets to this port that require encapsulation in a GTP tunnel with nonzero header fields or extension headers (i.e. E!=0, S!=0, or PN!=0). These packets require specialized handling. The local switch software slow path encapsulates the packets and performs the specialized handling. In addition to forwarding the packet, the switch fast path makes the OpenFlow metadata field avail-able to the slow path software.")<br><br>Kempf at [0093] ("The virtual port simply removes the GTP tunnel header and forwards the enclosed user data plane packet out the bound physical port.")<br><br>Kempf at [0101] ("In one embodiment, the system implements han-dling of user data plane packets requiring GTP-U encapsula-tion with extension headers, sequence numbers, and N-PDU numbers. User data plane packets that require extension head-ers, sequence numbers, or N-PDU numbers during GTP encapsulation require special handling by the software slow path. For these packets, the OpenFlow controller programs a rule matching the 4 tuple: IP source address; IP destination address; UDP/TCP/SCTP source port; and UDP/TCP/SCTP destination port. The instructions for matching packets are:<br><br>Write-Metadata ( GTP-TEID, 0x FFFFFFFF)<br>Apply-Actions (Set-Output-Port LOCAL_GTP _U_ENCAP )")<br><br>Kempf at [0145] ("In other embodiments, other control protocols can be utilized in place of OpenFlow as described herein. The use of OpenFlow is presented by way of example and not limita-tion. Other control protocols can also be utilized to manage the communication between the control plane and data plane and configuration of the data plane of the split EPC architec-ture. An example of such a protocol is FORCES, an IETF standard protocol |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 357 of 1100

| No. | '111 Patent Claim 31 | Swenson |
|---|---|---|
| | | for splitting the control plane and forward-ing plane in networks. The FORCES protocol specification is described in RFC 5810. RFC 5812 describes the architecture of a FORCES forwarding element, the equivalent of an Open-Flow switch. The FORCES protocol itself does not directly support programming routes into the forwarding element, it is, instead, a framework for handling the interaction between the FORCES controller and a FORCES forwarding element. The forwarding element architecture describes how to design the protocol that actually allows a FORCES controller to program a FORCES forwarding element. One skilled in the art would understand that a FORCES based system could include features described herein above in relation to the OpenFlow embodiment, such as the GTP OpenFlow exten-sion, to allow the controller to program the switches for GTP TEID routing.")<br><br>As another example, Chua discloses the network device's communication with the SDN controller over the control plane as controlling and programming the network devices to direct network traffic along one or more paths.<br><br>Chua at 1:64-2:5 ("In another example, a controller device for a software defined network includes one or more interfaces for commu-nicating with network devices in the software defined net-work, and one or more processors configured to determine connections between the network devices, determine one or more paths for network traffic between the network devices based on the determination of the connections, and program the network devices to direct network traffic along the one or more paths.")<br><br>Chua at 2:21-29 ("In another example, a controller device for a software defined network (SDN) includes one or more network inter-faces configured to communicate with network devices of the SDN, and one or more processors configured to program a first network device of the SDN to send packets of a packet flow to a service device, and program one or more network devices of the SDN to perform a programmed action on packets of the packet flow based on data received from the service device for the packet flow.")<br><br>Chua at 2:49-61 ("In another example, a controller device for a software defined network (SDN) includes one or more network interfaces configured to communicate with network devices of the SDN, and one or more processors configured to program a set of network devices of the SDN to form a path through the SDN and to send data representative of |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 358 of 1100

| No. | '111 Patent Claim 31 | Swenson |
|---|---|---|
| | | packets sent along the path to the controller device, send, via one of the network interfaces, packets of a packet flow corresponding to the path to one of the set of network devices, determine whether the set of network devices is properly forwarding the packets of the packet flow along the path based on data received from the set of network devices, and present a report representative of the determination.")<br><br>Chua at 23:62-24:4 ("OpenFlow is an example of an SDN protocol. That is, in some examples, SDN controller 270 may conform to the OpenFlow protocol. However, it should be understood that other protocols may be used in conjunction with a software defined network. In general, any protocol that gives access to the forwarding plane or data plane of a networking (e.g., a switch or router) to a remote device over a network may be used in accordance with the techniques of this disclo-sure, other example protocols include XMPP, RESTful APis, Cisco OnePK, IETF I2RS (Interface to Routing Systems).") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 359 of 1100

**EXHIBIT D-3**
Defendant's Preliminary Invalidity Contentions
*Orckit Corporation v. Cisco Systems, Inc.*, 2:22-cv-00276-JRG-RSP

---

**Chart for U.S. Patent 10,652,111 ("the '111 Patent")**
**U.S. Patent Publication No. 2014/0140211 to Chandrasekaran et al. ("Chandrasekaran")**

As shown in the chart below, all Asserted Claims of the '111 Patent are invalid under (1) AIA-35 U.S.C. § 102 (a) because Chandrasekaran meets each element of those claims, and/or (2) 35 U.S.C. § 103 because Chandrasekaran renders those claims obvious either alone, or in combination with the knowledge of a person having ordinary skill in the art, and in further combination with the references specifically identified below and in the following claim chart and/or one or more references identified in Defendant's Preliminary Invalidity Contentions. The following quotations and diagrams come from Chandrasekaran titled "Classification of Traffic For Application Aware Policies In A Wireless Network", which was filed on November 16, 2012, and published on May 22, 2014.

Motivations to combine the disclosures in Chandrasekaran with disclosures in other publications known in the art, as explained in this chart, include at least the similarity in subject matter between the references to the extent they concern methods relating to routing certain network traffic to entities for further analysis and inspection. Insofar as the references cite other patents or publications, or suggest additional changes, one of ordinary skill in the art would look beyond a single reference to other references in the field.

These invalidity contentions are based on Defendant's present understanding of the Asserted Claims, and Orckit's apparent construction of the claims in its November 3, 2022 Disclosure of Asserted Claims and Infringement Contentions Pursuant to P.R. 3-1, and Orckit's January 19, 2023 First Amended Disclosure of Asserted Claims and Infringement Contentions Pursuant to P.R. 3-1 (Orckit's "Infringement Disclosures"), which is deficient at least insofar as it fails to cite any documents or identify accused structures, acts, or materials in the Accused Products with particularity. Defendant does not agree with Orckit's application of the claims, or that the claims satisfy the requirements of 35 U.S.C. § 112. Defendant's contentions herein are not, and should in no way be seen as, admissions or adoptions as to any particular claim scope or construction, or as any admission that any particular element is met by any accused product in any particular way. Defendant objects to any attempt to imply claim construction from this chart. Defendant's prior art invalidity contentions are made in a variety of alternatives and do not represent Defendant's agreement or view as to the meaning, definiteness, written description support for, or enablement of any claim contained therein.



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 360 of 1100

The following contentions are subject to revision and amendment pursuant to Federal Rule of Civil Procedure 26(e), the Local Rules, and the Orders of record in this matter subject to further investigation and discovery regarding the prior art and the Court's construction of the claims at issue.

| No. | '111 Patent Claim 1 | Chandrasekaran |
|---|---|---|
| 1[preamble] | A method for use with a packet network including a network node for transporting packets between first and second entities under control of a controller that is external to the network node, the method comprising: | Chandrasekaran discloses a method for use with a packet network including a network node for transporting packets between first and second entities under control of a controller that is external to the network node, the method comprising.<br><br>For example, Chandrasekaran discloses a method used in a network in which packets are sent between mobile devices in which the network includes a controller in communication with mobile devices, through an external access point. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met.<br><br>Chandrasekaran at Abstract ("In one embodiment, a method includes performing stateful application classification on packets received at a controller and transmitting classification information to an access point. The classification information includes flow information and stateless rules for applying policies. The access point is con-figured to use the classification information to perform state-less application classification and apply policies to packets received from a mobile device. An apparatus and logic are also disclosed herein.")<br><br>Chandrasekaran at [0007] ("In one embodiment, a method generally comprises performing stateful application classification on packets received at a controller and transmitting classification infor-mation to an access point. The classification information comprises flow information and stateless rules for applying policies. The access point is configured to use the classifica-tion information to perform stateless application classifica-tion and apply policies to packets received from a mobile device.")<br><br>Chandrasekaran at [0012] ("Referring now to the drawings, and first to FIG.1, an example of a network in which embodiments described herein may be implemented is shown. For simplification, only a small number of network devices are shown. The network includes a wireless controller 12 in communication with a mobile device (client, wireless device, endpoint) 16 through an access point (AP) 14. In the example shown in FIG. 1, the controller 12 is in wired communication with two access points 14 for wireless communication with any number of mobile devices 16 via a wireless network ( e.g., WLAN |

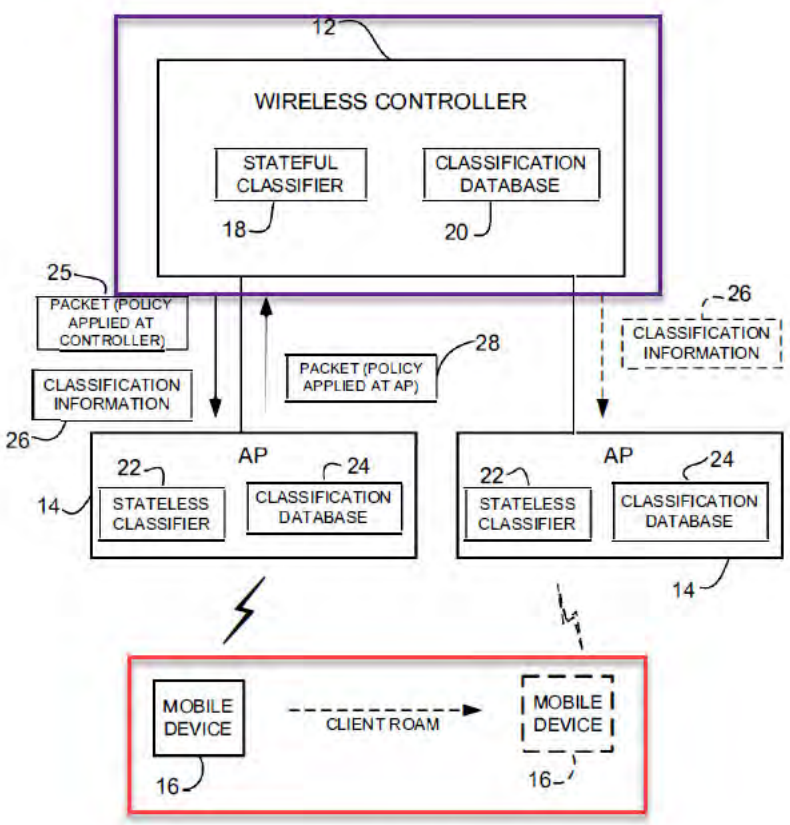


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 361 of 1100

| No. | '111 Patent Claim 1 | Chandrasekaran |
|---|---|---|
| | | (wire-less local area network)) at a network site. The wireless con-troller 12 may be in communication with one or more other networks (not shown) (e.g., Internet, intranet, local area net-work, wireless local area network, cellular network, metro-politan area network, wide area network, satellite network, radio access network, public switched network, virtual pri-vate network, or any other network or combination thereof). Communication paths between the wireless controller 12 and other networks or between the controller and access points 14 may include any number or type of intermediate nodes (e.g., routers, switches, gateways, or other network devices), which facilitate passage of data between network devices.")<br><br>Chandrasekaran at [0013] ("In one example, the wireless controller 12 receives upstream traffic transmitted from the mobile device 16 and destined for another endpoint ( e.g., host, user device), and transmits downstream traffic received from the endpoint to the mobile device in a communication session. As used herein, the term 'downstream' refers to traffic transmitted from the controller 12 towards the mobile device 16, and the term 'upstream' refers to traffic transmitted from the mobile device towards the controller.")<br><br>Chandrasekaran at [0014] ("The term 'wireless controller' or 'controller' as used herein may refer to a wireless LAN (local area network) controller, mobility controller, wireless control device, wire-less control system, or any other network device operable to perform control functions for a wireless network. The net-work site may also include a wireless control system or other platform for centralized wireless LAN planning, configura-tion, and management. The wireless controller 12 enables system wide functions for wireless applications and may support any number of access points 14. Each access point 14 may serve any number of mobile devices 16 in the wireless network. The wireless controller 12 may be, for example, a standalone device or a rack-mounted appliance. In the example shown in FIG. 1, the wireless controller 12 and access points 14 are separate devices and may be located remote from one another. The wireless controller 12 may also be integrated with the access point 14 ( e.g., autonomous AP) or located at a switch, router, switch/router, or other network device. Thus, the wireless controller 12 may be a physical device located at a standalone device, access point, switch, router, or other network device. The wireless controller 12 may also be a virtual device located in a network or cloud, for example.")<br><br>Chandrasekaran at Figure (annotations added) |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 362 of 1100

| No. | '111 Patent Claim 1 | Chandrasekaran |
|---|---|---|
| | |  <br>FIGURE 1 |
| 1[a] | sending, by the controller to the network node over the packet network, an instruction and a | Chandrasekaran discloses sending, by the controller to the network node over the packet network, an instruction and a packet-applicable criterion.<br><br>For example, Chandrasekaran discloses a controller sending classification information, including flow information and rules, to an access point. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 363 of 1100

| No. | '111 Patent Claim 1 | Chandrasekaran |
|---|---|---|
| | packet-applicable criterion; | Chandrasekaran at Abstract ("In one embodiment, a method includes performing stateful application classification on packets received at a controller and transmitting classification information to an access point. The classification information includes flow information and stateless rules for applying policies. The access point is con-figured to use the classification information to perform state-less application classification and apply policies to packets received from a mobile device. An apparatus and logic are also disclosed herein.")<br><br>Chandrasekaran at [0007] ("In one embodiment, a method generally comprises performing stateful application classification on packets received at a controller and transmitting classification infor-mation to an access point. The classification information comprises flow information and stateless rules for applying policies. The access point is configured to use the classifica-tion information to perform stateless application classifica-tion and apply policies to packets received from a mobile device.")<br><br>Chandrasekaran at [0008] ("In another embodiment, an apparatus generally comprises a stateful classifier for performing stateful appli-cation classification at a controller, a classification database for storing classification information, and a processor for transmitting the classification information to an access point. The classification information comprises flow information and stateless rules for applying policies. The access point is configured to use the classification information to perform stateless application classification and apply policies to pack-ets received from a mobile device.")<br><br>Chandrasekaran at [0016] ("The wireless controller 12 includes a stateful appli-cation classifier 18 and the AP 14 includes a stateless appli-cation classifier 22. After the stateful classifier 18 identifies the application, the controller 12 transmits ( e.g., pushes) clas-sification information 26 to the AP 14 so that the AP can perform stateless classification and apply policies ( e.g., QoS or other policies) to traffic received from the mobile device 16. The controller 12 may also provide the classification information 26 to another AP 14 if the client 16 roams to a new AP, as shown in FIG. 1. Implementation of the stateful classifier 18 at the controller 12 and stateless classifier 22 at the AP 14 allows for policies to be applied for downstream traffic (packet 25) at the wireless controller 12, and for upstream traffic (packet 28) at the access point 14.") |

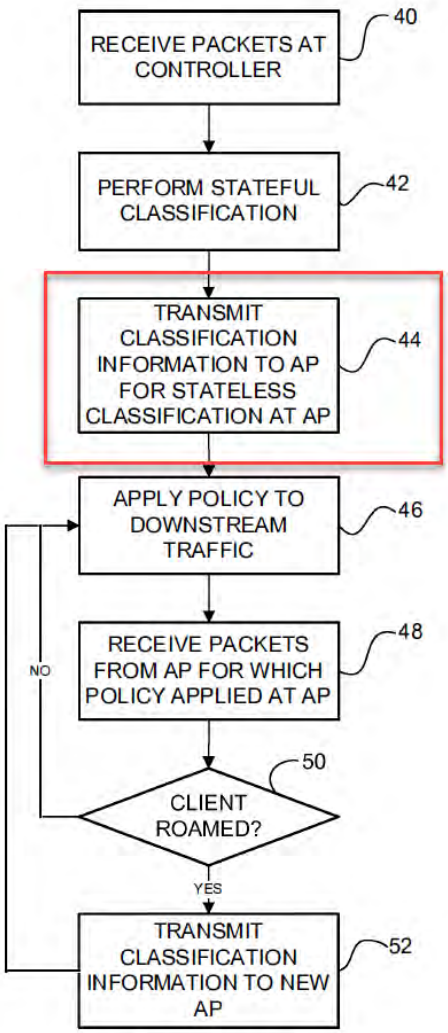


~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 364 of 1100

| No. | '111 Patent Claim 1 | Chandrasekaran |
|---|---|---|
| | | Chandrasekaran at [0022] ("The wireless controller 12 and AP 14 further include classification databases 20, 24, respectively, for storing clas-sification information. The classification database 20 at the controller 12 stores classification information obtained by the stateful classifier 18. The classification database 24 at the AP 14 stores classification information 26 transmitted to the AP from the controller 12. The classification information stored at the databases 20, 24 may include, for example, flow infor-mation, stateless rules, and policies, as described below.")<br><br>Chandrasekaran at [0023] ("In one embodiment, the classification information 26 transmitted from the controller 12 to the AP 14 includes tuple information for a flow ( e.g., source IP address, destina-tion IP address, source port, destination port, and protocol), application identifier (ID), and stateless DPI information. Stateless DPI information includes classification and sub-classification information ( e.g., fixed or variable offset with a pattern or regular expression) and rules for applying policies on the sub-classified packets. The policies may include, for example, drop packet, mark a DSCP (Differentiated Services Code Point) value in the packet, or rate limit the traffic.")<br><br>Chandrasekaran at [0031] ("FIG. 3 is a flowchart illustrating an example of a process at the controller 12 for classification of traffic for application aware policies in a wireless network, in accor-dance with one embodiment. At step 40, the controller 12 receives packets belonging to a network flow. The controller 12 performs stateful classification to identify an application associated with the flow ( step 42). The controller 12 transmits classification information ( e.g., flow information, stateless DPI rule, and policy) to the AP 14 for use in stateless classi-fication at the AP (step 44). The controller 12 applies policies to downstream traffic (received at the controller and destined for the client 16) (step 46) and receives upstream traffic for which policies have been applied at theAP 14 (step 48). If the controller 12 determines ( e.g., receives an indication) that the client 16 has roamed, it transmits the classification informa-tion to the new AP 14 to which the client has roamed (steps 50 and 52).")<br><br>Chandrasekaran at Figure 3 (annotations added) |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 365 of 1100

| No. | '111 Patent Claim 1 | Chandrasekaran |
|---|---|---|
| | |  <br>FIGURE 3 |
| 1[b] | receiving, by the network node from the controller, the | Chandrasekaran discloses receiving, by the network node from the controller, the instruction and the criterion. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 366 of 1100

| No. | '111 Patent Claim 1 | Chandrasekaran |
|---|---|---|
| | instruction and the criterion; | *See supra at* 1[a]. |
| 1[c] | receiving, by the network node from the first entity over the packet network, a packet addressed to the second entity; | Chandrasekaran discloses receiving, by the network node from the first entity over the packet network, a packet addressed to the second entity.<br><br>For example, Chandrasekaran discloses an access point that receives data packets and traffic over a packet network from a first mobile device that is destined for another endpoint.<br><br>Chandrasekaran at [0012] ("Referring now to the drawings, and first to FIG.1, an example of a network in which embodiments described herein may be implemented is shown. For simplification, only a small number of network devices are shown. The network includes a wireless controller 12 in communication with a mobile device (client, wireless device, endpoint) 16 through an access point (AP) 14. In the example shown in FIG. 1, the controller 12 is in wired communication with two access points 14 for wireless communication with any number of mobile devices 16 via a wireless network ( e.g., WLAN (wire-less local area network)) at a network site. The wireless con-troller 12 may be in communication with one or more other networks (not shown) (e.g., Internet, intranet, local area net-work, wireless local area network, cellular network, metro-politan area network, wide area network, satellite network, radio access network, public switched network, virtual pri-vate network, or any other network or combination thereof). Communication paths between the wireless controller 12 and other networks or between the controller and access points 14 may include any number or type of intermediate nodes (e.g., routers, switches, gateways, or other network devices), which facilitate passage of data between network devices.")<br><br>Chandrasekaran at [0013] ("In one example, the wireless controller 12 receives upstream traffic transmitted from the mobile device 16 and destined for another endpoint ( e.g., host, user device), and transmits downstream traffic received from the endpoint to the mobile device in a communication session. As used herein, the term 'downstream' refers to traffic transmitted from the controller 12 towards the mobile device 16, and the term 'upstream' refers to traffic transmitted from the mobile device towards the controller.")<br><br>Chandrasekaran at [0014] ("The term 'wireless controller' or 'controller' as used herein may refer to a wireless LAN (local area network) controller, mobility controller, wireless control |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 367 of 1100

| No. | '111 Patent Claim 1 | Chandrasekaran |
|---|---|---|
| | | device, wire-less control system, or any other network device operable to perform control functions for a wireless network. The net-work site may also include a wireless control system or other platform for centralized wireless LAN planning, configura-tion, and management. The wireless controller 12 enables system wide functions for wireless applications and may support any number of access points 14. Each access point 14 may serve any number of mobile devices 16 in the wireless network. The wireless controller 12 may be, for example, a standalone device or a rack-mounted appliance. In the example shown in FIG. 1, the wireless controller 12 and access points 14 are separate devices and may be located remote from one another. The wireless controller 12 may also be integrated with the access point 14 ( e.g., autonomous AP) or located at a switch, router, switch/router, or other network device. Thus, the wireless controller 12 may be a physical device located at a standalone device, access point, switch, router, or other network device. The wireless controller 12 may also be a virtual device located in a network or cloud, for example.")<br><br>Chandrasekaran at [0015] ("The mobile device 16 may be any suitable equip-ment that supports wireless communication, including for example, a mobile phone, personal digital assistant, portable computing device, laptop, tablet, multimedia device, or any other wireless device. The mobile device 16 and access point 14 are configured to perform wireless communication according to a wireless network communication protocol such as IEEE 802.11/Wi-Fi.")<br><br>Chandrasekaran at [0016] ("The wireless controller 12 includes a stateful appli-cation classifier 18 and the AP 14 includes a stateless appli-cation classifier 22. After the stateful classifier 18 identifies the application, the controller 12 transmits ( e.g., pushes) clas-sification information 26 to the AP 14 so that the AP can perform stateless classification and apply policies ( e.g., QoS or other policies) to traffic received from the mobile device 16. The controller 12 may also provide the classification information 26 to another AP 14 if the client 16 roams to a new AP, as shown in FIG. 1. Implementation of the stateful classifier 18 at the controller 12 and stateless classifier 22 at the AP 14 allows for policies to be applied for downstream traffic (packet 25) at the wireless controller 12, and for upstream traffic (packet 28) at the access point 14.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 368 of 1100

| No. | '111 Patent Claim 1 | Chandrasekaran |
|---|---|---|
| 1[d] | checking, by the network node, if the packet satisfies the criterion; | Chandrasekaran discloses checking, by the network node, if the packet satisfies the criterion.<br><br>For example, Chandrasekaran discloses a stateless classifier in the access point, that performs packets classification to determine if the packet satisfies the information in the policy. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Chandrasekaran is found to not meet this limitation, checking, by the network node, if the packet satisfies the criterion would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>Chandrasekaran at Abstract ("In one embodiment, a method includes performing stateful application classification on packets received at a controller and transmitting classification information to an access point. The classification information includes flow information and stateless rules for applying policies. The access point is con-figured to use the classification information to perform state-less application classification and apply policies to packets received from a mobile device. An apparatus and logic are also disclosed herein.")<br><br>Chandrasekaran at [0007] ("In one embodiment, a method generally comprises performing stateful application classification on packets received at a controller and transmitting classification infor-mation to an access point. The classification information comprises flow information and stateless rules for applying policies. The access point is configured to use the classifica-tion information to perform stateless application classifica-tion and apply policies to packets received from a mobile device.")<br><br>Chandrasekaran at [0008] ("In another embodiment, an apparatus generally comprises a stateful classifier for performing stateful appli-cation classification at a controller, a classification database for storing classification information, and a processor for transmitting the classification information to an access point. The classification information comprises flow information and stateless rules for applying policies. The access point is configured to use the classification information to perform stateless application classification and apply policies to pack-ets received from a mobile device.")<br><br>Chandrasekaran at [0016] ("The wireless controller 12 includes a stateful appli-cation classifier 18 and the AP 14 includes a stateless appli-cation classifier 22. After the stateful |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 369 of 1100

| No. | '111 Patent Claim 1 | Chandrasekaran |
|---|---|---|
| | | classifier 18 identifies the application, the controller 12 transmits ( e.g., pushes) clas-sification information 26 to the AP 14 so that the AP can perform stateless classification and apply policies ( e.g., QoS or other policies) to traffic received from the mobile device 16. The controller 12 may also provide the classification information 26 to another AP 14 if the client 16 roams to a new AP, as shown in FIG. 1. Implementation of the stateful classifier 18 at the controller 12 and stateless classifier 22 at the AP 14 allows for policies to be applied for downstream traffic (packet 25) at the wireless controller 12, and for upstream traffic (packet 28) at the access point 14.")<br><br>Chandrasekaran at [0018] ("The stateless classifier 22 at the AP 14 uses rules that can act on a per packet basis in the flow. Stateless classifica-tion (also referred to as packet classification) is based on individual packet inspection ( e.g., 5 tuple, pattern matching) without knowledge of any related stream of packets, flows, sessions, or protocols.")<br><br>Chandrasekaran at Figure 3 (annotations added) |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 370 of 1100

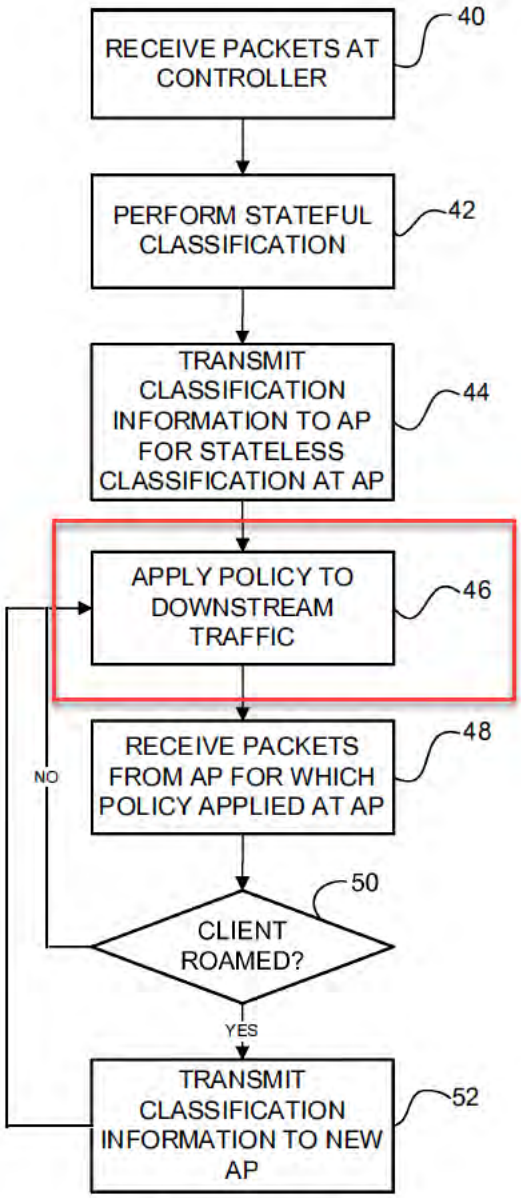



FIGURE 3



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 371 of 1100

| No. | '111 Patent Claim 1 | Chandrasekaran |
|---|---|---|
| | | Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Chandrasekaran in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 1[d] of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references.<br><br>For example, Kempf discloses determining by the network element if the packet header field match an associated action in the flow table.<br><br>Kempf at [0044] ("FIG. 1 is a diagram of one embodiment of an example network with an OpenFlow switch, conforming to the OpenFlow 1.0 specification. The OpenFlow 1.0 protocol enables a controller 101 to connect to an OpenFlow 1.0 enabled switch 109 using a secure channel 103 and control a single forwarding table 107 in the switch 109. The controller 101 is an external software component executed by a remote computing device that enables a user to configure the Open-Flow 1.0 switch 109. The secure channel 103 can be provided by any type of network including a local area network (LAN) or a wide area network (WAN), such as the Internet.")<br><br>Kempf at [0045] ("FIG. 2 is a diagram illustrating one embodiment of the contents of a flow table entry. The forwarding table 107 is populated with entries consisting of a rule 201 defining matches for fields in packet headers; an action 203 associated to the flow match; and a collection of statistics 205 on the flow. When an incoming packet is received a lookup for a matching rule is made in the flow table 107. If the incoming packet matches a particular rule, the associated action defined in that flow table entry is performed on the packet.")<br><br>Kempf at [0046] ("A rule 201 contains key fields from several headers in the protocol stack, for example source and destination Ethernet MAC addresses, source and destination IP addresses, IP protocol type number, incoming and outgoing TCP or UDP port numbers. To define a flow, all the available matching fields may be used. But it is also possible to restrict the matching rule to a subset of the available fields by using wildcards for the unwanted fields.") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 372 of 1100

| No. | '111 Patent Claim 1 | Chandrasekaran |
|---|---|---|
| | | Kempf at [0047] ("The actions that are defined by the specification of OpenFlow 1.0 are Drop, which drops the matching packets; Forward, which forwards the packet to one or all outgoing ports, the incoming physical port itself, the controller via the secure channel, or the local networking stack (if it exists). OpenFlow 1.0 protocol data units (PDU s) are defined with a set of structures specified using the C programming language. Some of the more commonly used messages are: report switch configuration message; modify state messages (in-cluding a modify flow entry message and port modification message); read state messages, where while the system is running, the datapath may be queried about its current state using this message; and send packet message, which is used when the controller wishes to send a packet out through the datapath.")<br><br>Kempf at [0050] ("FIG. 4 illustrates one embodiment of the processing of packets through an OpenFlow 1.1 switched packet pro-cessing pipeline. A received packet is compared against each of the flow tables 401. After each flow table match, the actions are accumulated into an action set. If processing requires matching against another flow table, the actions in the matched rule include an action directing processing to the next table in the pipeline. Absent the inclusion of an action in the set to execute all accumulated actions immediately, the actions are executed at the end 403 of the packet processing pipeline. An action allows the writing of data to a metadata register, which is carried along in the packet processing pipe-line like the packet header.")<br><br>Kempf at [0051] ("FIG. 5 is a flowchart of one embodiment of the OpenFlow 1.1 rule matching process. OpenFlow 1.1 contains support for packet tagging. OpenFlow 1.1 allows matching based on header fields and multi-protocol label switching (MPLS) labels. One virtual LAN (VLAN) label and one MPLS label can be matched per table. The rule matching process is initiated with the arrival of a packet to be processed (Block 501 ). Starting at the first table 0 a lookup is performed to determine a match with the received packet (Block 503). If there is no match in this table, then one of a set of default actions is taken (i.e., send packet to controller, drop the packet or continue to next table) (Block 509). If there is a match, then an update to the action set is made along with counters, packet or match set fields and meta data (Block 505). A check is made to determine the next table to process, which can be the next table sequentially or one specified by an action of a matching rule (Block 507). Once all of the tables have been processed, then the resulting action set is |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 373 of 1100

| No. | '111 Patent Claim 1 | Chandrasekaran |
|---|---|---|
| | | executed (Block 511). FIG. 6 is a diagram of the fields, which a matching process can utilize for identifying rules to apply to a packet.")<br><br>As another example, Swenson discloses determining by the steering device packet flows that match one or more signatures, conditions, or criteria of the packet.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0028] ("In contrast to conventional inline TCP throughput monitoring devices that monitor every single data packets transmitted and received, the network controller 140 is an "out-of-band" computer server that interfaces with the steer-ing device 130 to selectively inspect user flows of interest. The network controller 140 may further identify user flows (e.g., among the flows of interest) for optimization. In one embodiment, the network controller 140 may be imple-mented at the steering device 130 to monitor traffic. In other embodiments, the network controller 140 is coupled to and communicates with the steering device 130 for traffic moni-toring and optimization. When queried by the steering device 130, the network controller 140 determines if a given network flow should be ignored, monitored further or optimized. Opti-mization of a flow is often decided at the beginning of the flow because it is rarely possible to switch to optimized content mid-stream once non-optimized content delivery has begun. However, the network controller 140 may determine that existing flows associated with a particular subscriber or other entity should be optimized. In turn, new flows ( e.g., resulting from seek requests in media, new media requests, resume after pause, etc.) determined to be associated with the entity may be optimized. The network controller 140 uses the net-work state as well as historical traffic |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 374 of 1100

| No. | '111 Patent Claim 1 | Chandrasekaran |
|---|---|---|
| | | data in its decision for monitoring and optimization. Knowledge on the current net-work state, such as congestion, deems critical when it comes to data optimization.")<br><br>Swenson at [0045] ("The steering device interface 316 interacts with an external routing appliance, such as the steering device 130 to divert portions of the network traffic ( e.g., large object net-work flows). Existing routing appliances in most carrier net-works are designed to handle large amounts of network traf-fic. They are not, however, ideal devices to operate for monitoring and analysis individual flows. Through the steer-ing device interface 316, the network controller 140 may communicate with the external routing appliances, such as the steering device 130, to steer a portion of network traffic to the network controller 140 when certain conditions are met. Generally, network flows of interest to the network controller 140 contain larger media objects, such as videos and images. In one embodiment, the smaller flows, such as web page and text information, are not exchanged over the steering device interface 316.")<br><br>Swenson at [0059] ("In one embodiment, as the steering device 130 monitors network responses, it is looking for flows that match one or more signatures for video and images. When a match-ing flow is detected, the steering device 130 forwards the HTTP request and a portion of the HTTP response to the network controller 140 over the ICAP client interface 404. After receiving the request and the portion of response at the ICAP server interface 406, the flow analyzer 312 of the net-work controller 140 performs a deep flow inspection to deter-mine if the flow is worth bandwidth monitoring and/or user detection. For example, the flow inspection performed by the flow analyzer 312 may determine if the flow indeed contains large or medium object ( e.g., larger than 50 kB), and/or if the source IP address of the flow is from a user or a group of users that are required to be monitored by policies. The flow ana-lyzer 312 may also determine if the flow needs to be opti-mized based on historical flow statistical data.")<br><br>Swenson at [0060] ("If the flow is deemed of interest, the steering device 130 is notified to steer the flow through the network controller 140. This is known as the "continue" working mode for bandwidth monitoring. In the "continue" mode, the network controller 140 interfaces with the steering device 130 to func-tion, on-demand, as a traditional inline network element for flows deemed of interest. Thus, the network controller 140 ingests the network flow for inspection and subsequently forwards the network flow on the network |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 375 of 1100

| No. | '111 Patent Claim 1 | Chandrasekaran |
|---|---|---|
| | | response path. For example, for this particular flow, the origin server 160 responds to the user request by sending video or images over the network link 413 to the steering device 130, which for-wards the video or images to the network controller 140 over a network link 414. After the network controller 140 updates the flow statistics, the video or images are returned to the steering device 130 over a network link 415, which transmits the video or images to the user device 110 over the network link 416.")<br><br>Swenson at [0065] ("FIG. 5 is a block diagram illustrating an example event trace of "continue" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 512 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 512 to the intended origin server 160 and receives a response 514 back from the origin server 160. The steering device 130 then sends an ICAP request message 516 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 518. Upon receiving the instruc-tion, the steering device 130 re-writes the response 514 to an HTTP redirect response 520, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 520 directly to the user device 110. In case the flow dose not contain video or image objects, or the network controller 140 determines not to monitor the flow, the steering device 13 0 would forward the response to the user device 110.")<br><br>Swenson at [0069] ("FIG. 6 is a block diagram illustrating an example event trace of "counting" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 612 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 612 to the intended origin server 160 and receives a response 614 back from the origin server 160. The steering device 130 then sends an ICAP request message 616 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 376 of 1100

| No. | '111 Patent Claim 1 | Chandrasekaran |
|---|---|---|
| | | inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 618. Upon receiving the instruc-tion, the steering device 130 re-writes the response 614 to an HTTP redirect response 620, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 620 directly to the user device 110. In case the flow dose not contain video or image objects that need to be redirected, the steering device 130 would forward the response to the user device 110.")<br><br>Swenson at [0079] ("In the bandwidth calculation, flows are categorized into buckets based on the size of the objects being transferred. Small objects may not be factored into the bandwidth calcu-lation since they may come and go within a single interval. For example, flows with payload size less than 50 kB may be ignored because a transfer of 50 kB may never reach the full potential throughput of the link. While larger flows may reach the full throughput of the link for a long period of time intervals, they are grouped into 50-75 kB, 75-100 kB and 100 kB+ buckets because the characteristics of these flow sizes can be different, hence the bandwidth for each of the buckets is measured and calculated separately. In other embodiments, the flow size ranges (e.g., 50-75 kB, 75-100 kB and l00kB+) of the buckets may be altered depending on the network traffic and size of objects transmitted. Furthermore, the bucket sizes can also be adjusted based on network topology, such as buffer size, prior to transmission to the client. The calculated bandwidth per bucket is stored in a queue structure that allows for the computing and updating of minimum, maximum, and/or average measurements for each bucket. In one embodiment, the 100 kB+ bucket's current tail entry is checked against the average bandwidth for the 100 kB+ bucket. If the current entry is less than the average multiplied by the number of entries in the queue, the current entry is added to the bandwidth calculation for the current interval. This scheme can filter out large bursts of data from tempo-rarily idle flows. If the bandwidth exceeds the value, a number of bytes (e.g., 125 kB) will be subtracted from the current entry to account for TCP buffers in the network.") |
| 1[e] | responsive to the packet not satisfying the criterion, sending, by the network node | Chandrasekaran discloses responsive to the packet not satisfying the criterion, sending, by the network node over the packet network, the packet to the second entity. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 377 of 1100

| No. | '111 Patent Claim 1 | Chandrasekaran |
|---|---|---|
| | over the packet network, the packet to the second entity; and | For example, Chandrasekaran discloses in response to particular policies, not sending the packet to the controller and sending the packet to the second entity. A person of ordinary skill in the art would understand that Chandrasekaran discloses a number of embodiments in which a packet may not be sent to the controller in response to not satisfying the criterion and sending the packet to the second entity. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Chandrasekaran is found to not meet this limitation, responsive to the packet not satisfying the criterion, sending, by the network node over the packet network, the packet to the second entity would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>Chandrasekaran at [0032] ("It is to be understood that the process illustrated in FIG. 3 and described above is only an example and that steps may be modified, deleted, added, or combined without departing from the scope of the embodiments. For example, if traffic from the network destined for the mobile device 16 does not pass through the controller 12, policies are not applied by the controller for downstream traffic as shown in step 46. Also, if the policy applied at the AP 14 is to drop packets, those packets will not be received at the controller as shown in step 48.")<br><br>Chandrasekaran at Figure 3 (annotations added) |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 378 of 1100

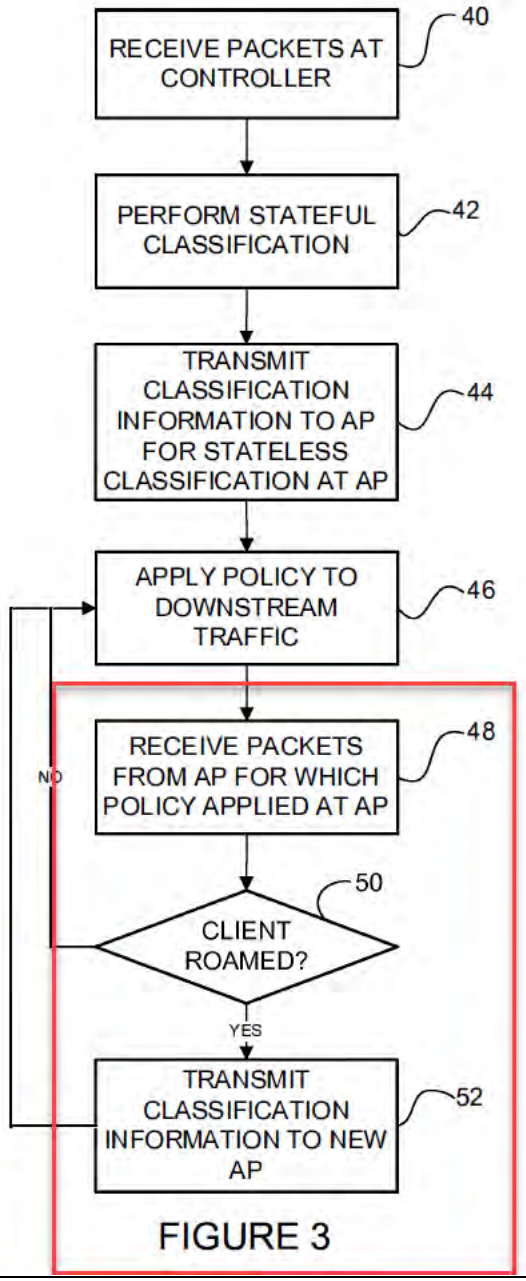



FIGURE 3



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 379 of 1100

| No. | '111 Patent Claim 1 | Chandrasekaran |
|---|---|---|
| | | Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Chandrasekaran in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 1[e] of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references.<br><br>For example, Kempf discloses sending the packet from the network element to the destination device in response to the packet not matching the action in the flow table.<br><br>Kempf at [0044] ("FIG. 1 is a diagram of one embodiment of an example network with an OpenFlow switch, conforming to the OpenFlow 1.0 specification. The OpenFlow 1.0 protocol enables a controller 101 to connect to an OpenFlow 1.0 enabled switch 109 using a secure channel 103 and control a single forwarding table 107 in the switch 109. The controller 101 is an external software component executed by a remote computing device that enables a user to configure the Open-Flow 1.0 switch 109. The secure channel 103 can be provided by any type of network including a local area network (LAN) or a wide area network (WAN), such as the Internet.")<br><br>Kempf at [0045] ("FIG. 2 is a diagram illustrating one embodiment of the contents of a flow table entry. The forwarding table 107 is populated with entries consisting of a rule 201 defining matches for fields in packet headers; an action 203 associated to the flow match; and a collection of statistics 205 on the flow. When an incoming packet is received a lookup for a matching rule is made in the flow table 107. If the incoming packet matches a particular rule, the associated action defined in that flow table entry is performed on the packet.")<br><br>Kempf at [0046] ("A rule 201 contains key fields from several headers in the protocol stack, for example source and destination Ethernet MAC addresses, source and destination IP addresses, IP protocol type number, incoming and outgoing TCP or UDP port numbers. To define a flow, all the available matching fields may be used. But it is also possible to restrict the matching rule to a subset of the available fields by using wildcards for the unwanted fields.") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 380 of 1100

| No. | '111 Patent Claim 1 | Chandrasekaran |
|---|---|---|
| | | Kempf at [0047] ("The actions that are defined by the specification of OpenFlow 1.0 are Drop, which drops the matching packets; Forward, which forwards the packet to one or all outgoing ports, the incoming physical port itself, the controller via the secure channel, or the local networking stack (if it exists). OpenFlow 1.0 protocol data units (PDU s) are defined with a set of structures specified using the C programming language. Some of the more commonly used messages are: report switch configuration message; modify state messages (in-cluding a modify flow entry message and port modification message); read state messages, where while the system is running, the datapath may be queried about its current state using this message; and send packet message, which is used when the controller wishes to send a packet out through the datapath.")<br><br>Kempf at [0050] ("FIG. 4 illustrates one embodiment of the processing of packets through an OpenFlow 1.1 switched packet pro-cessing pipeline. A received packet is compared against each of the flow tables 401. After each flow table match, the actions are accumulated into an action set. If processing requires matching against another flow table, the actions in the matched rule include an action directing processing to the next table in the pipeline. Absent the inclusion of an action in the set to execute all accumulated actions immediately, the actions are executed at the end 403 of the packet processing pipeline. An action allows the writing of data to a metadata register, which is carried along in the packet processing pipe-line like the packet header.")<br><br>Kempf at [0051] ("FIG. 5 is a flowchart of one embodiment of the OpenFlow 1.1 rule matching process. OpenFlow 1.1 contains support for packet tagging. OpenFlow 1.1 allows matching based on header fields and multi-protocol label switching (MPLS) labels. One virtual LAN (VLAN) label and one MPLS label can be matched per table. The rule matching process is initiated with the arrival of a packet to be processed (Block 501 ). Starting at the first table 0 a lookup is performed to determine a match with the received packet (Block 503). If there is no match in this table, then one of a set of default actions is taken (i.e., send packet to controller, drop the packet or continue to next table) (Block 509). If there is a match, then an update to the action set is made along with counters, packet or match set fields and meta data (Block 505). A check is made to determine the next table to process, which can be the next table sequentially or one specified by an action of a matching rule (Block 507). Once all of the tables have been processed, then the resulting action set is |

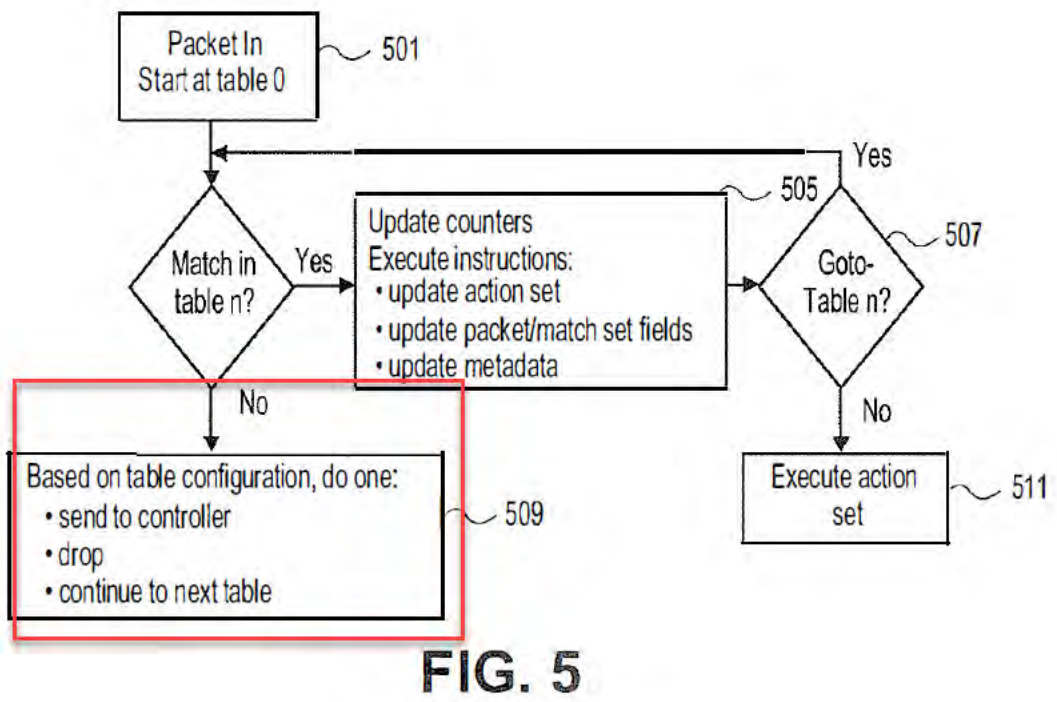


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 381 of 1100

| No. | '111 Patent Claim 1 | Chandrasekaran |
|---|---|---|
| | | executed (Block 511). FIG. 6 is a diagram of the fields, which a matching process can utilize for identifying rules to apply to a packet.")<br><br>Kempf at [0053] ("In one embodiment, a group table can be supported in conjunction with the OpenFlow 1.1 protocol. Group tables enable a method for allowing a single flow match to trigger forwarding on multiple ports. Group table entries consist of four fields: a group identifier, which is a 32 bit unsigned integer identifying the group; a group type that determines the group's semantics; counters that maintain statistics on the group; and an action bucket list, which is an ordered list of action buckets, where each bucket contains a set of actions to execute together with their parameters.")<br><br>Kempf at [0091] ("When a packet header matches a rule associated with the virtual port, the GTP TEID is written into the lower 32 bits of the metadata and the packet is directed to the virtual port. The virtual port calculates the hash of the TEID and looks up the tunnel header information in the tunnel header table. If no such tunnel information is present, the packet is forwarded to the controller with an error indication. Other-wise, the virtual port constructs a GTP tunnel header and encapsulates the packet. Any DSCP bits or VLAN priority bits are additionally set in the IP or MAC tunnel headers, and any VLAN tags or MPLS labels are pushed onto the packet. The encapsulated packet is forwarded out the physical port to which the virtual port is bound.")<br><br>Kempf at [0092] ("In one embodiment, the system implements a GTP fast path decapsulation virtual port. When requested by the S-GW and P-GW control plane software running in the cloud computing system, the gateway switch installs rules and actions for routing GTP encapsulated packets out of GTP tunnels. The rules match the GTP header flags and the GTP TEID for the packet, in the modified OpenFlow flow table shown in FIG. 17 as follows: the IP destination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); the UDP destination port is the GTP-U destination port (2152); and the header fields and message type field is wildcarded with the flag 0XFFF0 and the upper two bytes of the field match the G-PDU message type (255) while the lower two bytes match 0x30, i.e. the packet is a GTP packet not a GTP' packet and the version number is 1.")<br><br>Kempf at Figure 5 (annotation added) |

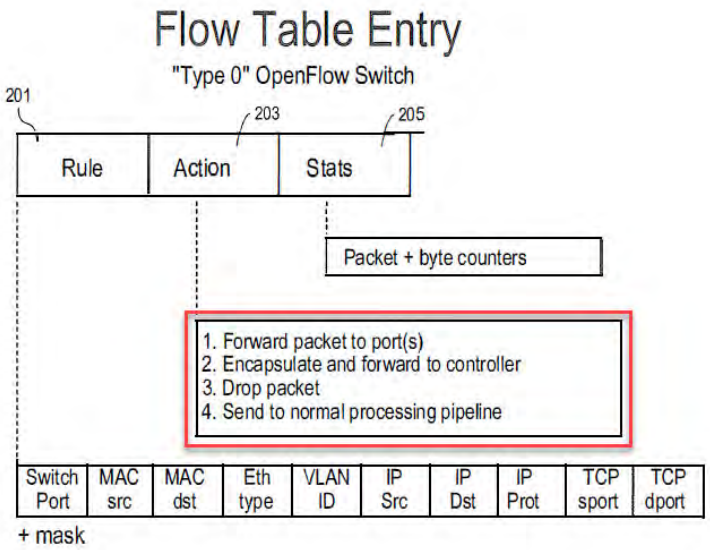


~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 382 of 1100

| No. | '111 Patent Claim 1 | Chandrasekaran |
|---|---|---|
| | |  <br>FIG. 5<br>Kempf at Figure 2 (annotation added) |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 383 of 1100

| No. | '111 Patent Claim 1 | Chandrasekaran |
|---|---|---|
| | |  <br>**FIG. 2**<br><br>As another example, Swenson discloses monitoring and categorizing network traffic by the steering device based on instructions and desired criteria sent by the network controller to determine if packet flows require further inspection. Based on the instruction and desired criteria, the network controller monitors and optimizes only a subset of network traffic. Packet flows that do not meet the desired criteria from the network controller's instructions at the steering device are not sent for further inspection and are sent to their originally intended destination.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 384 of 1100

| No. | '111 Patent Claim 1 | Chandrasekaran |
|---|---|---|
| | | device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0028] ("In contrast to conventional inline TCP throughput monitoring devices that monitor every single data packets transmitted and received, the network controller 140 is an "out-of-band" computer server that interfaces with the steer-ing device 130 to selectively inspect user flows of interest. The network controller 140 may further identify user flows (e.g., among the flows of interest) for optimization. In one embodiment, the network controller 140 may be imple-mented at the steering device 130 to monitor traffic. In other embodiments, the network controller 140 is coupled to and communicates with the steering device 130 for traffic moni-toring and optimization. When queried by the steering device 130, the network controller 140 determines if a given network flow should be ignored, monitored further or optimized. Opti-mization of a flow is often decided at the beginning of the flow because it is rarely possible to switch to optimized content mid-stream once non-optimized content delivery has begun. However, the network controller 140 may determine that existing flows associated with a particular subscriber or other entity should be optimized. In turn, new flows ( e.g., resulting from seek requests in media, new media requests, resume after pause, etc.) determined to be associated with the entity may be optimized. The network controller 140 uses the net-work state as well as historical traffic data in its decision for monitoring and optimization. Knowledge on the current net-work state, such as congestion, deems critical when it comes to data optimization.")<br><br>Swenson at [0038] ("Turning back to FIG. 1, the network controller 140 allows network operators to apply fine granular optimization policies to ensure high quality of experience (QoE) based on cell tower congestion, device types, subscriber profiles and service plans with lower hardware and software costs. The architecture of the network controller 140 provides an excel-lent fit for the net neutrality guideline of"reasonable network management", and better compliance to the copyright law (DMCA) than solutions that rely on long-term caching. Hav-ing the ability of monitoring network traffic on a per sub-scriber, per flow, or per video file basis, the network controller 140 also selectively monitors and |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 385 of 1100

| No. | '111 Patent Claim 1 | Chandrasekaran |
|---|---|---|
| | | optimizes only a subset of traffic that benefits from optimization the most, thus achiev-ing both scalability and efficiency for optimization at a com-petitive price-point. The core element of the network control-ler 140 lies in its mechanisms for congestion detection and mitigation, which allows optimization resources to be utilized in the most efficient and surgical manner.")<br><br>Swenson at [0042] ("The network controller 140 collects real-time statis-tical data on the network flows from core network side with-out probes deployed in the RAN network. The statistical data is stored and compared against historical flow data to estimate level of congestion and available network bandwidth. Instead of collecting traffic statistics for every flow and every session, the network controller 140 samples only large flows involving media objects such as videos and images above a certain size ( e.g., above 50 kB). The network controller 140 can choose to be a pass-through device to monitor the large flows as well as to determine whether to optimize the flows. Measuring only larger flows has the advantage to mitigate corruptions caused by origin server latency and network glitches. Furthermore, focusing on the large flows helps the network controller to reduce the background noise and to increase noise-to-signal ratio in bandwidth measuring by removing the impact of millions of tiny or small flows with delivery time in millisec-onds. Therefore the reliability of bandwidth estimation and congestion detection is much higher.")<br><br>Swenson at [0045] ("The steering device interface 316 interacts with an external routing appliance, such as the steering device 130 to divert portions of the network traffic ( e.g., large object net-work flows). Existing routing appliances in most carrier net-works are designed to handle large amounts of network traf-fic. They are not, however, ideal devices to operate for monitoring and analysis individual flows. Through the steer-ing device interface 316, the network controller 140 may communicate with the external routing appliances, such as the steering device 130, to steer a portion of network traffic to the network controller 140 when certain conditions are met. Generally, network flows of interest to the network controller 140 contain larger media objects, such as videos and images. In one embodiment, the smaller flows, such as web page and text information, are not exchanged over the steering device interface 316.")<br><br>Swenson at [0059] ("In one embodiment, as the steering device 130 monitors network responses, it is looking for flows that match one or more signatures for video and images |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 386 of 1100

| No. | '111 Patent Claim 1 | Chandrasekaran |
|---|---|---|
| | | When a match-ing flow is detected, the steering device 130 forwards the HTTP request and a portion of the HTTP response to the network controller 140 over the ICAP client interface 404. After receiving the request and the portion of response at the ICAP server interface 406, the flow analyzer 312 of the net-work controller 140 performs a deep flow inspection to deter-mine if the flow is worth bandwidth monitoring and/or user detection. For example, the flow inspection performed by the flow analyzer 312 may determine if the flow indeed contains large or medium object ( e.g., larger than 50 kB), and/or if the source IP address of the flow is from a user or a group of users that are required to be monitored by policies. The flow ana-lyzer 312 may also determine if the flow needs to be opti-mized based on historical flow statistical data.")<br><br>Swenson at [0060] ("If the flow is deemed of interest, the steering device 130 is notified to steer the flow through the network controller 140. This is known as the "continue" working mode for bandwidth monitoring. In the "continue" mode, the network controller 140 interfaces with the steering device 130 to func-tion, on-demand, as a traditional inline network element for flows deemed of interest. Thus, the network controller 140 ingests the network flow for inspection and subsequently forwards the network flow on the network response path. For example, for this particular flow, the origin server 160 responds to the user request by sending video or images over the network link 413 to the steering device 130, which for-wards the video or images to the network controller 140 over a network link 414. After the network controller 140 updates the flow statistics, the video or images are returned to the steering device 130 over a network link 415, which transmits the video or images to the user device 110 over the network link 416.")<br><br>Swenson at [0065] ("FIG. 5 is a block diagram illustrating an example event trace of "continue" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 512 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 512 to the intended origin server 160 and receives a response 514 back from the origin server 160. The steering device 130 then sends an ICAP request message 516 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 387 of 1100

| No. | '111 Patent Claim 1 | Chandrasekaran |
|---|---|---|
| | | case, the network controller 140 responds with a redirect to optimize the video in ICAP response 518. Upon receiving the instruc-tion, the steering device 130 re-writes the response 514 to an HTTP redirect response 520, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 520 directly to the user device 110. In case the flow dose not contain video or image objects, or the network controller 140 determines not to monitor the flow, the steering device 13 0 would forward the response to the user device 110.")<br><br>Swenson at [0069] ("FIG. 6 is a block diagram illustrating an example event trace of "counting" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 612 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 612 to the intended origin server 160 and receives a response 614 back from the origin server 160. The steering device 130 then sends an ICAP request message 616 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 618. Upon receiving the instruc-tion, the steering device 130 re-writes the response 614 to an HTTP redirect response 620, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 620 directly to the user device 110. In case the flow dose not contain video or image objects that need to be redirected, the steering device 130 would forward the response to the user device 110.") |
| 1[f] | responsive to the packet satisfying the criterion, sending the packet, by the network node over the packet network, to an entity that is included in the | Chandrasekaran discloses responsive to the packet satisfying the criterion, sending the packet, by the network node over the packet network, to an entity that is included in the instruction and is other than the second entity.<br><br>For example, Chandrasekaran discloses the controller receiving traffic for which initial classifications have been applied by the access node. |

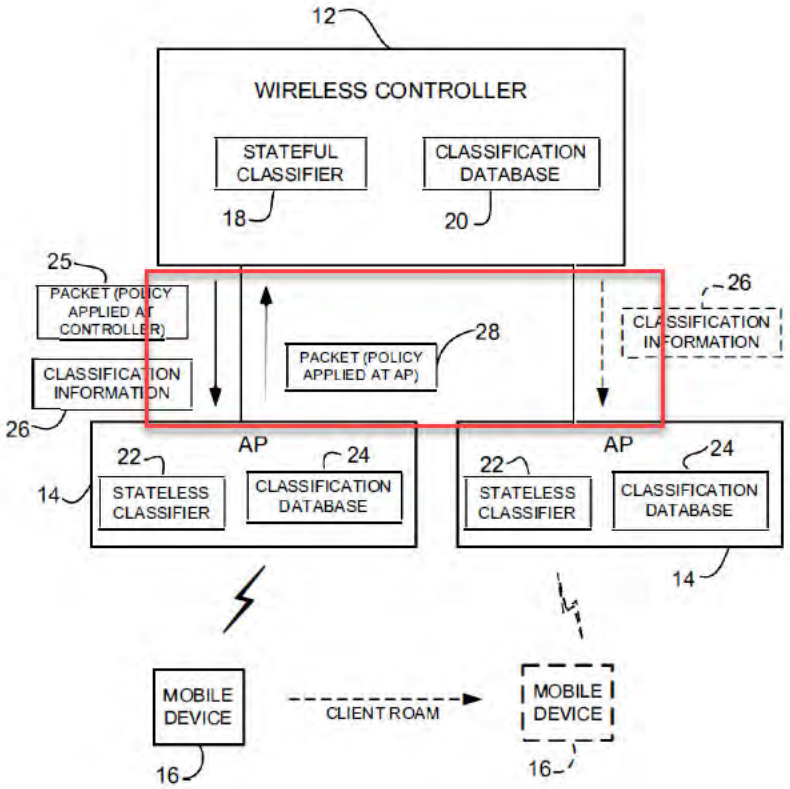


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 388 of 1100

| No. | '111 Patent Claim 1 | Chandrasekaran |
|---|---|---|
| | instruction and is other than the second entity. | Chandrasekaran at [0016] ("The wireless controller 12 includes a stateful appli-cation classifier 18 and the AP 14 includes a stateless appli-cation classifier 22. After the stateful classifier 18 identifies the application, the controller 12 transmits ( e.g., pushes) clas-sification information 26 to the AP 14 so that the AP can perform stateless classification and apply policies ( e.g., QoS or other policies) to traffic received from the mobile device 16. The controller 12 may also provide the classification information 26 to another AP 14 if the client 16 roams to a new AP, as shown in FIG. 1. Implementation of the stateful classifier 18 at the controller 12 and stateless classifier 22 at the AP 14 allows for policies to be applied for downstream traffic (packet 25) at the wireless controller 12, and for upstream traffic (packet 28) at the access point 14.")<br><br>Chandrasekaran at [0031] ("FIG. 3 is a flowchart illustrating an example of a process at the controller 12 for classification of traffic for application aware policies in a wireless network, in accor-dance with one embodiment. At step 40, the controller 12 receives packets belonging to a network flow. The controller 12 performs stateful classification to identify an application associated with the flow ( step 42). The controller 12 transmits classification information ( e.g., flow information, stateless DPI rule, and policy) to the AP 14 for use in stateless classi-fication at the AP (step 44). The controller 12 applies policies to downstream traffic (received at the controller and destined for the client 16) (step 46) and receives upstream traffic for which policies have been applied at the AP 14 (step 48). If the controller 12 determines ( e.g., receives an indication) that the client 16 has roamed, it transmits the classification informa-tion to the new AP 14 to which the client has roamed (steps 50 and 52).")<br><br>Chandrasekaran at [0033] ("The following describes an example of the above process for WebEx traffic that has different sub-classifications for voice and video traffic. Stateful classification is first performed by the controller 12 at the beginning of the flow. The controller 12 may need to process, for example, 10, 100, or any other number of packets to classify the flow as Web Ex traffic. Once the classification is performed, the controller 12 sends the stateless DPI rules and flow information to the AP 14 for stateless sub-classification to distinguish voice, video, or data within a WebEx flow. For example, after the controller 12 identifies the WebEx meeting traffic, it pushes the tuple, the stateless DPI rules (as shown below), and policies to the AP 14 for upstream traffic marking, dropping, or |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 389 of 1100

| No. | '111 Patent Claim 1 | Chandrasekaran |
|---|---|---|
| | | rate-limit-ing. If the client 16 roams, the controller 12 transmits the same classification information to the new AP to which the client has roamed.")<br><br>Chandrasekaran at Figure 1 (annotations added)<br><br>  <br><br>FIGURE 1 |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 390 of 1100

| No. | '111 Patent Claim 2 | Chandrasekaran |
|---|---|---|
| 2[a] | The method according to claim 1, wherein the instruction is 'probe', 'mirror', or 'terminate' instruction, and | Chandrasekaran discloses the method according to claim 1, wherein the instruction is 'probe', 'mirror', or 'terminate' instruction.<br><br>For example, Chandrasekaran discloses policies, which may include classification, copying, or dropping policies or actions. A person of ordinary skill in the art would understand that Chandrasekaran discloses many different policies, for example, probing, mirroring, or terminating packet flows. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Chandrasekaran is found to not meet this limitation, wherein the instruction is 'probe', 'mirror', or 'terminate' instruction would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>Chandrasekaran at Abstract ("In one embodiment, a method includes performing stateful application classification on packets received at a controller and transmitting classification information to an access point. The classification information includes flow information and stateless rules for applying policies. The access point is con-figured to use the classification information to perform state-less application classification and apply policies to packets received from a mobile device. An apparatus and logic are also disclosed herein.")<br><br>Chandrasekaran at [0007] ("In one embodiment, a method generally comprises performing stateful application classification on packets received at a controller and transmitting classification infor-mation to an access point. The classification information comprises flow information and stateless rules for applying policies. The access point is configured to use the classifica-tion information to perform stateless application classifica-tion and apply policies to packets received from a mobile device.")<br><br>Chandrasekaran at [0008] ("In another embodiment, an apparatus generally comprises a stateful classifier for performing stateful appli-cation classification at a controller, a classification database for storing classification information, and a processor for transmitting the classification information to an access point. The classification information comprises flow information and stateless rules for applying policies. The access point is configured to use the classification information to perform stateless application classification and apply policies to pack-ets received from a mobile device.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 391 of 1100

| No. | '111 Patent Claim 2 | Chandrasekaran |
|---|---|---|
| | | Chandrasekaran at [0023] ("In one embodiment, the classification information 26 transmitted from the controller 12 to the AP 14 includes tuple information for a flow ( e.g., source IP address, destina-tion IP address, source port, destination port, and protocol), application identifier (ID), and stateless DPI information. Stateless DPI information includes classification and sub-classification information ( e.g., fixed or variable offset with a pattern or regular expression) and rules for applying policies on the sub-classified packets. The policies may include, for example, drop packet, mark a DSCP (Differentiated Services Code Point) value in the packet, or rate limit the traffic.")<br><br>Chandrasekaran at [0033] ("The following describes an example of the above process for WebEx traffic that has different sub-classifications for voice and video traffic. Stateful classification is first performed by the controller 12 at the beginning of the flow. The controller 12 may need to process, for example, 10, 100, or any other number of packets to classify the flow as Web Ex traffic. Once the classification is performed, the controller 12 sends the stateless DPI rules and flow information to the AP 14 for stateless sub-classification to distinguish voice, video, or data within a WebEx flow. For example, after the controller 12 identifies the WebEx meeting traffic, it pushes the tuple, the stateless DPI rules (as shown below), and policies to the AP 14 for upstream traffic marking, dropping, or rate-limit-ing. If the client 16 roams, the controller 12 transmits the same classification information to the new AP to which the client has roamed.")<br><br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Chandrasekaran in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 2(a) of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references.<br><br>For example, Chua discloses programming network nodes with redirecting, mirroring, and blocking programmed actions.<br><br>Chua at 7:28-54 ("SDN controller 112 may receive data as input from service devices 116. For example, SDN controller 112 may be con-figured to receive data from an intrusion detection system (IDS) device, a Denial of Service (DoS) device, a Distributed Denial of |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 392 of 1100

| No. | '111 Patent Claim 2 | Chandrasekaran |
|---|---|---|
| | | Service (DDoS) device, an intrusion prevention system (IPS) device, or the like. Based on this information, SDN controller 112 may make network enforcement decisions for specific traffic flows. That is, SDN controller 112 may program network devices of SDN 106 to perform pro-grammed actions on packets of a packet flow based on this data. Such programmed actions may include:<br><br>Allow-explicitly allow a certain network flow to proceed to its destination<br>Block-explicitly block a certain flow from traversing SDN 106<br>Mirror-allow the traffic, but send a copy of the traffic for deeper inspection or recording to, e.g., one of service devices 116<br>Redirect-redirect the traffic to another network (such as a honeypot device or other device of service devices 116) for either inspection or to keep a potential hacker 'busy' to determine if there is a real security threat.<br>Transform-modify or translate values of headers of packets in the network flow<br>Encapsulate-encapsulate packets in the network flow with a particular header")<br><br>Chua at 28:7-32 ("In addition, SDN controller 112 may configure the service device to send service-related data to one or more network devices (334). The service-related data may cause the net-work devices to change a path along which the packet is forwarded. For example, when the service device is a security device (e.g., a firewall or an IDS), if the security device determines that one or more packets of a packet flow are malicious, the security device may send service data indicat-ing that the packet flow includes malicious data. SDN con-troller 112 may program the network devices of the SDN to perform a programmed action based on the service-related data (336). For example, SDN controller 112 may program network devices to, in response to an indication that packets of a packet flow include malicious data, forward packets of the packet flow to a destination of the packet flow, forward packets of malicious packet flows to a collection device for further analysis, cause network devices to drop packets of the malicious packet flows, send a close session message to devices from which packets of the malicious packet flows were received, block the packets of the packet flow, mirror copies of the packets of the packet flow to a second service device while forwarding the packets of the packet flow to the destination of the packet flow, redirect the packets of the packet flow to a third service device, transform one or more values of headers of the packets, and/or encapsulate the pack-ets with a particular header, or other such actions.") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 393 of 1100

| No. | '111 Patent Claim 2 | Chandrasekaran |
|---|---|---|
| | | As another example, Copeland discloses probing, copying, and terminating rules configured on the network device.<br><br>Copeland at [0057] ("In accordance with an aspect of the invention, a flow is considered terminated after a predetermined period of time has elapsed on a particular connection or port. For example, if HTTP Web traffic on port 80 ceases for a predetermined period of time, but other traffic begins to occur on port 80 after the expiration of that predetermined time period, it is considered that a new flow has begun, and the system responds accordingly to assign a new flow number and track the statistics and characteristics thereof. In the disclosed embodiment, the predetermined time period is 330 seconds, but those skilled in the art will understand that this time is arbitrary and may be heuristically adjusted.")<br><br>Copeland at [0082] ("Following the reserved field, the next 6 bits are a series of one-bit flags, shown in FIG. 2 as flags U, A, P, R, S, F. The first flag is the urgent flag (U). If the U flag is set, it indicates that the urgent pointer is valid and points to urgent data that should be acted upon as soon as possible. The next flag is the A ( or ACK or "acknowledgment") flag. The ACK flag indicates that an acknowledgment number is valid, and acknowledges that data has been received. The next flag, the push (P) flag, tells the receiving end to push all buffered data to the receiving application. The reset (R) flag is the following flag, which terminates both ends of the TCP connection. Next, the S (or SYN for "synchronize") flag is set in the initial packet of a TCP connection where both ends have to synchronize their TCP buffers. Following the SYN flag is the F (for FIN or "finish") flag. This flag signifies that the sending end of the communication and the host will not send any more data but still may acknowledge data that is received.")<br><br>Copeland at [0093] ("As illustrated, when Hostl terminates its end of the session, it sends a packet with the FIN and ACK flags set. The FIN flag informs Host2 that Hostl will send no more data. The ACK flag acknowledges the last data received by Hostl by informing Host2 of the next sequence number it expects to receive.")<br><br>Copeland at [0095] ("When Host 2 is ready to terminate the session, it sends its own packet with the FIN and ACK flags set. Hostl responds that it has received the final packet with an |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 394 of 1100

| No. | '111 Patent Claim 2 | Chandrasekaran |
|---|---|---|
| | | ACK packet providing to Host2 an acknowledgment number one greater than the sequence number provided in the FIN-ACK packet of Host2.")<br><br>Copeland at [0099] ("As another example, if a particular host sends a large number of SYN packets to a target host and in response receives numerous R packets from the targeted host, a potential TCP probe is indicated. Likewise, numerous UDP packets sent from one host to a targeted host and numerous ICMP "port unavailable" packets received from the targeted host indicate a potential UDP probe. A stealth probe is indicated by multiple packets from the same source port number sent to different port numbers on a targeted host.")<br><br>Copeland at [0107] ("A flow is terminated if no communications occur between the two IP addresses and the one low port ( e.g. port 80) for 330 seconds. Most Web browsers or a TCP connec-tion send a reset packet (i.e. a packet with the R flag set) if no communications are sent or received for 5 minutes. An analysis can determine if the flow is abnormal or not for HTTP communications.")<br><br>Copeland at [0123] ("Flow processing is done for TCP and UDP packets, and the port numbers in the transport layer header are used to identify the flow record to be updated. For ICMP packets that constitute rejections of a packet, the copy of the rejected packet in the ICMP data field is used to identify the IP addresses and port numbers of the corresponding flow.")<br><br>Copeland at [0145] ("A list IP of addresses contacted or probed by each host can be maintained. When this list indicates that more than a threshold number of other hosts (e.g., 8) have been contacted in the same subnet, CI is added to the to the host and a bit in the host record is set to indicate that the host has received CI for "address scanning." Note that the number of hosts to designate a scan is not required to be a fixed value, but could be adjusted based on the sample rate or other means to enhance the accuracy making the number of hosts scanned "statistically significant". These and other values of concern index are shown for non-flow based events in FIG. 7.")<br><br>Copeland at [0158] ("Flow processing is done for TCP and UDP packets, and the port numbers in the transport layer header are used to identify the flow record to be updated. For ICMP packets that constitute rejections of a packet, the copy of the rejected packet in the |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 395 of 1100

| No. | '111 Patent Claim 2 | Chandrasekaran |
|---|---|---|
| | | ICMP data field is used to identify the IP addresses and port numbers of the corresponding flow.") |
| 2[b] | upon receiving by the network node the 'terminate' instruction, the method further comprising blocking, by the network node, the packet from being sent to the second entity and to the controller. | Chandrasekaran discloses upon receiving by the network node the 'terminate' instruction, the method further comprising blocking, by the network node, the packet from being sent to the second entity and to the controller.<br><br>For example, Chandrasekaran discloses dropping packets by the access point in response to a dropping instruction.<br><br>Chandrasekaran at [0023] ("In one embodiment, the classification information 26 transmitted from the controller 12 to the AP 14 includes tuple information for a flow ( e.g., source IP address, destina-tion IP address, source port, destination port, and protocol), application identifier (ID), and stateless DPI information. Stateless DPI information includes classification and sub-classification information ( e.g., fixed or variable offset with a pattern or regular expression) and rules for applying policies on the sub-classified packets. The policies may include, for example, drop packet, mark a DSCP (Differentiated Services Code Point) value in the packet, or rate limit the traffic.")<br><br>Chandrasekaran at [0032] ("It is to be understood that the process illustrated in FIG. 3 and described above is only an example and that steps may be modified, deleted, added, or combined without departing from the scope of the embodiments. For example, if traffic from the network destined for the mobile device 16 does not pass through the controller 12, policies are not applied by the controller for downstream traffic as shown in step 46. Also, if the policy applied at the AP 14 is to drop packets, those packets will not be received at the controller as shown in step 48.")<br><br>Chandrasekaran at [0033] ("The following describes an example of the above process for WebEx traffic that has different sub-classifications for voice and video traffic. Stateful classification is first performed by the controller 12 at the beginning of the flow. The controller 12 may need to process, for example, 10, 100, or any other number of packets to classify the flow as Web Ex traffic. Once the classification is performed, the controller 12 sends the stateless DPI rules and flow information to the AP 14 for stateless sub-classification to distinguish voice, video, or data within a WebEx flow. For example, after |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 396 of 1100

| No. | '111 Patent Claim 2 | Chandrasekaran |
|---|---|---|
| | | the controller 12 identifies the WebEx meeting traffic, it pushes the tuple, the stateless DPI rules (as shown below), and policies to the AP 14 for upstream traffic marking, dropping, or rate-limit-ing. If the client 16 roams, the controller 12 transmits the same classification information to the new AP to which the client has roamed.") |

| No. | '111 Patent Claim 3 | Chandrasekaran |
|---|---|---|
| 3[a] | The method according to claim 1, wherein the instruction is a 'probe', a 'mirror', or a 'terminate' instruction, and | Chandrasekaran discloses the method according to claim 1, wherein the instruction is a 'probe', a 'mirror', or a 'terminate' instruction.<br><br>*See supra* at 2(a). |
| 3[b] | upon receiving by the network node the 'mirror' instruction and responsive to the packet satisfying the criterion, the method further comprising sending the packet, by the network node, to the second entity and to the controller. | Chandrasekaran discloses upon receiving by the network node the 'mirror' instruction and responsive to the packet satisfying the criterion, the method further comprising sending the packet, by the network node, to the second entity and to the controller.<br><br>For example, Chandrasekaran discloses policies which may include copying, policies or actions. A person of ordinary skill in the art would understand that Chandrasekaran discloses many different policies, for example mirroring packet flows. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Chandrasekaran is found to not meet this limitation, upon receiving by the network node the 'mirror' instruction and responsive to the packet satisfying the criterion, method further comprising sending the packet, by the network node, to the second entity and to the controller would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>Chandrasekaran at Abstract ("In one embodiment, a method includes performing stateful application classification on packets received at a controller and transmitting classification |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 397 of 1100

| No. | '111 Patent Claim 3 | Chandrasekaran |
|---|---|---|
| | | information to an access point. The classification information includes flow information and stateless rules for applying policies. The access point is con-figured to use the classification information to perform state-less application classification and apply policies to packets received from a mobile device. An apparatus and logic are also disclosed herein.")<br><br>Chandrasekaran at [0007] ("In one embodiment, a method generally comprises performing stateful application classification on packets received at a controller and transmitting classification infor-mation to an access point. The classification information comprises flow information and stateless rules for applying policies. The access point is configured to use the classifica-tion information to perform stateless application classifica-tion and apply policies to packets received from a mobile device.")<br><br>Chandrasekaran at [0008] ("In another embodiment, an apparatus generally comprises a stateful classifier for performing stateful appli-cation classification at a controller, a classification database for storing classification information, and a processor for transmitting the classification information to an access point. The classification information comprises flow information and stateless rules for applying policies. The access point is configured to use the classification information to perform stateless application classification and apply policies to pack-ets received from a mobile device.")<br><br>Chandrasekaran at [0023] ("In one embodiment, the classification information 26 transmitted from the controller 12 to the AP 14 includes tuple information for a flow ( e.g., source IP address, destina-tion IP address, source port, destination port, and protocol), application identifier (ID), and stateless DPI information. Stateless DPI information includes classification and sub-classification information ( e.g., fixed or variable offset with a pattern or regular expression) and rules for applying policies on the sub-classified packets. The policies may include, for example, drop packet, mark a DSCP (Differentiated Services Code Point) value in the packet, or rate limit the traffic.")<br><br>Chandrasekaran at [0033] ("The following describes an example of the above process for WebEx traffic that has different sub-classifications for voice and video traffic. Stateful classification is first performed by the controller 12 at the beginning of the flow. The controller 12 may need to process, for example, 10, 100, or any other number of packets to classify the flow as Web Ex traffic. Once the classification is performed, the controller 12 |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 398 of 1100

| No. | '111 Patent Claim 3 | Chandrasekaran |
|---|---|---|
| | | sends the stateless DPI rules and flow information to the AP 14 for stateless sub-classification to distinguish voice, video, or data within a WebEx flow. For example, after the controller 12 identifies the WebEx meeting traffic, it pushes the tuple, the stateless DPI rules (as shown below), and policies to the AP 14 for upstream traffic marking, dropping, or rate-limit-ing. If the client 16 roams, the controller 12 transmits the same classification information to the new AP to which the client has roamed.")<br><br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Chandrasekaran in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 3(b) of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references.<br><br>For example, Chua discloses a mirror program in response to an indication based on the packet header, in which the network devices mirror copies of the packets of the packet flow to a second service device while forwarding the packets of the packet flow to the destination of the packet flow.<br><br>Chua at 7:28-54 ("SDN controller 112 may receive data as input from service devices 116. For example, SDN controller 112 may be con-figured to receive data from an intrusion detection system (IDS) device, a Denial of Service (DoS) device, a Distributed Denial of Service (DDoS) device, an intrusion prevention system (IPS) device, or the like. Based on this information, SDN controller 112 may make network enforcement decisions for specific traffic flows. That is, SDN controller 112 may program network devices of SDN 106 to perform pro-grammed actions on packets of a packet flow based on this data. Such programmed actions may include:<br><br>Allow-explicitly allow a certain network flow to proceed to its destination<br>Block-explicitly block a certain flow from traversing SDN 106<br>Mirror-allow the traffic, but send a copy of the traffic for deeper inspection or recording to, e.g., one of service devices 116<br>Redirect-redirect the traffic to another network (such as a honeypot device or other device of service devices 116) for either inspection or to keep a potential hacker 'busy' to determine if there is a real security threat. |


~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 399 of 1100

| No. | '111 Patent Claim 3 | Chandrasekaran |
|---|---|---|
| | | Transform-modify or translate values of headers of packets in the network flow Encapsulate-encapsulate packets in the network flow with a particular header")<br><br>Chua at 16:23-44 ("More particularly, control unit 130 may configure any of service devices 116 to send data representative of a particular event to SDN controller 112, and control unit 130 may auto-matically reprogram one or more network devices of SDN 106 in response to such data. For example, security monitor-ing applications of service devices 116 may determine that a specific source port, destination port, source IP address, des-tination IP address, or the like should be acted upon. Alter-natively, security monitoring applications may determine that, due to content or deep packet inspection, a specific type of traffic is malicious and should be blocked. In either case, the corresponding one of service devices 116 may send a message to SDN controller 112 representative of these deter-minations. As yet another example, a network performance device may monitor various performance metrics, such as latency, jitter, packet loss, or the like, and provide feedback data to SDN controller 112 based on these metrics. SDN controller 112 may respond by programming network devices of SDN 106 to perform a programmed action, such as allowing corresponding traffic, blocking corresponding traf-fic, mirroring corresponding traffic, redirecting correspond-ing traffic.")<br><br>Chua at 28:7-32 ("In addition, SDN controller 112 may configure the service device to send service-related data to one or more network devices (334). The service-related data may cause the net-work devices to change a path along which the packet is forwarded. For example, when the service device is a security device (e.g., a firewall or an IDS), if the security device determines that one or more packets of a packet flow are malicious, the security device may send service data indicat-ing that the packet flow includes malicious data. SDN con-troller 112 may program the network devices of the SDN to perform a programmed action based on the service-related data (336). For example, SDN controller 112 may program network devices to, in response to an indication that packets of a packet flow include malicious data, forward packets of the packet flow to a destination of the packet flow, forward packets of malicious packet flows to a collection device for further analysis, cause network devices to drop packets of the malicious packet flows, send a close session message to devices from which packets of the malicious packet flows were received, block the packets of the packet flow, mirror copies of the packets of the packet flow to a second service device while forwarding the packets of the packet flow to the destination of |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 400 of 1100

| No. | '111 Patent Claim 3 | Chandrasekaran |
|---|---|---|
| | | the packet flow, redirect the packets of the packet flow to a third service device, transform one or more values of headers of the packets, and/or encapsulate the pack-ets with a particular header, or other such actions.")<br><br>As another example, Swenson discloses a counting mode instructed by the network controller to the steering device for monitoring and optimizing, in which the steering device forwards the packet flow to the user device/origin server and at the same time, sending the packet flow to the network controller.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0028] ("In contrast to conventional inline TCP throughput monitoring devices that monitor every single data packets transmitted and received, the network controller 140 is an "out-of-band" computer server that interfaces with the steer-ing device 130 to selectively inspect user flows of interest. The network controller 140 may further identify user flows (e.g., among the flows of interest) for optimization. In one embodiment, the network controller 140 may be imple-mented at the steering device 130 to monitor traffic. In other embodiments, the network controller 140 is coupled to and communicates with the steering device 130 for traffic moni-toring and optimization. When queried by the steering device 130, the network controller 140 determines if a given network flow should be ignored, monitored further or optimized. Opti-mization of a flow is often decided at the beginning of the flow because it is rarely possible to switch to optimized content mid-stream once non-optimized content delivery has begun. However, the network controller 140 may determine that existing flows associated with a particular subscriber or other entity should be optimized. In turn, new flows ( e.g., resulting from seek requests in media, new media |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 401 of 1100

| No. | '111 Patent Claim 3 | Chandrasekaran |
|---|---|---|
| | | requests, resume after pause, etc.) determined to be associated with the entity may be optimized. The network controller 140 uses the net-work state as well as historical traffic data in its decision for monitoring and optimization. Knowledge on the current net-work state, such as congestion, deems critical when it comes to data optimization.")<br><br>Swenson at [0059] ("In one embodiment, as the steering device 130 monitors network responses, it is looking for flows that match one or more signatures for video and images. When a match-ing flow is detected, the steering device 130 forwards the HTTP request and a portion of the HTTP response to the network controller 140 over the ICAP client interface 404. After receiving the request and the portion of response at the ICAP server interface 406, the flow analyzer 312 of the net-work controller 140 performs a deep flow inspection to deter-mine if the flow is worth bandwidth monitoring and/or user detection. For example, the flow inspection performed by the flow analyzer 312 may determine if the flow indeed contains large or medium object ( e.g., larger than 50 kB), and/or if the source IP address of the flow is from a user or a group of users that are required to be monitored by policies. The flow ana-lyzer 312 may also determine if the flow needs to be opti-mized based on historical flow statistical data.")<br><br>Swenson at [0064] ("Similar to the "continue" mode, after receiving the initial HTTP messages of a flow and determining to monitor the flow, the network controller 140 notify the steering device 130 to work in a "counting" mode for bandwidth monitoring. In contrast to the "continue" mode, when a matching flow is detected for "counting" mode, the steering device 130 for-wards the HTTP response directly to the user device 110. While at the same time, the steering device 130 send a cus-tomized ICAP message to the network controller 140 over the network link 425. In one embodiment, the customized ICAP message contains the HTTP request and response headers, as well as a count of payload size of the current flow. After updating the flow statistics, the network controller 140 may acknowledge the gateway over the network line 426. In the "counting" mode, the network controller 140 does not join the network response path as an inline network element, but simply listens to the counting of flow size. The benefit of the "counting" mode is to off-load the network controller 140 from ingesting and forwarding the network flow on the net-work response path, while still enabling the detection of con-gestions and estimation of bandwidth associated with the flows of interest.") |


~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 402 of 1100

| No. | '111 Patent Claim 3 | Chandrasekaran |
|---|---|---|
| | | Swenson at [0071] ("After receiving the request, the video optimizer 150 forwards the video HTTP GET requests 622 to the origin server 160 and in return, receives a video file 624 from the origin server 160. The video optimizer 150 transcodes the video file to a format usable by the client device 110 based on network bandwidth available to the user device 110. The optimized video 626 is then transmitted from the video opti-mizer 150 to the steering device 130. In one embodiment, the steering device 130 intercepts the optimized video 626. The steering device 130 will then send an ICAP request to the network controller 140 for inspection. The network controller 140 deems this flow to be monitored and sends ICAP response 630. The steering device 130 then allows the flow to go through to the user device 110. The steering device 130 next sends periodic ICAP "counting" updates 632 to the network controller 140 until the flow completes. As such, the client receives the optimized video 626 for substantially real-time playback on an application executing on the user device 110.")<br>Swenson at [0072] ("In one embodiment, if the video optimizer 150 failed to retrieve user requested video file from the origin server 160, the video optimizer 150 appends a "do not transcode" flag to the HTTP redirect request and returned to the user device 110, which re-sends the request out over the network to the origin server 160. The origin server 160 responds appropriately to the request by sending back video 624, which is intercepted by the steering device 130 only. The steering device 130 forwards the video to the user device 110 and at the same time reports the flow size to the network controller 140 for monitoring purpose.") |

| No. | '111 Patent Claim 4 | Chandrasekaran |
|---|---|---|
| 4[a] | The method according to claim 1, wherein the instruction is 'probe', 'mirror', or 'terminate' instruction, and | Chandrasekaran discloses the method according to claim 1, wherein the instruction is 'probe', 'mirror', or 'terminate' instruction.<br><br>*See supra* at 2(a). |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 403 of 1100

| No. | '111 Patent Claim 4 | Chandrasekaran |
|---|---|---|
| 4[b] | upon receiving by the network node the 'probe' instruction and responsive to the packet satisfying the criterion, the method further comprising: sending the packet, by the network node, to the controller; | Chandrasekaran discloses upon receiving by the network node the 'probe' instruction and responsive to the packet satisfying the criterion, the method further comprising: sending the packet, by the network node, to the controller.<br><br>For example, Chandrasekaran discloses classification policies that involve sending the packet to the controller from the access point, in response to a classification policy.<br><br>Chandrasekaran at [0016] ("The wireless controller 12 includes a stateful appli-cation classifier 18 and the AP 14 includes a stateless appli-cation classifier 22. After the stateful classifier 18 identifies the application, the controller 12 transmits ( e.g., pushes) clas-sification information 26 to the AP 14 so that the AP can perform stateless classification and apply policies ( e.g., QoS or other policies) to traffic received from the mobile device 16. The controller 12 may also provide the classification information 26 to another AP 14 if the client 16 roams to a new AP, as shown in FIG. 1. Implementation of the stateful classifier 18 at the controller 12 and stateless classifier 22 at the AP 14 allows for policies to be applied for downstream traffic (packet 25) at the wireless controller 12, and for upstream traffic (packet 28) at the access point 14.")<br><br>Chandrasekaran at [0031] ("FIG. 3 is a flowchart illustrating an example of a process at the controller 12 for classification of traffic for application aware policies in a wireless network, in accor-dance with one embodiment. At step 40, the controller 12 receives packets belonging to a network flow. The controller 12 performs stateful classification to identify an application associated with the flow ( step 42). The controller 12 transmits classification information ( e.g., flow information, stateless DPI rule, and policy) to the AP 14 for use in stateless classi-fication at the AP (step 44). The controller 12 applies policies to downstream traffic (received at the controller and destined for the client 16) (step 46) and receives upstream traffic for which policies have been applied at the AP 14 (step 48). If the controller 12 determines ( e.g., receives an indication) that the client 16 has roamed, it transmits the classification informa-tion to the new AP 14 to which the client has roamed (steps 50 and 52).")<br><br>Chandrasekaran at [0033] ("The following describes an example of the above process for WebEx traffic that has different sub-classifications for voice and video traffic. Stateful classification is first performed by the controller 12 at the beginning of the flow. The |

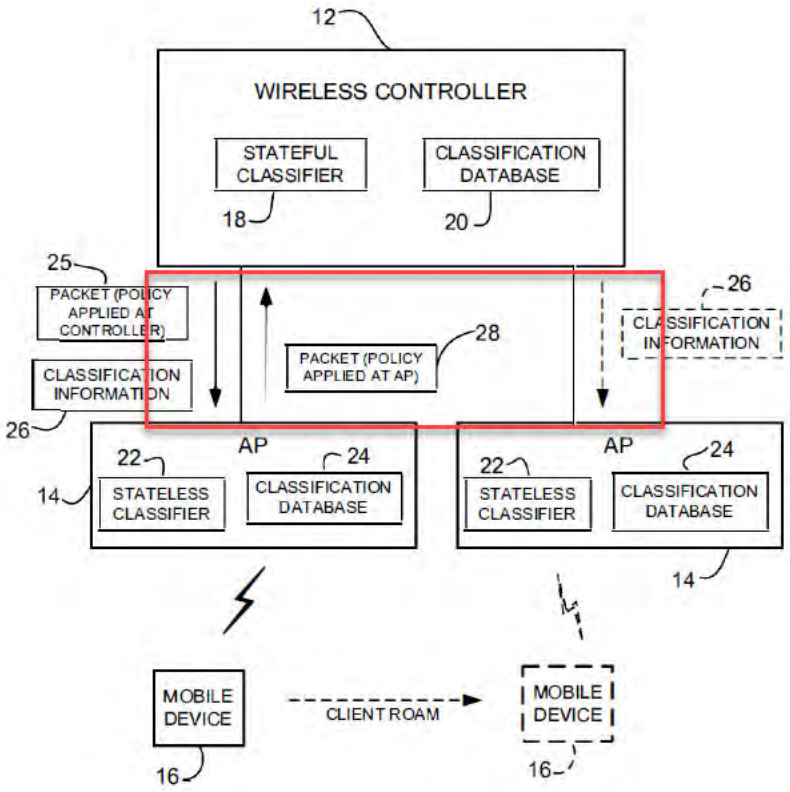


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 404 of 1100

| No. | '111 Patent Claim 4 | Chandrasekaran |
|---|---|---|
| | | controller 12 may need to process, for example, 10, 100, or any other number of packets to classify the flow as Web Ex traffic. Once the classification is performed, the controller 12 sends the stateless DPI rules and flow information to the AP 14 for stateless sub-classification to distinguish voice, video, or data within a WebEx flow. For example, after the controller 12 identifies the WebEx meeting traffic, it pushes the tuple, the stateless DPI rules (as shown below), and policies to the AP 14 for upstream traffic marking, dropping, or rate-limit-ing. If the client 16 roams, the controller 12 transmits the same classification information to the new AP to which the client has roamed.")<br><br>Chandrasekaran at Figure 1 (annotations added) |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 405 of 1100

| No. | '111 Patent Claim 4 | Chandrasekaran |
|---|---|---|
| | |  <br>FIGURE 1 |
| 4[c] | responsive to receiving the packet, analyzing the packet, by the controller; | Chandrasekaran discloses responsive to receiving the packet, analyzing the packet, by the controller.<br><br>For example, Chandrasekaran discloses a controller performing stateful application classification, in response to receiving the packet. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 406 of 1100

| No. | '111 Patent Claim 4 | Chandrasekaran |
|---|---|---|
| | | Chandrasekaran at [0014] ("The term 'wireless controller' or 'controller' as used herein may refer to a wireless LAN (local area network controller, mobility controller, wireless control device, wire-less control system, or any other network device operable to perform control functions for a wireless network. The net-work site may also include a wireless control system or other platform for centralized wireless LAN planning, configura-tion, and management. The wireless controller 12 enables system wide functions for wireless applications and may support any number of access points 14. Each access point 14 may serve any number of mobile devices 16 in the wireless network. The wireless controller 12 may be, for example, a standalone device or a rack-mounted appliance. In the example shown in FIG. 1, the wireless controller 12 and access points 14 are separate devices and may be located remote from one another. The wireless controller 12 may also be integrated with the access point 14 ( e.g., autonomous AP) or located at a switch, router, switch/router, or other network device. Thus, the wireless controller 12 may be a physical device located at a standalone device, access point, switch, router, or other network device. The wireless controller 12 may also be a virtual device located in a network or cloud, for example.")<br><br>Chandrasekaran at [0016] ("The wireless controller 12 includes a stateful appli-cation classifier 18 and the AP 14 includes a stateless appli-cation classifier 22. After the stateful classifier 18 identifies the application, the controller 12 transmits ( e.g., pushes) clas-sification information 26 to the AP 14 so that the AP can perform stateless classification and apply policies ( e.g., QoS or other policies) to traffic received from the mobile device 16. The controller 12 may also provide the classification information 26 to another AP 14 if the client 16 roams to a new AP, as shown in FIG. 1. Implementation of the stateful classifier 18 at the controller 12 and stateless classifier 22 at the AP 14 allows for policies to be applied for downstream traffic (packet 25) at the wireless controller 12, and for upstream traffic (packet 28) at the access point 14.")<br><br>Chandrasekaran at [0020] ("In one embodiment, the stateful classifier 18 is a classification engine configured for NBAR (Network Based Application Recognition) or other technology used to classify applications. The classifier 18 is operable to recognize a wide variety of applications, including Web-based and client/ server applications. The applications may include, for example, Skype, YouTube, Netflix, WebEx, Google Voice, BitTorrent, Citrix, virtual desktop, PCoIP, or any other appli-cation. The classification engine may be |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 407 of 1100

| No. | '111 Patent Claim 4 | Chandrasekaran |
|---|---|---|
| | | configured, for example, to identify generic protocols and perform heuristic analysis for encrypted protocols. The classifiers 18, 22 are configured to perform deep packet inspection (DPI), which provides the ability to look into the packet past basic header information so that the contents of a particular packet can be determined.")<br><br>Chandrasekaran at [0031] ("FIG. 3 is a flowchart illustrating an example of a process at the controller 12 for classification of traffic for application aware policies in a wireless network, in accor-dance with one embodiment. At step 40, the controller 12 receives packets belonging to a network flow. The controller 12 performs stateful classification to identify an application associated with the flow ( step 42). The controller 12 transmits classification information ( e.g., flow information, stateless DPI rule, and policy) to the AP 14 for use in stateless classi-fication at the AP (step 44). The controller 12 applies policies to downstream traffic (received at the controller and destined for the client 16) (step 46) and receives upstream traffic for which policies have been applied at the AP 14 (step 48). If the controller 12 determines ( e.g., receives an indication) that the client 16 has roamed, it transmits the classification informa-tion to the new AP 14 to which the client has roamed (steps 50 and 52).")<br><br>Chandrasekaran at Figure 3 (annotations added) |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 408 of 1100

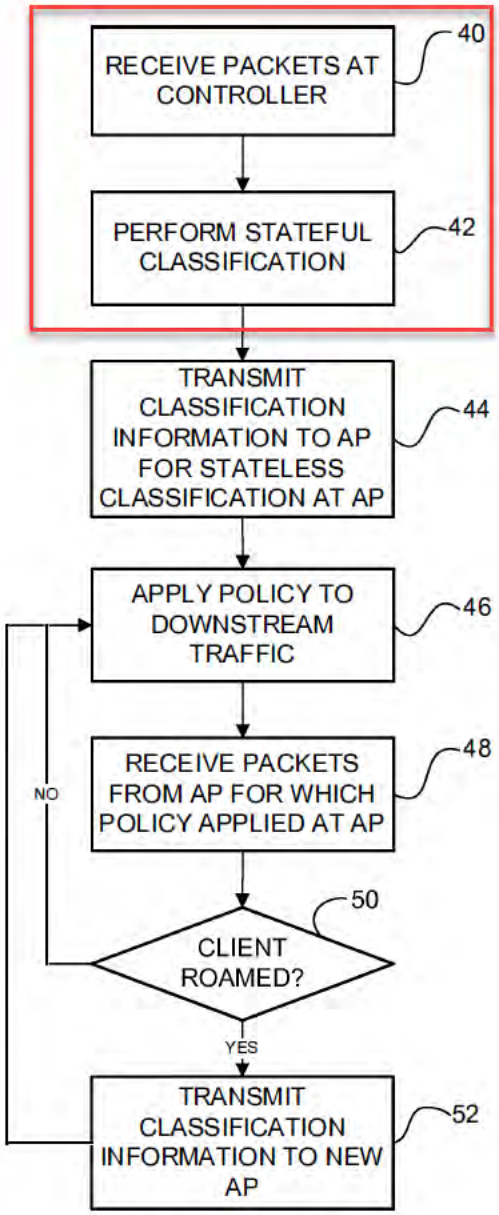



FIGURE 3

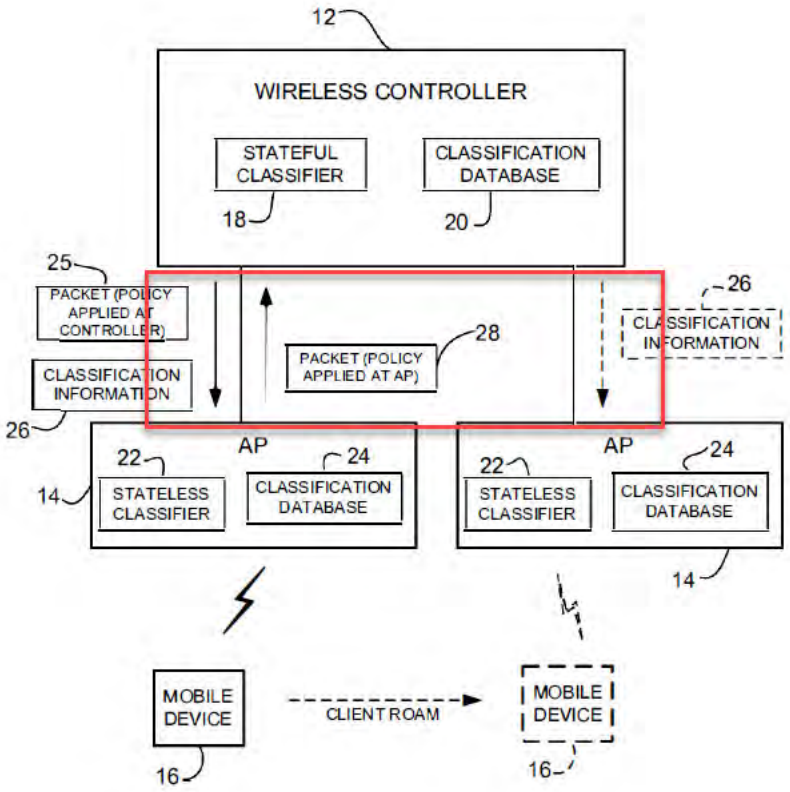


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 409 of 1100

| No. | '111 Patent Claim 4 | Chandrasekaran |
|---|---|---|
| 4[d] | sending the packet, by the controller, to the network node; and | Chandrasekaran discloses sending the packet, by the controller, to the network node.<br><br>For example, Chandrasekaran discloses transmitting the packet and information from the controller to the access point.<br><br>Chandrasekaran at [0031] ("FIG. 3 is a flowchart illustrating an example of a process at the controller 12 for classification of traffic for application aware policies in a wireless network, in accor-dance with one embodiment. At step 40, the controller 12 receives packets belonging to a network flow. The controller 12 performs stateful classification to identify an application associated with the flow ( step 42). The controller 12 transmits classification information ( e.g., flow information, stateless DPI rule, and policy) to the AP 14 for use in stateless classi-fication at the AP (step 44). The controller 12 applies policies to downstream traffic (received at the controller and destined for the client 16) (step 46) and receives upstream traffic for which policies have been applied at theAP 14 (step 48). If the controller 12 determines ( e.g., receives an indication) that the client 16 has roamed, it transmits the classification informa-tion to the new AP 14 to which the client has roamed (steps 50 and 52).")<br><br>Chandrasekaran at [0033] ("The following describes an example of the above process for WebEx traffic that has different sub-classifications for voice and video traffic. Stateful classification is first performed by the controller 12 at the beginning of the flow. The controller 12 may need to process, for example, 10, 100, or any other number of packets to classify the flow as Web Ex traffic. Once the classification is performed, the controller 12 sends the stateless DPI rules and flow information to the AP 14 for stateless sub-classification to distinguish voice, video, or data within a WebEx flow. For example, after the controller 12 identifies the WebEx meeting traffic, it pushes the tuple, the stateless DPI rules (as shown below), and policies to the AP 14 for upstream traffic marking, dropping, or rate-limit-ing. If the client 16 roams, the controller 12 transmits the same classification information to the new AP to which the client has roamed.")<br><br>Chandrasekaran at Figure 1 (annotations added) |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 410 of 1100

| No. | '111 Patent Claim 4 | Chandrasekaran |
|---|---|---|
| | |  <br>FIGURE 1 |
| 4[e] | responsive to receiving the packet, sending the packet, by the network node, to the second entity. | Chandrasekaran discloses responsive to receiving the packet, sending the packet, by the network node, to the second entity.<br><br>For example, Chandrasekaran discloses sending the packet by the access point to the endpoint, after receiving the packet. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 411 of 1100

| No. | '111 Patent Claim 4 | Chandrasekaran |
|---|---|---|
| | | Chandrasekaran at [0012] ("Referring now to the drawings, and first to FIG.1, an example of a network in which embodiments described herein may be implemented is shown. For simplification, only a small number of network devices are shown. The network includes a wireless controller 12 in communication with a mobile device (client, wireless device, endpoint) 16 through an access point (AP) 14. In the example shown in FIG. 1, the controller 12 is in wired communication with two access points 14 for wireless communication with any number of mobile devices 16 via a wireless network ( e.g., WLAN (wire-less local area network)) at a network site. The wireless con-troller 12 may be in communication with one or more other networks (not shown) (e.g., Internet, intranet, local area net-work, wireless local area network, cellular network, metro-politan area network, wide area network, satellite network, radio access network, public switched network, virtual pri-vate network, or any other network or combination thereof). Communication paths between the wireless controller 12 and other networks or between the controller and access points 14 may include any number or type of intermediate nodes (e.g., routers, switches, gateways, or other network devices), which facilitate passage of data between network devices.")<br><br>Chandrasekaran at [0013] ("In one example, the wireless controller 12 receives upstream traffic transmitted from the mobile device 16 and destined for another endpoint ( e.g., host, user device), and transmits downstream traffic received from the endpoint to the mobile device in a communication session. As used herein, the term 'downstream' refers to traffic transmitted from the controller 12 towards the mobile device 16, and the term 'upstream' refers to traffic transmitted from the mobile device towards the controller.")<br><br>Chandrasekaran at [0014] ("The term 'wireless controller' or 'controller' as used herein may refer to a wireless LAN (local area network) controller, mobility controller, wireless control device, wire-less control system, or any other network device operable to perform control functions for a wireless network. The net-work site may also include a wireless control system or other platform for centralized wireless LAN planning, configura-tion, and management. The wireless controller 12 enables system wide functions for wireless applications and may support any number of access points 14. Each access point 14 may serve any number of mobile devices 16 in the wireless network. The wireless controller 12 may be, for example, a standalone device or a rack-mounted appliance. In the example, |

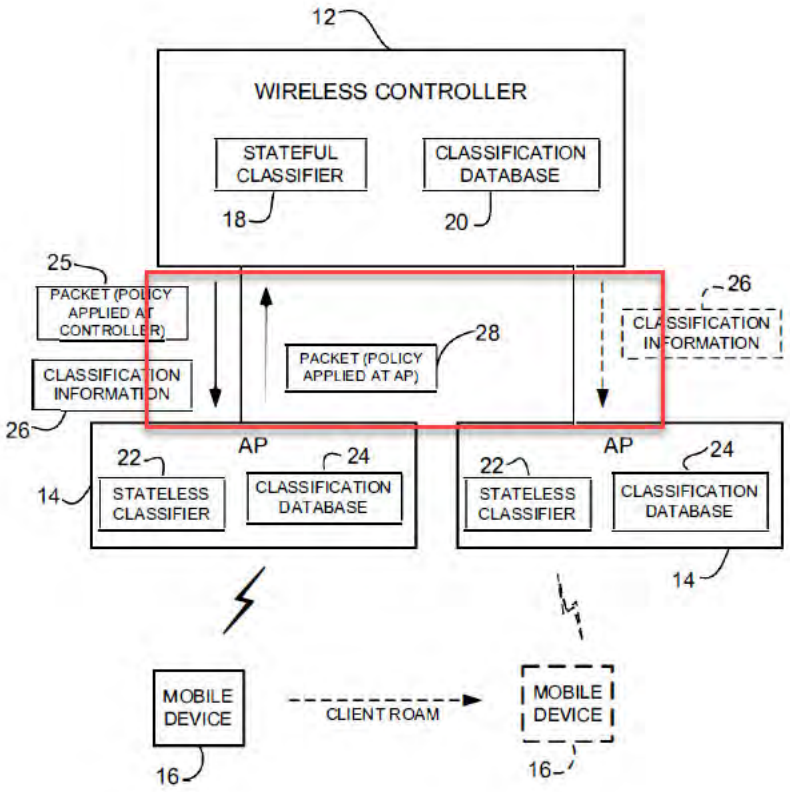


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 412 of 1100

| No. | '111 Patent Claim 4 | Chandrasekaran |
|---|---|---|
| | | shown in FIG. 1, the wireless controller 12 and access points 14 are separate devices and may be located remote from one another. The wireless controller 12 may also be integrated with the access point 14 ( e.g., autonomous AP) or located at a switch, router, switch/router, or other network device. Thus, the wireless controller 12 may be a physical device located at a standalone device, access point, switch, router, or other network device. The wireless controller 12 may also be a virtual device located in a network or cloud, for example.")<br><br>Chandrasekaran at [0015] ("The mobile device 16 may be any suitable equip-ment that supports wireless communication, including for example, a mobile phone, personal digital assistant, portable computing device, laptop, tablet, multimedia device, or any other wireless device. The mobile device 16 and access point 14 are configured to perform wireless communication according to a wireless network communication protocol such as IEEE 802.11/Wi-Fi.")<br><br>Chandrasekaran at [0016] ("The wireless controller 12 includes a stateful appli-cation classifier 18 and the AP 14 includes a stateless appli-cation classifier 22. After the stateful classifier 18 identifies the application, the controller 12 transmits ( e.g., pushes) clas-sification information 26 to the AP 14 so that the AP can perform stateless classification and apply policies ( e.g., QoS or other policies) to traffic received from the mobile device 16. The controller 12 may also provide the classification information 26 to another AP 14 if the client 16 roams to a new AP, as shown in FIG. 1. Implementation of the stateful classifier 18 at the controller 12 and stateless classifier 22 at the AP 14 allows for policies to be applied for downstream traffic (packet 25) at the wireless controller 12, and for upstream traffic (packet 28) at the access point 14.")<br><br>Chandrasekaran at Figure 1 (annotations added) |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 413 of 1100

| No. | '111 Patent Claim 4 | Chandrasekaran |
|---|---|---|
| | |  <br>FIGURE 1 |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 414 of 1100

| No. | '111 Patent Claim 5 | Chandrasekaran |
|---|---|---|
| 5 | The method according to claim 1, further comprising responsive to the packet satisfying the criterion and to the instruction, sending the packet or a portion thereof, by the network node, to the controller. | Chandrasekaran discloses the method according to claim 1, further comprising responsive to the packet satisfying the criterion and to the instruction, sending the packet or a portion thereof, by the network node, to the controller.<br><br>For example, Chandrasekaran discloses sending the packet or information about the packet to the controller by the access point, in response to performing the classification policy.<br><br>*See supra* at Claim 1.<br><br>Chandrasekaran at [0012] ("Referring now to the drawings, and first to FIG.1, an example of a network in which embodiments described herein may be implemented is shown. For simplification, only a small number of network devices are shown. The network includes a wireless controller 12 in communication with a mobile device (client, wireless device, endpoint) 16 through an access point (AP) 14. In the example shown in FIG. 1, the controller 12 is in wired communication with two access points 14 for wireless communication with any number of mobile devices 16 via a wireless network ( e.g., WLAN (wire-less local area network)) at a network site. The wireless con-troller 12 may be in communication with one or more other networks (not shown) (e.g., Internet, intranet, local area net-work, wireless local area network, cellular network, metro-politan area network, wide area network, satellite network, radio access network, public switched network, virtual pri-vate network, or any other network or combination thereof). Communication paths between the wireless controller 12 and other networks or between the controller and access points 14 may include any number or type of intermediate nodes (e.g., routers, switches, gateways, or other network devices), which facilitate passage of data between network devices.")<br><br>Chandrasekaran at [0013] ("In one example, the wireless controller 12 receives upstream traffic transmitted from the mobile device 16 and destined for another endpoint ( e.g., host, user device), and transmits downstream traffic received from the endpoint to the mobile device in a communication session. As used herein, the term 'downstream' refers to traffic transmitted from the controller 12 towards the mobile device 16, and the term 'upstream' refers to traffic transmitted from the mobile device towards the controller.") |

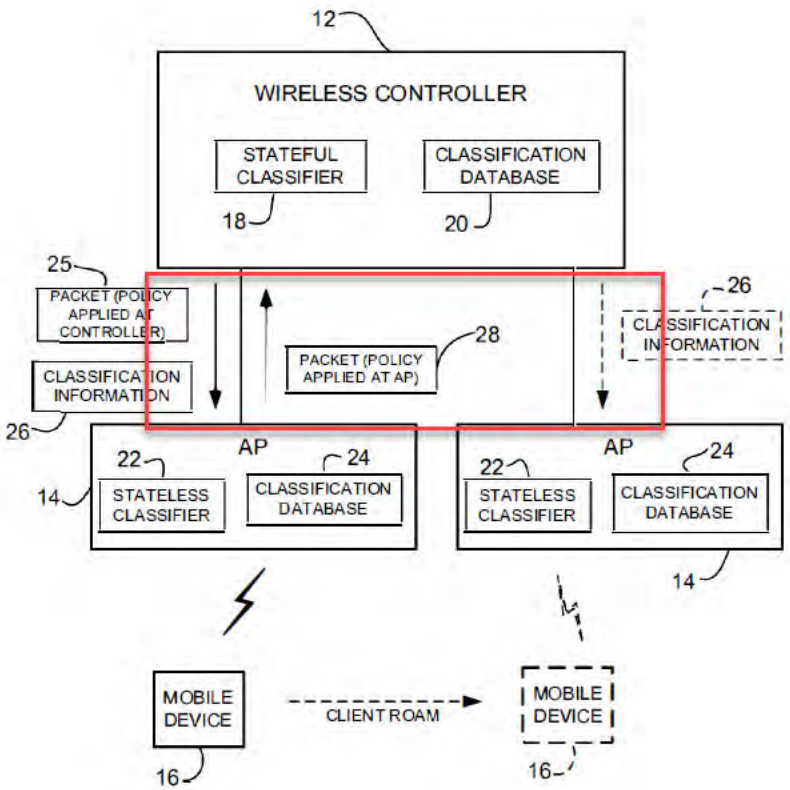


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 415 of 1100

| No. | '111 Patent Claim 5 | Chandrasekaran |
|---|---|---|
| | | Chandrasekaran at [0014] ("The term 'wireless controller' or 'controller' as used herein may refer to a wireless LAN (local area network) controller, mobility controller, wireless control device, wire-less control system, or any other network device operable to perform control functions for a wireless network. The net-work site may also include a wireless control system or other platform for centralized wireless LAN planning, configura-tion, and management. The wireless controller 12 enables system wide functions for wireless applications and may support any number of access points 14. Each access point 14 may serve any number of mobile devices 16 in the wireless network. The wireless controller 12 may be, for example, a standalone device or a rack-mounted appliance. In the example shown in FIG. 1, the wireless controller 12 and access points 14 are separate devices and may be located remote from one another. The wireless controller 12 may also be integrated with the access point 14 ( e.g., autonomous AP) or located at a switch, router, switch/router, or other network device. Thus, the wireless controller 12 may be a physical device located at a standalone device, access point, switch, router, or other network device. The wireless controller 12 may also be a virtual device located in a network or cloud, for example.")<br><br>Chandrasekaran at [0015] ("The mobile device 16 may be any suitable equip-ment that supports wireless communication, including for example, a mobile phone, personal digital assistant, portable computing device, laptop, tablet, multimedia device, or any other wireless device. The mobile device 16 and access point 14 are configured to perform wireless communication according to a wireless network communication protocol such as IEEE 802.11/Wi-Fi.")<br><br>Chandrasekaran at [0016] ("The wireless controller 12 includes a stateful appli-cation classifier 18 and the AP 14 includes a stateless appli-cation classifier 22. After the stateful classifier 18 identifies the application, the controller 12 transmits ( e.g., pushes) clas-sification information 26 to the AP 14 so that the AP can perform stateless classification and apply policies ( e.g., QoS or other policies) to traffic received from the mobile device 16. The controller 12 may also provide the classification information 26 to another AP 14 if the client 16 roams to a new AP, as shown in FIG. 1. Implementation of the stateful classifier 18 at the controller 12 and stateless classifier 22 at the AP 14 allows for policies to be applied for downstream traffic (packet 25) at the wireless controller 12, and for upstream traffic (packet 28) at the access point 14.") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 416 of 1100

| No. | '111 Patent Claim 5 | Chandrasekaran |
|---|---|---|
| | | Chandrasekaran at Figure 1 (annotations added)<br>  <br><br>FIGURE 1 |

| No. | '111 Patent Claim 6 | Chandrasekaran |
|---|---|---|
| 6 | The method according to claim 5, further | Chandrasekaran discloses the method according to claim 5, further comprising storing the received packet or a portion thereof, by the controller, in a memory. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 417 of 1100

| No. | '111 Patent Claim 6 | Chandrasekaran |
|---|---|---|
| | comprising storing the received packet or a portion thereof, by the controller, in a memory. | For example, Chandrasekaran discloses storing the packet or information about the packet in the memory of the controller.<br><br>*See supra* at Claim 5.<br><br>Chandrasekaran at [0022] ("The wireless controller 12 and AP 14 further include classification databases 20, 24, respectively, for storing clas-sification information. The classification database 20 at the controller 12 stores classification information obtained by the stateful classifier 18. The classification database 24 at the AP 14 stores classification information 26 transmitted to the AP from the controller 12. The classification information stored at the databases 20, 24 may include, for example, flow infor-mation, stateless rules, and policies, as described below.")<br><br>Chandrasekaran at [0024] ("It is to be understood that the network shown in FIG. 1 and described herein is only an example and that other networks having different components or configurations may be used, without departing from the scope of the embodi-ments. For example, there may be any number of APs 14 in communication with the controller 12 for supporting any number of mobile devices 16. Also, as described above, the controller 12 may be located at various locations and devices in the network.")<br><br>Chandrasekaran at [0026] ("Memory 34 may be a volatile memory or non-vola-tile storage, which stores various applications, operating sys-tems, modules, and data for execution and use by the proces-sor 32. Memory 34 may include, for example, classification database 35. The classification database 35 may be any data structure configured for at least temporarily storing classifi-cation information including, for example, flow information, application ID, stateless DPI rules, and policies.") |

| No. | '111 Patent Claim 7 | Chandrasekaran |
|---|---|---|
| 7 | The method according to claim 5, further comprising responsive | Chandrasekaran discloses the method according to claim 5, further comprising responsive to the packet satisfying the criterion and to the instruction, sending a portion of the packet, by the network node, to the controller. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 418 of 1100

| No. | '111 Patent Claim 7 | Chandrasekaran |
|---|---|---|
| | to the packet satisfying the criterion and to the instruction, sending a portion of the packet, by the network node, to the controller. | For example, Chandrasekaran discloses sending information about the packet to the controller by the access point, in response to performing the classification policy.<br><br>*See supra* at Claim 5.<br><br>Chandrasekaran at [0016] ("The wireless controller 12 includes a stateful appli-cation classifier 18 and the AP 14 includes a stateless appli-cation classifier 22. After the stateful classifier 18 identifies the application, the controller 12 transmits ( e.g., pushes) clas-sification information 26 to the AP 14 so that the AP can perform stateless classification and apply policies ( e.g., QoS or other policies) to traffic received from the mobile device 16. The controller 12 may also provide the classification information 26 to another AP 14 if the client 16 roams to a new AP, as shown in FIG. 1. Implementation of the stateful classifier 18 at the controller 12 and stateless classifier 22 at the AP 14 allows for policies to be applied for downstream traffic (packet 25) at the wireless controller 12, and for upstream traffic (packet 28) at the access point 14.")<br><br>Chandrasekaran at [0031] ("FIG. 3 is a flowchart illustrating an example of a process at the controller 12 for classification of traffic for application aware policies in a wireless network, in accor-dance with one embodiment. At step 40, the controller 12 receives packets belonging to a network flow. The controller 12 performs stateful classification to identify an application associated with the flow ( step 42). The controller 12 transmits classification information ( e.g., flow information, stateless DPI rule, and policy) to the AP 14 for use in stateless classi-fication at the AP (step 44). The controller 12 applies policies to downstream traffic (received at the controller and destined for the client 16) (step 46) and receives upstream traffic for which policies have been applied at theAP 14 (step 48). If the controller 12 determines ( e.g., receives an indication) that the client 16 has roamed, it transmits the classification informa-tion to the new AP 14 to which the client has roamed (steps 50 and 52).")<br><br>Chandrasekaran at [0033] ("The following describes an example of the above process for WebEx traffic that has different sub-classifications for voice and video traffic. Stateful classification is first performed by the controller 12 at the beginning of the flow. The controller 12 may need to process, for example, 10, 100, or any other number of packets to |

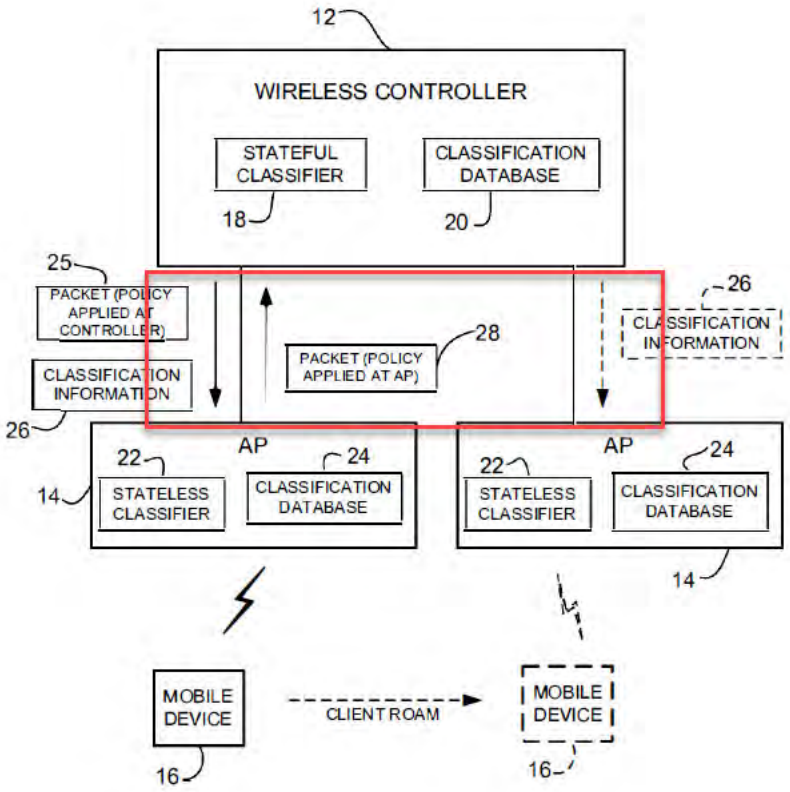


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 419 of 1100

| No. | '111 Patent Claim 7 | Chandrasekaran |
|---|---|---|
| | | classify the flow as Web Ex traffic. Once the classification is performed, the controller 12 sends the stateless DPI rules and flow information to the AP 14 for stateless sub-classification to distinguish voice, video, or data within a WebEx flow. For example, after the controller 12 identifies the WebEx meeting traffic, it pushes the tuple, the stateless DPI rules (as shown below), and policies to the AP 14 for upstream traffic marking, dropping, or rate-limit-ing. If the client 16 roams, the controller 12 transmits the same classification information to the new AP to which the client has roamed.")<br><br>Chandrasekaran at Figure 1 (annotations added) |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 420 of 1100

| No. | '111 Patent Claim 7 | Chandrasekaran |
|---|---|---|
| | |  <br>FIGURE 1 |

| No. | '111 Patent Claim 8 | Chandrasekaran |
|---|---|---|
| 8[a] | The method according to claim 7, wherein the portion of the packet | Chandrasekaran discloses the method according to claim 7, wherein the portion of the packet consists of multiple consecutive bytes. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 421 of 1100

| No. | '111 Patent Claim 8 | Chandrasekaran |
|---|---|---|
| | consists of multiple consecutive bytes, and | For example, Chandrasekaran discloses information about a packet consisting of multiple bytes.<br><br>*See supra* at Claim 7.<br><br>Chandrasekaran at [0014] ("The term 'wireless controller' or 'controller' as used herein may refer to a wireless LAN (local area network) controller, mobility controller, wireless control device, wire-less control system, or any other network device operable to perform control functions for a wireless network. The net-work site may also include a wireless control system or other platform for centralized wireless LAN planning, configura-tion, and management. The wireless controller 12 enables system wide functions for wireless applications and may support any number of access points 14. Each access point 14 may serve any number of mobile devices 16 in the wireless network. The wireless controller 12 may be, for example, a standalone device or a rack-mounted appliance. In the example shown in FIG. 1, the wireless controller 12 and access points 14 are separate devices and may be located remote from one another. The wireless controller 12 may also be integrated with the access point 14 ( e.g., autonomous AP) or located at a switch, router, switch/router, or other network device. Thus, the wireless controller 12 may be a physical device located at a standalone device, access point, switch, router, or other network device. The wireless controller 12 may also be a virtual device located in a network or cloud, for example.")<br><br>Chandrasekaran at [0016] ("The wireless controller 12 includes a stateful appli-cation classifier 18 and the AP 14 includes a stateless appli-cation classifier 22. After the stateful classifier 18 identifies the application, the controller 12 transmits ( e.g., pushes) clas-sification information 26 to the AP 14 so that the AP can perform stateless classification and apply policies ( e.g., QoS or other policies) to traffic received from the mobile device 16. The controller 12 may also provide the classification information 26 to another AP 14 if the client 16 roams to a new AP, as shown in FIG. 1. Implementation of the stateful classifier 18 at the controller 12 and stateless classifier 22 at the AP 14 allows for policies to be applied for downstream traffic (packet 25) at the wireless controller 12, and for upstream traffic (packet 28) at the access point 14.")<br><br>Chandrasekaran at [0020] ("In one embodiment, the stateful classifier 18 is a classification engine configured for NBAR (Network Based Application Recognition) or other technology |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 422 of 1100

| No. | '111 Patent Claim 8 | Chandrasekaran |
|---|---|---|
| | | used to classify applications. The classifier 18 is operable to recognize a wide variety of applications, including Web-based and client/ server applications. The applications may include, for example, Skype, YouTube, Netflix, WebEx, Google Voice, BitTorrent, Citrix, virtual desktop, PCoIP, or any other appli-cation. The classification engine may be configured, for example, to identify generic protocols and perform heuristic analysis for encrypted protocols. The classifiers 18, 22 are configured to perform deep packet inspection (DPI), which provides the ability to look into the packet past basic header information so that the contents of a particular packet can be determined.")<br><br>Chandrasekaran at [0031] ("FIG. 3 is a flowchart illustrating an example of a process at the controller 12 for classification of traffic for application aware policies in a wireless network, in accor-dance with one embodiment. At step 40, the controller 12 receives packets belonging to a network flow. The controller 12 performs stateful classification to identify an application associated with the flow ( step 42). The controller 12 transmits classification information ( e.g., flow information, stateless DPI rule, and policy) to the AP 14 for use in stateless classi-fication at the AP (step 44). The controller 12 applies policies to downstream traffic (received at the controller and destined for the client 16) (step 46) and receives upstream traffic for which policies have been applied at the AP 14 (step 48). If the controller 12 determines ( e.g., receives an indication) that the client 16 has roamed, it transmits the classification informa-tion to the new AP 14 to which the client has roamed (steps 50 and 52).")<br><br>Chandrasekaran at [0035]-[0044] ("WebEx Video:<br>UDP Payload<br>First byte=0x06<br>Bytes [6-9]=Data length<br>10th byte=0x50<br><br>WebEx Voice:<br>UDP Payload<br>First byte=0x06<br>Bytes [6-9]=Data length<br>10th byte=0x48") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 423 of 1100

| No. | '111 Patent Claim 8 | Chandrasekaran |
|---|---|---|
| 8[b] | wherein the instruction comprises identification of the consecutive bytes in the packet. | Chandrasekaran discloses wherein the instruction comprises identification of the consecutive bytes in the packet.<br><br>For example, Chandrasekaran discloses the classification policy identifying the bytes in the packet.<br><br>Chandrasekaran at [0014] ("The term 'wireless controller' or 'controller' as used herein may refer to a wireless LAN (local area network) controller, mobility controller, wireless control device, wire-less control system, or any other network device operable to perform control functions for a wireless network. The net-work site may also include a wireless control system or other platform for centralized wireless LAN planning, configura-tion, and management. The wireless controller 12 enables system wide functions for wireless applications and may support any number of access points 14. Each access point 14 may serve any number of mobile devices 16 in the wireless network. The wireless controller 12 may be, for example, a standalone device or a rack-mounted appliance. In the example shown in FIG. 1, the wireless controller 12 and access points 14 are separate devices and may be located remote from one another. The wireless controller 12 may also be integrated with the access point 14 ( e.g., autonomous AP) or located at a switch, router, switch/router, or other network device. Thus, the wireless controller 12 may be a physical device located at a standalone device, access point, switch, router, or other network device. The wireless controller 12 may also be a virtual device located in a network or cloud, for example.")<br><br>Chandrasekaran at [0016] ("The wireless controller 12 includes a stateful appli-cation classifier 18 and the AP 14 includes a stateless appli-cation classifier 22. After the stateful classifier 18 identifies the application, the controller 12 transmits ( e.g., pushes) clas-sification information 26 to the AP 14 so that the AP can perform stateless classification and apply policies ( e.g., QoS or other policies) to traffic received from the mobile device 16. The controller 12 may also provide the classification information 26 to another AP 14 if the client 16 roams to a new AP, as shown in FIG. 1. Implementation of the stateful classifier 18 at the controller 12 and stateless classifier 22 at the AP 14 allows for policies to be applied for downstream traffic (packet 25) at the wireless controller 12, and for upstream traffic (packet 28) at the access point 14.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 424 of 1100

| No. | '111 Patent Claim 8 | Chandrasekaran |
|---|---|---|
| | | Chandrasekaran at [0020] ("In one embodiment, the stateful classifier 18 is a classification engine configured for NBAR (Network Based Application Recognition) or other technology used to classify applications. The classifier 18 is operable to recognize a wide variety of applications, including Web-based and client/ server applications. The applications may include, for example, Skype, YouTube, Netflix, WebEx, Google Voice, BitTorrent, Citrix, virtual desktop, PCoIP, or any other appli-cation. The classification engine may be configured, for example, to identify generic protocols and perform heuristic analysis for encrypted protocols. The classifiers 18, 22 are configured to perform deep packet inspection (DPI), which provides the ability to look into the packet past basic header information so that the contents of a particular packet can be determined.")<br><br>Chandrasekaran at [0031] ("FIG. 3 is a flowchart illustrating an example of a process at the controller 12 for classification of traffic for application aware policies in a wireless network, in accor-dance with one embodiment. At step 40, the controller 12 receives packets belonging to a network flow. The controller 12 performs stateful classification to identify an application associated with the flow ( step 42). The controller 12 transmits classification information ( e.g., flow information, stateless DPI rule, and policy) to the AP 14 for use in stateless classi-fication at the AP (step 44). The controller 12 applies policies to downstream traffic (received at the controller and destined for the client 16) (step 46) and receives upstream traffic for which policies have been applied at the AP 14 (step 48). If the controller 12 determines ( e.g., receives an indication) that the client 16 has roamed, it transmits the classification informa-tion to the new AP 14 to which the client has roamed (steps 50 and 52).")<br><br>Chandrasekaran at [0035]-[0044] ("WebEx Video:<br>UDP Payload<br>First byte=0x06<br>Bytes [6-9]=Data length<br>10th byte=0x50<br><br>WebEx Voice:<br>UDP Payload<br>First byte=0x06<br>Bytes [6-9]=Data length |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 425 of 1100

| No. | '111 Patent Claim 8 | Chandrasekaran |
|---|---|---|
| | | 10th byte=0x48") |

| No. | '111 Patent Claim 9 | Chandrasekaran |
|---|---|---|
| 9 | The method according to claim 5, further comprising responsive to receiving the packet, analyzing the packet, by the controller. | Chandrasekaran discloses the method according to claim 5, further comprising responsive to receiving the packet, analyzing the packet, by the controller.<br><br>For example, Chandrasekaran discloses analyzing the packet through stateful classification by the controller.<br><br>*See supra* at Claim 5.<br><br>Chandrasekaran at [0014] ("The term 'wireless controller' or 'controller' as used herein may refer to a wireless LAN (local area network) controller, mobility controller, wireless control device, wire-less control system, or any other network device operable to perform control functions for a wireless network. The net-work site may also include a wireless control system or other platform for centralized wireless LAN planning, configura-tion, and management. The wireless controller 12 enables system wide functions for wireless applications and may support any number of access points 14. Each access point 14 may serve any number of mobile devices 16 in the wireless network. The wireless controller 12 may be, for example, a standalone device or a rack-mounted appliance. In the example shown in FIG. 1, the wireless controller 12 and access points 14 are separate devices and may be located remote from one another. The wireless controller 12 may also be integrated with the access point 14 ( e.g., autonomous AP) or located at a switch, router, switch/router, or other network device. Thus, the wireless controller 12 may be a physical device located at a standalone device, access point, switch, router, or other network device. The wireless controller 12 may also be a virtual device located in a network or cloud, for example.")<br><br>Chandrasekaran at [0016] ("The wireless controller 12 includes a stateful appli-cation classifier 18 and the AP 14 includes a stateless appli-cation classifier 22. After the stateful classifier 18 identifies the application, the controller 12 transmits ( e.g., pushes) clas-sification information 26 to the AP 14 so that the AP can perform stateless classification and apply policies ( e.g., QoS or other policies) to traffic received from the mobile device 16. The controller 12 may also provide the classification information 26 to |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 426 of 1100

| No. | '111 Patent Claim 9 | Chandrasekaran |
|---|---|---|
| | | another AP 14 if the client 16 roams to a new AP, as shown in FIG. 1. Implementation of the stateful classifier 18 at the controller 12 and stateless classifier 22 at the AP 14 allows for policies to be applied for downstream traffic (packet 25) at the wireless controller 12, and for upstream traffic (packet 28) at the access point 14.")<br><br>Chandrasekaran at [0020] ("In one embodiment, the stateful classifier 18 is a classification engine configured for NBAR (Network Based Application Recognition) or other technology used to classify applications. The classifier 18 is operable to recognize a wide variety of applications, including Web-based and client/ server applications. The applications may include, for example, Skype, YouTube, Netflix, WebEx, Google Voice, BitTorrent, Citrix, virtual desktop, PCoIP, or any other appli-cation. The classification engine may be configured, for example, to identify generic protocols and perform heuristic analysis for encrypted protocols. The classifiers 18, 22 are configured to perform deep packet inspection (DPI), which provides the ability to look into the packet past basic header information so that the contents of a particular packet can be determined.")<br><br>Chandrasekaran at [0021] ("Once the application is recognized, QoS or other policies associated with the application can be applied to traffic so that the network can invoke services for that par-ticular application. For example, the application may have certain requirements and expectations from the network infrastructure, which may be specified in terms of bandwidth, delay, jitter, throughput, packet loss, or other performance attributes.")<br><br>Chandrasekaran at [0023] ("In one embodiment, the classification information 26 transmitted from the controller 12 to the AP 14 includes tuple information for a flow ( e.g., source IP address, destina-tion IP address, source port, destination port, and protocol), application identifier (ID), and stateless DPI information. Stateless DPI information includes classification and sub-classification information ( e.g., fixed or variable offset with a pattern or regular expression) and rules for applying policies on the sub-classified packets. The policies may include, for example, drop packet, mark a DSCP (Differentiated Services Code Point) value in the packet, or rate limit the traffic.")<br><br>Chandrasekaran at [0026] ("Memory 34 may be a volatile memory or non-vola-tile storage, which stores various applications, operating sys-tems, modules, and data for execution and use by the proces-sor 32. Memory 34 may include, for example, classification database 35, |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 427 of 1100

| No. | '111 Patent Claim 9 | Chandrasekaran |
|---|---|---|
| | | The classification database 35 may be any data structure configured for at least temporarily storing classifi-cation information including, for example, flow information, application ID, stateless DPI rules, and policies.")<br><br>Chandrasekaran at [0031] ("FIG. 3 is a flowchart illustrating an example of a process at the controller 12 for classification of traffic for application aware policies in a wireless network, in accor-dance with one embodiment. At step 40, the controller 12 receives packets belonging to a network flow. The controller 12 performs stateful classification to identify an application associated with the flow ( step 42). The controller 12 transmits classification information ( e.g., flow information, stateless DPI rule, and policy) to the AP 14 for use in stateless classi-fication at the AP (step 44). The controller 12 applies policies to downstream traffic (received at the controller and destined for the client 16) (step 46) and receives upstream traffic for which policies have been applied at the AP 14 (step 48). If the controller 12 determines ( e.g., receives an indication) that the client 16 has roamed, it transmits the classification informa-tion to the new AP 14 to which the client has roamed (steps 50 and 52).")<br><br>Chandrasekaran at [0033] ("The following describes an example of the above process for WebEx traffic that has different sub-classifications for voice and video traffic. Stateful classification is first performed by the controller 12 at the beginning of the flow. The controller 12 may need to process, for example, 10, 100, or any other number of packets to classify the flow as Web Ex traffic. Once the classification is performed, the controller 12 sends the stateless DPI rules and flow information to the AP 14 for stateless sub-classification to distinguish voice, video, or data within a WebEx flow. For example, after the controller 12 identifies the WebEx meeting traffic, it pushes the tuple, the stateless DPI rules (as shown below), and policies to the AP 14 for upstream traffic marking, dropping, or rate-limit-ing. If the client 16 roams, the controller 12 transmits the same classification information to the new AP to which the client has roamed.")<br><br>Chandrasekaran at Figure 3 (annotations added) |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 428 of 1100

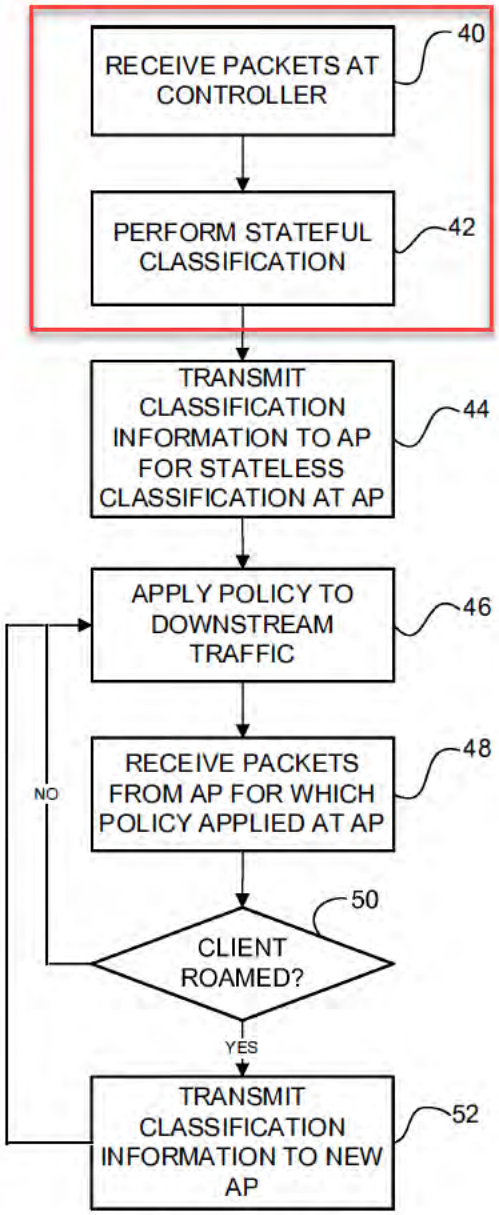



FIGURE 3



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 429 of 1100

| No. | '111 Patent Claim 12 | Chandrasekaran |
|---|---|---|
| 12 | The method according to claim 9, wherein the analyzing comprises applying security or data analytic application. | Chandrasekaran discloses the method according to claim 9, wherein the analyzing comprises applying security or data analytic application.<br><br>For example, Chandrasekaran discloses analyzing the packet by the controller through stateful classification where the classification is done in order to recognize the packet. A person of ordinary skill in the art would understand that stateful classification to recognize the packet is performed as a security and data analytic measure. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met.<br><br>*See supra* at Claim 9.<br><br>Chandrasekaran at [0014] ("The term 'wireless controller' or 'controller' as used herein may refer to a wireless LAN (local area network) controller, mobility controller, wireless control device, wire-less control system, or any other network device operable to perform control functions for a wireless network. The net-work site may also include a wireless control system or other platform for centralized wireless LAN planning, configura-tion, and management. The wireless controller 12 enables system wide functions for wireless applications and may support any number of access points 14. Each access point 14 may serve any number of mobile devices 16 in the wireless network. The wireless controller 12 may be, for example, a standalone device or a rack-mounted appliance. In the example shown in FIG. 1, the wireless controller 12 and access points 14 are separate devices and may be located remote from one another. The wireless controller 12 may also be integrated with the access point 14 ( e.g., autonomous AP) or located at a switch, router, switch/router, or other network device. Thus, the wireless controller 12 may be a physical device located at a standalone device, access point, switch, router, or other network device. The wireless controller 12 may also be a virtual device located in a network or cloud, for example.")<br><br>Chandrasekaran at [0016] ("The wireless controller 12 includes a stateful appli-cation classifier 18 and the AP 14 includes a stateless appli-cation classifier 22. After the stateful classifier 18 identifies the application, the controller 12 transmits ( e.g., pushes) clas-sification information 26 to the AP 14 so that the AP can perform stateless classification and apply policies ( e.g., QoS or other policies) to traffic received from the mobile device 16. The controller 12 may also provide the classification information 26 to |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 430 of 1100

| No. | '111 Patent Claim 12 | Chandrasekaran |
|---|---|---|
| | | another AP 14 if the client 16 roams to a new AP, as shown in FIG. 1. Implementation of the stateful classifier 18 at the controller 12 and stateless classifier 22 at the AP 14 allows for policies to be applied for downstream traffic (packet 25) at the wireless controller 12, and for upstream traffic (packet 28) at the access point 14.")<br><br>Chandrasekaran at [0020] ("In one embodiment, the stateful classifier 18 is a classification engine configured for NBAR (Network Based Application Recognition) or other technology used to classify applications. The classifier 18 is operable to recognize a wide variety of applications, including Web-based and client/ server applications. The applications may include, for example, Skype, YouTube, Netflix, WebEx, Google Voice, BitTorrent, Citrix, virtual desktop, PCoIP, or any other appli-cation. The classification engine may be configured, for example, to identify generic protocols and perform heuristic analysis for encrypted protocols. The classifiers 18, 22 are configured to perform deep packet inspection (DPI), which provides the ability to look into the packet past basic header information so that the contents of a particular packet can be determined.")<br><br>Chandrasekaran at [0021] ("Once the application is recognized, QoS or other policies associated with the application can be applied to traffic so that the network can invoke services for that par-ticular application. For example, the application may have certain requirements and expectations from the network infrastructure, which may be specified in terms of bandwidth, delay, jitter, throughput, packet loss, or other performance attributes.")<br><br>Chandrasekaran at [0023] ("In one embodiment, the classification information 26 transmitted from the controller 12 to the AP 14 includes tuple information for a flow ( e.g., source IP address, destina-tion IP address, source port, destination port, and protocol), application identifier (ID), and stateless DPI information. Stateless DPI information includes classification and sub-classification information ( e.g., fixed or variable offset with a pattern or regular expression) and rules for applying policies on the sub-classified packets. The policies may include, for example, drop packet, mark a DSCP (Differentiated Services Code Point) value in the packet, or rate limit the traffic.")<br><br>Chandrasekaran at [0026] ("Memory 34 may be a volatile memory or non-vola-tile storage, which stores various applications, operating sys-tems, modules, and data for execution and use by the proces-sor 32. Memory 34 may include, for example, classification database 35, |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 431 of 1100

| No. | '111 Patent Claim 12 | Chandrasekaran |
|---|---|---|
| | | The classification database 35 may be any data structure configured for at least temporarily storing classifi-cation information including, for example, flow information, application ID, stateless DPI rules, and policies.")<br><br>Chandrasekaran at [0031] ("FIG. 3 is a flowchart illustrating an example of a process at the controller 12 for classification of traffic for application aware policies in a wireless network, in accor-dance with one embodiment. At step 40, the controller 12 receives packets belonging to a network flow. The controller 12 performs stateful classification to identify an application associated with the flow ( step 42). The controller 12 transmits classification information ( e.g., flow information, stateless DPI rule, and policy) to the AP 14 for use in stateless classi-fication at the AP (step 44). The controller 12 applies policies to downstream traffic (received at the controller and destined for the client 16) (step 46) and receives upstream traffic for which policies have been applied at the AP 14 (step 48). If the controller 12 determines ( e.g., receives an indication) that the client 16 has roamed, it transmits the classification informa-tion to the new AP 14 to which the client has roamed (steps 50 and 52).")<br><br>Chandrasekaran at [0033] ("The following describes an example of the above process for WebEx traffic that has different sub-classifications for voice and video traffic. Stateful classification is first performed by the controller 12 at the beginning of the flow. The controller 12 may need to process, for example, 10, 100, or any other number of packets to classify the flow as Web Ex traffic. Once the classification is performed, the controller 12 sends the stateless DPI rules and flow information to the AP 14 for stateless sub-classification to distinguish voice, video, or data within a WebEx flow. For example, after the controller 12 identifies the WebEx meeting traffic, it pushes the tuple, the stateless DPI rules (as shown below), and policies to the AP 14 for upstream traffic marking, dropping, or rate-limit-ing. If the client 16 roams, the controller 12 transmits the same classification information to the new AP to which the client has roamed.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 432 of 1100

| No. | '111 Patent Claim 13 | Chandrasekaran |
|---|---|---|
| 13 | The method according to claim 9, wherein the analyzing comprises applying security application that comprises firewall or intrusion detection functionality. | Chandrasekaran discloses the method according to claim 9, wherein the analyzing comprises applying security application that comprises firewall or intrusion detection functionality.<br><br>For example, Chandrasekaran discloses analyzing the packet by the controller through stateful classification where the classification is done in order to recognize the packet. A person of ordinary skill in the art would understand that stateful classification in order to recognize the packet may be performed by applying security application including a firewall or intrusion detection functionality. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Chandrasekaran is found to not meet this limitation, wherein the analyzing comprises applying security application that comprises firewall or intrusion detection functionality would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>*See supra* at Claim 9.<br><br>Chandrasekaran at [0014] ("The term 'wireless controller' or 'controller' as used herein may refer to a wireless LAN (local area network) controller, mobility controller, wireless control device, wire-less control system, or any other network device operable to perform control functions for a wireless network. The net-work site may also include a wireless control system or other platform for centralized wireless LAN planning, configura-tion, and management. The wireless controller 12 enables system wide functions for wireless applications and may support any number of access points 14. Each access point 14 may serve any number of mobile devices 16 in the wireless network. The wireless controller 12 may be, for example, a standalone device or a rack-mounted appliance. In the example shown in FIG. 1, the wireless controller 12 and access points 14 are separate devices and may be located remote from one another. The wireless controller 12 may also be integrated with the access point 14 ( e.g., autonomous AP) or located at a switch, router, switch/router, or other network device. Thus, the wireless controller 12 may be a physical device located at a standalone device, access point, switch, router, or other network device. The wireless controller 12 may also be a virtual device located in a network or cloud, for example.")<br><br>Chandrasekaran at [0016] ("The wireless controller 12 includes a stateful appli-cation classifier 18 and the AP 14 includes a stateless appli-cation classifier 22. After the stateful |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 433 of 1100

| No. | '111 Patent Claim 13 | Chandrasekaran |
|---|---|---|
| | | classifier 18 identifies the application, the controller 12 transmits ( e.g., pushes) clas-sification information 26 to the AP 14 so that the AP can perform stateless classification and apply policies ( e.g., QoS or other policies) to traffic received from the mobile device 16. The controller 12 may also provide the classification information 26 to another AP 14 if the client 16 roams to a new AP, as shown in FIG. 1. Implementation of the stateful classifier 18 at the controller 12 and stateless classifier 22 at the AP 14 allows for policies to be applied for downstream traffic (packet 25) at the wireless controller 12, and for upstream traffic (packet 28) at the access point 14.")<br><br>Chandrasekaran at [0020] ("In one embodiment, the stateful classifier 18 is a classification engine configured for NBAR (Network Based Application Recognition) or other technology used to classify applications. The classifier 18 is operable to recognize a wide variety of applications, including Web-based and client/ server applications. The applications may include, for example, Skype, YouTube, Netflix, WebEx, Google Voice, BitTorrent, Citrix, virtual desktop, PCoIP, or any other appli-cation. The classification engine may be configured, for example, to identify generic protocols and perform heuristic analysis for encrypted protocols. The classifiers 18, 22 are configured to perform deep packet inspection (DPI), which provides the ability to look into the packet past basic header information so that the contents of a particular packet can be determined.")<br><br>Chandrasekaran at [0021] ("Once the application is recognized, QoS or other policies associated with the application can be applied to traffic so that the network can invoke services for that par-ticular application. For example, the application may have certain requirements and expectations from the network infrastructure, which may be specified in terms of bandwidth, delay, jitter, throughput, packet loss, or other performance attributes.")<br><br>Chandrasekaran at [0023] ("In one embodiment, the classification information 26 transmitted from the controller 12 to the AP 14 includes tuple information for a flow ( e.g., source IP address, destina-tion IP address, source port, destination port, and protocol), application identifier (ID), and stateless DPI information. Stateless DPI information includes classification and sub-classification information ( e.g., fixed or variable offset with a pattern or regular expression) and rules for applying policies on the sub-classified packets. The policies may include, for example, drop packet, mark a DSCP (Differentiated Services Code Point) value in the packet, or rate limit the traffic.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 434 of 1100

| No. | '111 Patent Claim 13 | Chandrasekaran |
|---|---|---|
| | | Chandrasekaran at [0026] ("Memory 34 may be a volatile memory or non-vola-tile storage, which stores various applications, operating sys-tems, modules, and data for execution and use by the proces-sor 32. Memory 34 may include, for example, classification database 35. The classification database 35 may be any data structure configured for at least temporarily storing classifi-cation information including, for example, flow information, application ID, stateless DPI rules, and policies.")<br><br>Chandrasekaran at [0031] ("FIG. 3 is a flowchart illustrating an example of a process at the controller 12 for classification of traffic for application aware policies in a wireless network, in accor-dance with one embodiment. At step 40, the controller 12 receives packets belonging to a network flow. The controller 12 performs stateful classification to identify an application associated with the flow ( step 42). The controller 12 transmits classification information ( e.g., flow information, stateless DPI rule, and policy) to the AP 14 for use in stateless classi-fication at the AP (step 44). The controller 12 applies policies to downstream traffic (received at the controller and destined for the client 16) (step 46) and receives upstream traffic for which policies have been applied at the AP 14 (step 48). If the controller 12 determines ( e.g., receives an indication) that the client 16 has roamed, it transmits the classification informa-tion to the new AP 14 to which the client has roamed (steps 50 and 52).")<br><br>Chandrasekaran at [0033] ("The following describes an example of the above process for WebEx traffic that has different sub-classifications for voice and video traffic. Stateful classification is first performed by the controller 12 at the beginning of the flow. The controller 12 may need to process, for example, 10, 100, or any other number of packets to classify the flow as Web Ex traffic. Once the classification is performed, the controller 12 sends the stateless DPI rules and flow information to the AP 14 for stateless sub-classification to distinguish voice, video, or data within a WebEx flow. For example, after the controller 12 identifies the WebEx meeting traffic, it pushes the tuple, the stateless DPI rules (as shown below), and policies to the AP 14 for upstream traffic marking, dropping, or rate-limit-ing. If the client 16 roams, the controller 12 transmits the same classification information to the new AP to which the client has roamed.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 435 of 1100

| No. | '111 Patent Claim 13 | Chandrasekaran |
|---|---|---|
| | | Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Chandrasekaran in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 13 of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references.<br><br>For example, Copeland discloses analysis by the intrusion detection engine on the monitoring appliance to detect communication intruders and suspicious activity. The monitoring appliance may also work in coordination with a firewall.<br><br>Copeland at [0065] ("The intrusion detection engine 155 analyzes the flow data 160 to determine if the flow appears to be legitimate traffic or possible suspicious activity. Flows with suspicious activity are assigned a predetermined concern index (CI) value based upon a heuristically predetermined assessment of the significance of the threat of the particular traffic or flow or suspicious activity. The flow concern index values have been derived heuristically from extensive net-work traffic analysis. Concern index values are associated with particular hosts and stored in the host data structure 166 (FIG. 1). Exemplary concern index values for various exemplary flow-based events and other types of events are illustrated in connection with FIGS. 6 and 7.")<br><br>Copeland at [0067] ("The host servers 130 are directly or indirectly coupled to one or more network devices 135 such as routers or switches that support providing a sampled data stream such as that provided by sFlow. In a typical preferred configuration for the present invention, a monitoring appli-ance 150 operating a flow-based intrusion detection engine 155 is receiving sampled packet headers from one or more network devices 135. The monitoring appliance 150 moni-tors the communications between the host server 130 and other hosts 120, 110 in the attempt to detect intrusion activity.")<br><br>Copeland at [0068] ("Those skilled in the art understand that many networks utilize firewalls to limit unwanted network traffic. A monitoring appliance 150 can be connected before a firewall to detect intrusions directed at the network. Con-versely, the monitoring appliance 150 may be installed behind a firewall to detect intrusions that bypass the firewall. Some systems install two firewalls with web and e-mails servers in the so-called "demilitarized zone" or "DMZ" between firewalls. One common placement of the |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 436 of 1100

| No. | '111 Patent Claim 13 | Chandrasekaran |
|---|---|---|
| | | monitor-ing appliance 150 is in this demilitarized zone. Of course, those skilled in the art will appreciate that the flow-based intrusion detection system 155 or appliance 150 can operate without the existence of any firewalls.")<br><br>Copeland at [0069] ("It will now be appreciated that the disclosed meth-odology of intrusion detection is accomplished at least in part by analyzing communication flows to determine if such communications have the flow characteristics of probes or attacks. By analyzing communications for abnormal flow characteristics, attacks can be determined without the need for resource-intensive packet data analysis. A flow can be determined from the packets 101 that are transmitted between two hosts utilizing a single service. The addresses and port numbers of communications are easily discerned by analysis of the header information in a datagram.")<br><br>Copeland at [0112] ("FIG. 5 illustrates a logical software architecture of a flow-based intrusion detection engine 155 constructed in accordance with an embodiment of the present invention. As will be understood by those skilled in the art, the system is constructed utilizing Internet-enabled computer systems with computer programs designed to carry out the functions described herein. Preferably, the various computing func-tions are implemented as different but related processes known as "threads" which executed concurrently on modern day multi-threaded, multitasking computer systems.")<br><br>Copeland at Figure 5 |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 437 of 1100

| No. | '111 Patent Claim 13 | Chandrasekaran |
|---|---|---|

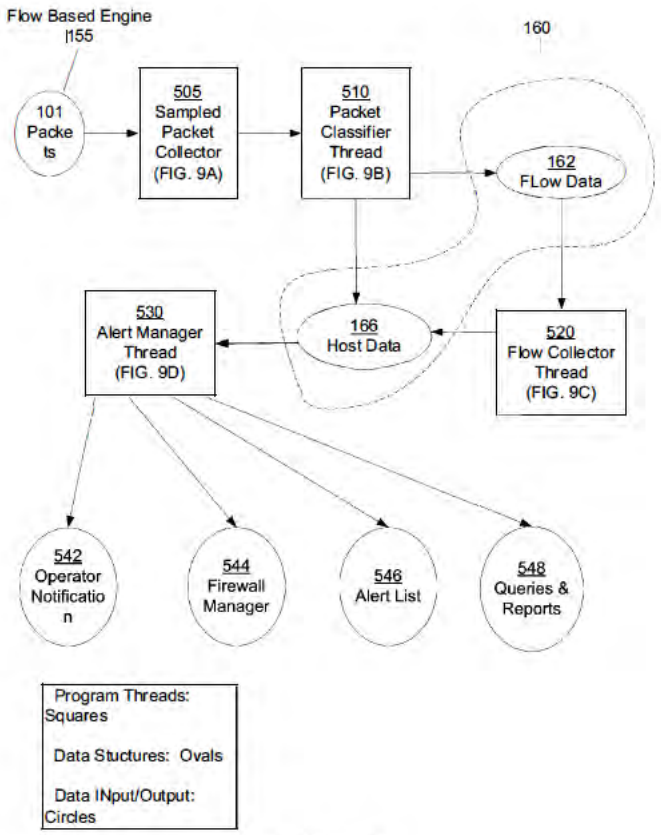



FIG. 5

Copeland at [0149] ("The alert manager 530 also looks for hosts whose CI or traffic (byte rate) exceeds preset alarm thresholds and which have not been handled on previous runs. The new alarm conditions can cause immediate operator notification by an operator notification process 542. These conditions can be highlighted on the user interface, and cause SNMP trap messages to be sent to a network monitor such as HP Openview, and/or email messages to the network adminis-trator that in turn may cause messages to be sent to beepers or cell phones. Messages can also be sent to cause automated devices such as a firewall manager 544 to drop packets going to or from an offending host. It will thus be



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 438 of 1100

| No. | '111 Patent Claim 13 | Chandrasekaran |
|---|---|---|
| | | appreciated that the present invention advantageously operates in conjunc-tion with firewalls and other network security devices and processes to provide additional protection for an entity's computer network and computer resources.")<br><br>Copeland at [0177] ("If an alarm threshold has been exceeded, the "Yes" branch of step 975 is followed to step 976. In step 976, the alert manager thread generates certain predetermined signals designed to drawn the attention of a system administrator or other interested person. The alert manager 530 looks for hosts whose CI or traffic (byte rate) exceeds preset alarm thresholds and have not been handled on previous runs. The new alarm conditions can cause immediate operator notifi-cation. These conditions can be highlighted on the user interface, and cause SNMP trap messages to be sent to a network monitor such as HP Openview, and/or email mes-sages to the network administrator that in turn may cause messages to be sent to beepers or cell phones. Messages can also be sent to cause automated devices such as a firewall manager to drop packets going to or from an offending host. Step 976 is followed by step 972, in which the thread 530 awaits the requisite amount of time.")<br><br>For example, Chua '877 discloses security determinations and analysis that involve the use of firewalls or intrusion detection services.<br><br>Chua '877 at 31:48-59 ("In some examples, SDN controller 112 further performs deep packet inspection (DPI) on packets from client device 102 ( 402). For example, SDN controller 112 may inspect one or more preliminary packets of packet flows originating from or directed to client device 102, and after determining that the packet flows are not malicious ( after a predetermined number of packets), stop inspecting the packet flows. Alternatively, SDN controller 112 may program network devices of SDN 106 to forward a predetermined number of packets of the packet flows originating from or destined for client device 102 through a deep packet inspection service device, which may correspond to one of service devices 116.")<br><br>Chua '877 at 25:32-52 ("In the example of FIG. 5, SDN controller 112 determines zones for packet flows through the network devices forming the SDN (304). The zones generally correspond to packet flows, that is, paths through the SDN followed by particular packets. SDN controller 112 may store data defining the zones in the data model discussed above, |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 439 of 1100

| No. | '111 Patent Claim 13 | Chandrasekaran |
|---|---|---|
| | | The data defining the zones may specify entities (e.g., users, devices, or the like) that have access to each zone. Thus, SDN controller 112 may program network devices of the SDN such that entities that are not authorized to access a particular zone are prevented from accessing the zone. SDN controller 112 may specify a zone using packet header field values, such as a source port, a destination port, a source IP address, a destination IP address, a virtual local area network (VLAN) tag, multiprotocol label switching (MPLS) labels, a packet protocol, and/or an IP subnet. In some cases, SDN controller 112 may specify whether a corresponding packet flow for a zone is suspect or malicious and construct the zone such that packets of the packet flow are prevented from reaching an intended destination. As noted above, zones may be ordered based on priority values when overlap occurs.")<br><br>Chua '877 at 25:53-65 ("Furthermore, SDN controller 112 determines trusted packet flows (306). For example, SDN controller 112 may determine that certain packet flows can be trusted based on security controls, and that other packet flows cannot be trusted based on the security controls. That is, SDN controller 112 may determine whether a packet flow can be trusted based on values of packet headers for the packet flows, e.g., values of headers at various layers of the OSI model ( e.g., any or all of layers 2-7 of the OSI model). In some examples, SDN controller 112 may omit any or all of steps 302, 304, and 306, e.g., omitting any or all of determination of service devices, determination of zones, and/or determination of trusted packet flows.")<br><br>Chua '877 at 5:50-6:5 ("SDN 106 generally serves to interconnect various endpoint devices, such as client device 102 and server device 104. In addition, SDN 106 may provide services to network traffic flowing between client device 102 and server device 104. Alternatively, SDN 106 may provide services to client device 102, without further directing traffic to server device 106. For example, administrator 114 may use SDN controller 112 to program network devices of SDN 106 to direct network traffic for client device 102 to one or more of service devices 116. Service devices 116 may include, for example, intrusion detection service (IDS) devices, intrusion prevention system (IPS) devices, web proxies, web servers, web-application firewalls and the like. In other examples, service devices 116 may, additionally or alternatively, include devices for provid-ing services such as, for example, denial of service (DoS) protection, distributed denial of service (DDoS) protection, traffic filtering, wide area network (WAN) acceleration, or other such services. Service devices |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 440 of 1100

| No. | '111 Patent Claim 13 | Chandrasekaran |
|---|---|---|
| | | 116 may also addition-ally or alternatively include malware detection devices, net-work anti-virus devices, network packet capture and analysis devices, honeypot devices, reflector net devices, tar pit devices, domain name service (DNS) and global DNS server devices, mail proxies, and anti-spam devices.")<br><br>Chua '877 at 6:6-24 ("Service devices 116 may, additionally or alternatively, include devices in various device categories such as, for example, network and application security devices, application optimization devices, scaling devices, traffic shaping devices, and/or monitoring and analytics devices. Moreover, although shown as individual devices, it should be understood that service devices may be realized by physical devices, multi-tenant devices, or using virtual services (e.g., cloud-based services). Moreover, service devices 116 may represent multi-function devices. For purposes of example and ease of explanation, this disclosure primarily describes individual service devices. However, it should be understood that the techniques of this disclosure may be readily applied to virtual devices and cloud-based applications, in addition or in the alternative to physical devices. Likewise, where this disclosure refers to a switch or other network device, it should be understood that these techniques may apply to virtual switches or other virtual network devices.")<br><br>Chua '877 at 7:3-13 ("Devices that may be plugged into (that is, communicatively coupled to) SDN controller 112 ( also sometimes referred to as a "FlowDirector") generally include classes of devices found in most network-based DMZs, including firewalls, web proxies, mail proxies, AV (anti-virus) proxies, mail systems, IDS (intrusion detection systems), IPS (intrusion prevention systems), VPN (virtual private network) servers, web application firewalls, vulnerability scanners, network recording and analysis systems, and packet shapers. Most of these devices are either security devices, or traffic engineering or visibility devices, in some examples.")<br><br>Chua '877 at 14:32-51 ("One example use case for SDN controller 112 includes performing internal security zone partitioning. In today's enterprise environment, certain flows can be trusted, based on security controls placed on the end points, while others must be assumed to have some potential for risk. SDN controller 112 may create security zones based both on physical topol-ogy as well as threat assessments based on L2-L4 header information. Business-level security rules can be implemented directly on SDN controller 112 to direct only higher risk flows through specific L4-L 7 devices ( e.g., service devices 116) to |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 441 of 1100

| No. | '111 Patent Claim 13 | Chandrasekaran |
|---|---|---|
| | | monitor for or block malicious traffic. That is, when an SDN interconnects a set of enterprise network devices of a common enterprise network and also provides connections for the enterprise network devices outside of the enterprise network, SDN controller 112 may determine that connections between the enterprise network devices within the enterprise network can be trusted, whereas connections to network devices outside the enterprise network cannot be trusted and, therefore, should be monitored by a security device.")<br><br>Chua '877 at 14:52-63 ("Thus, SDN controller 112 may determine separate sets of packet flows based on security controls, e.g., a first set of packet flows that can be trusted and a second set of packet flows that are not trusted. Then, SDN controller 112 may determine a first set of one or more paths for the first set of packet flows that omit a security device for the first set of packet flows (that is, based on the determination that the first set of packet flows can be trusted), and a second set of one or more paths for the second set of packet flows that direct the second set of packet flows through the security device (based on the determination that the second set of packet flows are not trusted).")<br><br>Chua '877 at 14:64-15:3 ("The security controls may include various types of information. For example, the security controls may specify values for one or more packet headers at various layers of the Open Systems Interconnection (OSI) network model. The security controls may specify information for any or all of network layers two, three, four, five, six, and/or seven of the OSI model.")<br><br>Chua '877 at 16:23-44 ("More particularly, control unit 130 may configure any of service devices 116 to send data representative of a particular event to SDN controller 112, and control unit 130 may auto-matically reprogram one or more network devices of SDN 106 in response to such data. For example, security monitor-ing applications of service devices 116 may determine that a specific source port, destination port, source IP address, des-tination IP address, or the like should be acted upon. Alter-natively, security monitoring applications may determine that, due to content or deep packet inspection, a specific type of traffic is malicious and should be blocked. In either case, the corresponding one of service devices 116 may send a message to SDN controller 112 representative of these deter-minations. As yet another example, a network performance device may monitor various performance metrics, such as latency, jitter, packet loss, or the like, and provide feedback data to SDN |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 442 of 1100

| No. | '111 Patent Claim 13 | Chandrasekaran |
|---|---|---|
| | | controller 112 based on these metrics. SDN controller 112 may respond by programming network devices of SDN 106 to perform a programmed action, such as allowing corresponding traffic, blocking corresponding traf-fic, mirroring corresponding traffic, redirecting correspond-ing traffic.")<br><br>Chua '877 at 19:60-20:4 ("FIG. 3 is a conceptual diagram illustrating an example 60 system 200 including various devices that may be used in accordance with the techniques of this disclosure. In this example, system 200 includes various network devices, including firewall 206, router 208, switch 210, web proxy 212, intrusion detection system (IDS) 214, web server 216, 65 administrator ("admin") workstation 220, and software defined network (SDN) controller 218. Web clients 202 can access system 200 via a network, such as the Internet, e.g., Internet 204. Internet 204 may include additional network devices not explicitly shown in FIG. 3, such as routers, switches, hubs, gateways, security devices, or the like.") |

| No. | '111 Patent Claim 14 | Chandrasekaran |
|---|---|---|
| 14 | The method according to claim 9, wherein the analyzing comprises performing Deep Packet Inspection (DPI) or using a DPI engine on the packet. | Chandrasekaran discloses the method according to claim 9, wherein the analyzing comprises performing Deep Packet Inspection (DPI) or using a DPI engine on the packet.<br><br>For example, Chandrasekaran discloses the controller performing Deep Packet Inspection, which provides the ability to look into the packet past basic header information so that the contents of a particular packet can be determined.<br><br>*See supra* at Claim 9.<br><br>Chandrasekaran at [0014] ("The term 'wireless controller' or 'controller' as used herein may refer to a wireless LAN (local area network) controller, mobility controller, wireless control device, wire-less control system, or any other network device operable to perform control functions for a wireless network. The net-work site may also include a wireless control system or other platform for centralized wireless LAN planning, configura-tion, and management. The wireless controller 12 enables system wide functions for wireless applications and may support any number of access points 14. Each access point 14 may |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 443 of 1100

| No. | '111 Patent Claim 14 | Chandrasekaran |
|---|---|---|
| | | serve any number of mobile devices 16 in the wireless network. The wireless controller 12 may be, for example, a standalone device or a rack-mounted appliance. In the example shown in FIG. 1, the wireless controller 12 and access points 14 are separate devices and may be located remote from one another. The wireless controller 12 may also be integrated with the access point 14 ( e.g., autonomous AP) or located at a switch, router, switch/router, or other network device. Thus, the wireless controller 12 may be a physical device located at a standalone device, access point, switch, router, or other network device. The wireless controller 12 may also be a virtual device located in a network or cloud, for example.")<br><br>Chandrasekaran at [0016] ("The wireless controller 12 includes a stateful appli-cation classifier 18 and the AP 14 includes a stateless appli-cation classifier 22. After the stateful classifier 18 identifies the application, the controller 12 transmits ( e.g., pushes) clas-sification information 26 to the AP 14 so that the AP can perform stateless classification and apply policies ( e.g., QoS or other policies) to traffic received from the mobile device 16. The controller 12 may also provide the classification information 26 to another AP 14 if the client 16 roams to a new AP, as shown in FIG. 1. Implementation of the stateful classifier 18 at the controller 12 and stateless classifier 22 at the AP 14 allows for policies to be applied for downstream traffic (packet 25) at the wireless controller 12, and for upstream traffic (packet 28) at the access point 14.")<br><br>Chandrasekaran at [0020] ("In one embodiment, the stateful classifier 18 is a classification engine configured for NBAR (Network Based Application Recognition) or other technology used to classify applications. The classifier 18 is operable to recognize a wide variety of applications, including Web-based and client/ server applications. The applications may include, for example, Skype, YouTube, Netflix, WebEx, Google Voice, BitTorrent, Citrix, virtual desktop, PCoIP, or any other appli-cation. The classification engine may be configured, for example, to identify generic protocols and perform heuristic analysis for encrypted protocols. The classifiers 18, 22 are configured to perform deep packet inspection (DPI), which provides the ability to look into the packet past basic header information so that the contents of a particular packet can be determined.")<br><br>Chandrasekaran at [0021] ("Once the application is recognized, QoS or other policies associated with the application can be applied to traffic so that the network can invoke services for that par-ticular application. For example, the application may have certain |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 444 of 1100

| No. | '111 Patent Claim 14 | Chandrasekaran |
|---|---|---|
| | | requirements and expectations from the network infrastructure, which may be specified in terms of bandwidth, delay, jitter, throughput, packet loss, or other performance attributes.")<br><br>Chandrasekaran at [0023] ("In one embodiment, the classification information 26 transmitted from the controller 12 to the AP 14 includes tuple information for a flow ( e.g., source IP address, destina-tion IP address, source port, destination port, and protocol), application identifier (ID), and stateless DPI information. Stateless DPI information includes classification and sub-classification information ( e.g., fixed or variable offset with a pattern or regular expression) and rules for applying policies on the sub-classified packets. The policies may include, for example, drop packet, mark a DSCP (Differentiated Services Code Point) value in the packet, or rate limit the traffic.")<br><br>Chandrasekaran at [0026] ("Memory 34 may be a volatile memory or non-vola-tile storage, which stores various applications, operating sys-tems, modules, and data for execution and use by the proces-sor 32. Memory 34 may include, for example, classification database 35. The classification database 35 may be any data structure configured for at least temporarily storing classifi-cation information including, for example, flow information, application ID, stateless DPI rules, and policies.")<br><br>Chandrasekaran at [0031] ("FIG. 3 is a flowchart illustrating an example of a process at the controller 12 for classification of traffic for application aware policies in a wireless network, in accor-dance with one embodiment. At step 40, the controller 12 receives packets belonging to a network flow. The controller 12 performs stateful classification to identify an application associated with the flow ( step 42). The controller 12 transmits classification information ( e.g., flow information, stateless DPI rule, and policy) to the AP 14 for use in stateless classi-fication at the AP (step 44). The controller 12 applies policies to downstream traffic (received at the controller and destined for the client 16) (step 46) and receives upstream traffic for which policies have been applied at the AP 14 (step 48). If the controller 12 determines ( e.g., receives an indication) that the client 16 has roamed, it transmits the classification informa-tion to the new AP 14 to which the client has roamed (steps 50 and 52).")<br><br>Chandrasekaran at [0033] ("The following describes an example of the above process for WebEx traffic that has different sub-classifications for voice and video traffic. Stateful |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 445 of 1100

| No. | '111 Patent Claim 14 | Chandrasekaran |
|---|---|---|
| | | classification is first performed by the controller 12 at the beginning of the flow. The controller 12 may need to process, for example, 10, 100, or any other number of packets to classify the flow as Web Ex traffic. Once the classification is performed, the controller 12 sends the stateless DPI rules and flow information to the AP 14 for stateless sub-classification to distinguish voice, video, or data within a WebEx flow. For example, after the controller 12 identifies the WebEx meeting traffic, it pushes the tuple, the stateless DPI rules (as shown below), and policies to the AP 14 for upstream traffic marking, dropping, or rate-limit-ing. If the client 16 roams, the controller 12 transmits the same classification information to the new AP to which the client has roamed.") |

| No. | '111 Patent Claim 15 | Chandrasekaran |
|---|---|---|
| 15[a] | The method according to claim 9, wherein the packet comprises distinct header and payload fields, and | Chandrasekaran discloses the method according to claim 9, wherein the packet comprises distinct header and payload fields.<br><br>For example, Chandrasekaran discloses packets with header and payload fields.<br><br>*See supra* at Claim 9.<br><br>Chandrasekaran at [0014] ("The term 'wireless controller' or 'controller' as used herein may refer to a wireless LAN (local area network) controller, mobility controller, wireless control device, wire-less control system, or any other network device operable to perform control functions for a wireless network. The net-work site may also include a wireless control system or other platform for centralized wireless LAN planning, configura-tion, and management. The wireless controller 12 enables system wide functions for wireless applications and may support any number of access points 14. Each access point 14 may serve any number of mobile devices 16 in the wireless network. The wireless controller 12 may be, for example, a standalone device or a rack-mounted appliance. In the example shown in FIG. 1, the wireless controller 12 and access points 14 are separate devices and may be located remote from one another. The wireless controller 12 may also be integrated with the access point 14 ( e.g., autonomous AP) or located at a switch, router, switch/router, or other network device. Thus, the wireless controller 12 may be a physical device located at |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 446 of 1100

| No. | '111 Patent Claim 15 | Chandrasekaran |
|---|---|---|
| | | a standalone device, access point, switch, router, or other network device. The wireless controller 12 may also be a virtual device located in a network or cloud, for example.")<br><br>Chandrasekaran at [0016] ("The wireless controller 12 includes a stateful appli-cation classifier 18 and the AP 14 includes a stateless appli-cation classifier 22. After the stateful classifier 18 identifies the application, the controller 12 transmits ( e.g., pushes) clas-sification information 26 to the AP 14 so that the AP can perform stateless classification and apply policies ( e.g., QoS or other policies) to traffic received from the mobile device 16. The controller 12 may also provide the classification information 26 to another AP 14 if the client 16 roams to a new AP, as shown in FIG. 1. Implementation of the stateful classifier 18 at the controller 12 and stateless classifier 22 at the AP 14 allows for policies to be applied for downstream traffic (packet 25) at the wireless controller 12, and for upstream traffic (packet 28) at the access point 14.")<br><br>Chandrasekaran at [0020] ("In one embodiment, the stateful classifier 18 is a classification engine configured for NBAR (Network Based Application Recognition) or other technology used to classify applications. The classifier 18 is operable to recognize a wide variety of applications, including Web-based and client/ server applications. The applications may include, for example, Skype, YouTube, Netflix, WebEx, Google Voice, BitTorrent, Citrix, virtual desktop, PCoIP, or any other appli-cation. The classification engine may be configured, for example, to identify generic protocols and perform heuristic analysis for encrypted protocols. The classifiers 18, 22 are configured to perform deep packet inspection (DPI), which provides the ability to look into the packet past basic header information so that the contents of a particular packet can be determined.")<br><br>Chandrasekaran at [0021] ("Once the application is recognized, QoS or other policies associated with the application can be applied to traffic so that the network can invoke services for that par-ticular application. For example, the application may have certain requirements and expectations from the network infrastructure, which may be specified in terms of bandwidth, delay, jitter, throughput, packet loss, or other performance attributes.")<br><br>Chandrasekaran at [0023] ("In one embodiment, the classification information 26 transmitted from the controller 12 to the AP 14 includes tuple information for a flow ( e.g., source IP address, destina-tion IP address, source port, destination port, and protocol), |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 447 of 1100

| No. | '111 Patent Claim 15 | Chandrasekaran |
|---|---|---|
| | | application identifier (ID), and stateless DPI information. Stateless DPI information includes classification and sub-classification information ( e.g., fixed or variable offset with a pattern or regular expression) and rules for applying policies on the sub-classified packets. The policies may include, for example, drop packet, mark a DSCP (Differentiated Services Code Point) value in the packet, or rate limit the traffic.")<br><br>Chandrasekaran at [0026] ("Memory 34 may be a volatile memory or non-vola-tile storage, which stores various applications, operating sys-tems, modules, and data for execution and use by the proces-sor 32. Memory 34 may include, for example, classification database 35. The classification database 35 may be any data structure configured for at least temporarily storing classifi-cation information including, for example, flow information, application ID, stateless DPI rules, and policies.")<br><br>Chandrasekaran at [0031] ("FIG. 3 is a flowchart illustrating an example of a process at the controller 12 for classification of traffic for application aware policies in a wireless network, in accor-dance with one embodiment. At step 40, the controller 12 receives packets belonging to a network flow. The controller 12 performs stateful classification to identify an application associated with the flow ( step 42). The controller 12 transmits classification information ( e.g., flow information, stateless DPI rule, and policy) to the AP 14 for use in stateless classi-fication at the AP (step 44). The controller 12 applies policies to downstream traffic (received at the controller and destined for the client 16) (step 46) and receives upstream traffic for which policies have been applied at the AP 14 (step 48). If the controller 12 determines ( e.g., receives an indication) that the client 16 has roamed, it transmits the classification informa-tion to the new AP 14 to which the client has roamed (steps 50 and 52).")<br><br>Chandrasekaran at [0033] ("The following describes an example of the above process for WebEx traffic that has different sub-classifications for voice and video traffic. Stateful classification is first performed by the controller 12 at the beginning of the flow. The controller 12 may need to process, for example, 10, 100, or any other number of packets to classify the flow as Web Ex traffic. Once the classification is performed, the controller 12 sends the stateless DPI rules and flow information to the AP 14 for stateless sub-classification to distinguish voice, video, or data within a WebEx flow. For example, after the controller 12 identifies the WebEx meeting traffic, it pushes the tuple, the stateless DPI |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 448 of 1100

| No. | '111 Patent Claim 15 | Chandrasekaran |
|---|---|---|
| | | rules (as shown below), and policies to the AP 14 for upstream traffic marking, dropping, or rate-limit-ing. If the client 16 roams, the controller 12 transmits the same classification information to the new AP to which the client has roamed.")<br><br>Chandrasekaran at [0035]-[0044] ("WebEx Video:<br>UDP Payload<br>First byte=0x06<br>Bytes [6-9]=Data length<br>10th byte=0x50<br><br>WebEx Voice:<br>UDP Payload<br>First byte=0x06<br>Bytes [6-9]=Data length<br>10th byte=0x48") |
| 15[b] | wherein the analyzing comprises checking part of, or whole of, the payload field. | Chandrasekaran discloses wherein the analyzing comprises checking part of, or whole of, the payload field.<br><br>For example, Chandrasekaran discloses analyzing, including by deep packet inspection, by to looking into the packet past basic header information so that the contents, or payload, of a particular packet can be determined.<br><br>Chandrasekaran at [0014] ("The term 'wireless controller' or 'controller' as used herein may refer to a wireless LAN (local area network) controller, mobility controller, wireless control device, wire-less control system, or any other network device operable to perform control functions for a wireless network. The net-work site may also include a wireless control system or other platform for centralized wireless LAN planning, configura-tion, and management. The wireless controller 12 enables system wide functions for wireless applications and may support any number of access points 14. Each access point 14 may serve any number of mobile devices 16 in the wireless network. The wireless controller 12 may be, for example, a standalone device or a rack-mounted appliance. In the example shown in FIG. 1, the wireless controller 12 and access points 14 are separate devices and |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 449 of 1100

| No. | '111 Patent Claim 15 | Chandrasekaran |
|---|---|---|
| | | may be located remote from one another. The wireless controller 12 may also be integrated with the access point 14 ( e.g., autonomous AP) or located at a switch, router, switch/router, or other network device. Thus, the wireless controller 12 may be a physical device located at a standalone device, access point, switch, router, or other network device. The wireless controller 12 may also be a virtual device located in a network or cloud, for example.")<br><br>Chandrasekaran at [0016] ("The wireless controller 12 includes a stateful appli-cation classifier 18 and the AP 14 includes a stateless appli-cation classifier 22. After the stateful classifier 18 identifies the application, the controller 12 transmits ( e.g., pushes) clas-sification information 26 to the AP 14 so that the AP can perform stateless classification and apply policies ( e.g., QoS or other policies) to traffic received from the mobile device 16. The controller 12 may also provide the classification information 26 to another AP 14 if the client 16 roams to a new AP, as shown in FIG. 1. Implementation of the stateful classifier 18 at the controller 12 and stateless classifier 22 at the AP 14 allows for policies to be applied for downstream traffic (packet 25) at the wireless controller 12, and for upstream traffic (packet 28) at the access point 14.")<br><br>Chandrasekaran at [0020] ("In one embodiment, the stateful classifier 18 is a classification engine configured for NBAR (Network Based Application Recognition) or other technology used to classify applications. The classifier 18 is operable to recognize a wide variety of applications, including Web-based and client/ server applications. The applications may include, for example, Skype, YouTube, Netflix, WebEx, Google Voice, BitTorrent, Citrix, virtual desktop, PCoIP, or any other appli-cation. The classification engine may be configured, for example, to identify generic protocols and perform heuristic analysis for encrypted protocols. The classifiers 18, 22 are configured to perform deep packet inspection (DPI), which provides the ability to look into the packet past basic header information so that the contents of a particular packet can be determined.")<br><br>Chandrasekaran at [0021] ("Once the application is recognized, QoS or other policies associated with the application can be applied to traffic so that the network can invoke services for that par-ticular application. For example, the application may have certain requirements and expectations from the network infrastructure, which may be specified in terms of bandwidth, delay, jitter, throughput, packet loss, or other performance attributes.") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 450 of 1100

| No. | '111 Patent Claim 15 | Chandrasekaran |
|---|---|---|
| | | Chandrasekaran at [0023] ("In one embodiment, the classification information 26 transmitted from the controller 12 to the AP 14 includes tuple information for a flow ( e.g., source IP address, destina-tion IP address, source port, destination port, and protocol), application identifier (ID), and stateless DPI information. Stateless DPI information includes classification and sub-classification information ( e.g., fixed or variable offset with a pattern or regular expression) and rules for applying policies on the sub-classified packets. The policies may include, for example, drop packet, mark a DSCP (Differentiated Services Code Point) value in the packet, or rate limit the traffic.")<br><br>Chandrasekaran at [0026] ("Memory 34 may be a volatile memory or non-vola-tile storage, which stores various applications, operating sys-tems, modules, and data for execution and use by the proces-sor 32. Memory 34 may include, for example, classification database 35. The classification database 35 may be any data structure configured for at least temporarily storing classifi-cation information including, for example, flow information, application ID, stateless DPI rules, and policies.")<br><br>Chandrasekaran at [0031] ("FIG. 3 is a flowchart illustrating an example of a process at the controller 12 for classification of traffic for application aware policies in a wireless network, in accor-dance with one embodiment. At step 40, the controller 12 receives packets belonging to a network flow. The controller 12 performs stateful classification to identify an application associated with the flow ( step 42). The controller 12 transmits classification information ( e.g., flow information, stateless DPI rule, and policy) to the AP 14 for use in stateless classi-fication at the AP (step 44). The controller 12 applies policies to downstream traffic (received at the controller and destined for the client 16) (step 46) and receives upstream traffic for which policies have been applied at the AP 14 (step 48). If the controller 12 determines ( e.g., receives an indication) that the client 16 has roamed, it transmits the classification informa-tion to the new AP 14 to which the client has roamed (steps 50 and 52).")<br><br>Chandrasekaran at [0033] ("The following describes an example of the above process for WebEx traffic that has different sub-classifications for voice and video traffic. Stateful classification is first performed by the controller 12 at the beginning of the flow. The controller 12 may need to process, for example, 10, 100, or any other number of packets to classify the flow as Web Ex traffic. Once the classification is performed, the controller 12 |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 451 of 1100

| No. | '111 Patent Claim 15 | Chandrasekaran |
|---|---|---|
| | | sends the stateless DPI rules and flow information to the AP 14 for stateless sub-classification to distinguish voice, video, or data within a WebEx flow. For example, after the controller 12 identifies the WebEx meeting traffic, it pushes the tuple, the stateless DPI rules (as shown below), and policies to the AP 14 for upstream traffic marking, dropping, or rate-limit-ing. If the client 16 roams, the controller 12 transmits the same classification information to the new AP to which the client has roamed.")<br><br>Chandrasekaran at [0035]-[0044] ("WebEx Video:<br>UDP Payload<br>First byte=0x06<br>Bytes [6-9]=Data length<br>10th byte=0x50<br><br>WebEx Voice:<br>UDP Payload<br>First byte=0x06<br>Bytes [6-9]=Data length<br>10th byte=0x48") |

| No. | '111 Patent Claim 16 | Chandrasekaran |
|---|---|---|
| 16[a] | The method according to claim 1, wherein the packet comprises distinct header and payload fields, | Chandrasekaran discloses the method according to claim 1, wherein the packet comprises distinct header and payload fields.<br><br>*See supra* at Claim 1, 15[a]. |
| 16[b] | the header comprises one or more flag bits, and | Chandrasekaran discloses the header comprises one or more flag bits.<br><br>For example, Chandrasekaran discloses packets with header fields. A person of ordinary skill in the art would understand that the header could be comprised of one or more flag bits. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Chandrasekaran is found to not meet this |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 452 of 1100

| No. | '111 Patent Claim 16 | Chandrasekaran |
|---|---|---|
| | | limitation, the header comprises one or more flag bits would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>Chandrasekaran at [0014] ("The term 'wireless controller' or 'controller' as used herein may refer to a wireless LAN (local area network) controller, mobility controller, wireless control device, wire-less control system, or any other network device operable to perform control functions for a wireless network. The net-work site may also include a wireless control system or other platform for centralized wireless LAN planning, configura-tion, and management. The wireless controller 12 enables system wide functions for wireless applications and may support any number of access points 14. Each access point 14 may serve any number of mobile devices 16 in the wireless network. The wireless controller 12 may be, for example, a standalone device or a rack-mounted appliance. In the example shown in FIG. 1, the wireless controller 12 and access points 14 are separate devices and may be located remote from one another. The wireless controller 12 may also be integrated with the access point 14 ( e.g., autonomous AP) or located at a switch, router, switch/router, or other network device. Thus, the wireless controller 12 may be a physical device located at a standalone device, access point, switch, router, or other network device. The wireless controller 12 may also be a virtual device located in a network or cloud, for example.")<br><br>Chandrasekaran at [0016] ("The wireless controller 12 includes a stateful appli-cation classifier 18 and the AP 14 includes a stateless appli-cation classifier 22. After the stateful classifier 18 identifies the application, the controller 12 transmits ( e.g., pushes) clas-sification information 26 to the AP 14 so that the AP can perform stateless classification and apply policies ( e.g., QoS or other policies) to traffic received from the mobile device 16. The controller 12 may also provide the classification information 26 to another AP 14 if the client 16 roams to a new AP, as shown in FIG. 1. Implementation of the stateful classifier 18 at the controller 12 and stateless classifier 22 at the AP 14 allows for policies to be applied for downstream traffic (packet 25) at the wireless controller 12, and for upstream traffic (packet 28) at the access point 14.")<br><br>Chandrasekaran at [0020] ("In one embodiment, the stateful classifier 18 is a classification engine configured for NBAR (Network Based Application Recognition) or other technology used to classify applications. The classifier 18 is operable to recognize a wide variety of applications, including Web-based and client/ server applications. The applications may |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 453 of 1100

| No. | '111 Patent Claim 16 | Chandrasekaran |
|---|---|---|
| | | include, for example, Skype, YouTube, Netflix, WebEx, Google Voice, BitTorrent, Citrix, virtual desktop, PCoIP, or any other appli-cation. The classification engine may be configured, for example, to identify generic protocols and perform heuristic analysis for encrypted protocols. The classifiers 18, 22 are configured to perform deep packet inspection (DPI), which provides the ability to look into the packet past basic header information so that the contents of a particular packet can be determined.")<br><br>Chandrasekaran at [0021] ("Once the application is recognized, QoS or other policies associated with the application can be applied to traffic so that the network can invoke services for that par-ticular application. For example, the application may have certain requirements and expectations from the network infrastructure, which may be specified in terms of bandwidth, delay, jitter, throughput, packet loss, or other performance attributes.")<br><br>Chandrasekaran at [0023] ("In one embodiment, the classification information 26 transmitted from the controller 12 to the AP 14 includes tuple information for a flow ( e.g., source IP address, destina-tion IP address, source port, destination port, and protocol), application identifier (ID), and stateless DPI information. Stateless DPI information includes classification and sub-classification information ( e.g., fixed or variable offset with a pattern or regular expression) and rules for applying policies on the sub-classified packets. The policies may include, for example, drop packet, mark a DSCP (Differentiated Services Code Point) value in the packet, or rate limit the traffic.")<br><br>Chandrasekaran at [0026] ("Memory 34 may be a volatile memory or non-vola-tile storage, which stores various applications, operating sys-tems, modules, and data for execution and use by the proces-sor 32. Memory 34 may include, for example, classification database 35. The classification database 35 may be any data structure configured for at least temporarily storing classifi-cation information including, for example, flow information, application ID, stateless DPI rules, and policies.")<br><br>Chandrasekaran at [0031] ("FIG. 3 is a flowchart illustrating an example of a process at the controller 12 for classification of traffic for application aware policies in a wireless network, in accor-dance with one embodiment. At step 40, the controller 12 receives packets belonging to a network flow. The controller 12 performs stateful classification to identify an application associated with the flow ( step 42). The controller 12 transmits classification |

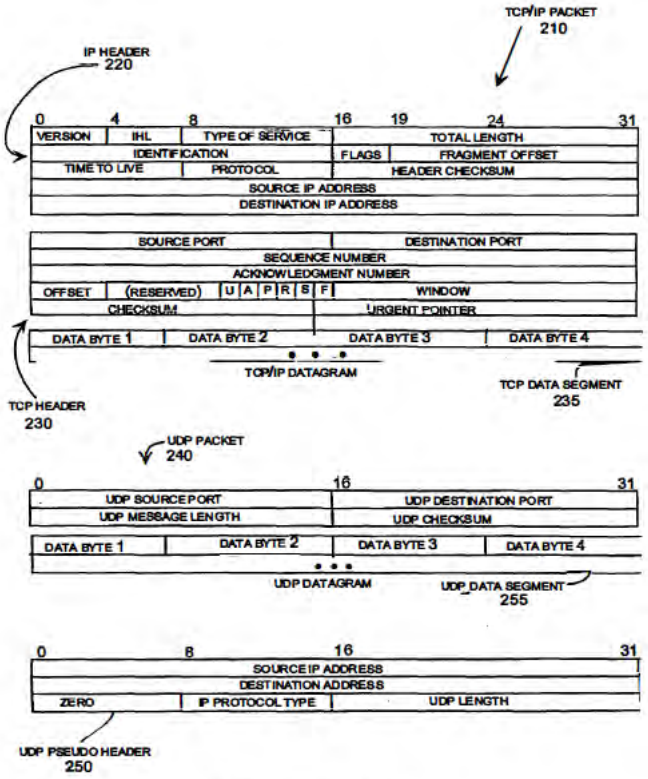


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 454 of 1100

| No. | '111 Patent Claim 16 | Chandrasekaran |
|---|---|---|
| | | information ( e.g., flow information, stateless DPI rule, and policy) to the AP 14 for use in stateless classi-fication at the AP (step 44). The controller 12 applies policies to downstream traffic (received at the controller and destined for the client 16) (step 46) and receives upstream traffic for which policies have been applied at the AP 14 (step 48). If the controller 12 determines ( e.g., receives an indication) that the client 16 has roamed, it transmits the classification informa-tion to the new AP 14 to which the client has roamed (steps 50 and 52).")<br><br>Chandrasekaran at [0033] ("The following describes an example of the above process for WebEx traffic that has different sub-classifications for voice and video traffic. Stateful classification is first performed by the controller 12 at the beginning of the flow. The controller 12 may need to process, for example, 10, 100, or any other number of packets to classify the flow as Web Ex traffic. Once the classification is performed, the controller 12 sends the stateless DPI rules and flow information to the AP 14 for stateless sub-classification to distinguish voice, video, or data within a WebEx flow. For example, after the controller 12 identifies the WebEx meeting traffic, it pushes the tuple, the stateless DPI rules (as shown below), and policies to the AP 14 for upstream traffic marking, dropping, or rate-limit-ing. If the client 16 roams, the controller 12 transmits the same classification information to the new AP to which the client has roamed.")<br><br>Chandrasekaran at [0035]-[0044] ("WebEx Video:<br>UDP Payload<br>First byte=0x06<br>Bytes [6-9]=Data length<br>10th byte=0x50<br><br>WebEx Voice:<br>UDP Payload<br>First byte=0x06<br>Bytes [6-9]=Data length<br>10th byte=0x48")<br><br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Chandrasekaran in combination with (1) the knowledge of a person of ordinary skill in the |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 455 of 1100

| No. | '111 Patent Claim 16 | Chandrasekaran |
|---|---|---|
| | | art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 16[b] of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references.<br><br>For example, Copeland discloses packet headers with flag bits.<br><br>Copeland at Figure 2<br><br>  <br><br>PACKET HEADERS<br>FIG. 2 |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 456 of 1100

| No. | '111 Patent Claim 16 | Chandrasekaran |
|---|---|---|
| | | Copeland at [0076] ("FIG. 2 illustrates an exemplary TCP/IP packet or datagram 210 and an exemplary UDP datagram 240. In a typical TCP/IP packet like 210, each packet typically includes a header portion comprising an IP header 220 and a TCP header 230, followed by a data portion that contains the information to be communicated in the packet. The information in the IP header 220 contained in a TCP/IP packet 210, or any other IP packet, contains the IP addresses and assures that the packet is delivered to the right host. The transport layer protocol (TCP) header follows the Internet protocol header and specifies the port numbers for the associated service.")<br><br>Copeland at [0077] ("The header portion in the typical TCP/IP datagram 210 is 40 bytes including 20 bytes of IP header 220 information and 20 bytes of TCP header 230 information. The data portion or segment associated with the packet 210 follows the header information.")<br><br>Copeland at [0078] ("In regards to a typical IP packet 210, the first 4 bits of the IP header 220 identify the Internet protocol (IP) version. The following 4 bits identify the IP header length in 32 bit words. The next 8 bits differentiate the type of service by describing how the packet should be handled in transit. The following 16 bits convey the total packet length.")<br><br>Copeland at [0081] ("In a TCP/IP datagram 210, the initial data of the IP datagram is the TCP header 230 information. The initial TCP header 230 information includes the 16-bit source and 16-bit destination port numbers. A 32-bit sequence number for the data in the packet follows the port numbers. Following the sequence number is a 32-bit acknowledgement number. If an ACK flag (discussed below) is set, this number is the next sequence number the sender of the packet expects to receive. Next is a 4-bit data offset, which is the number of 32-bit words in the TCP header. A 6-bit reserved field follows.")<br><br>Copeland at [0082] ("Following the reserved field, the next 6 bits are a series of one-bit flags, shown in FIG. 2 as flags U, A, P, R, S, F. The first flag is the urgent flag (U). If the U flag is set, it indicates that the urgent pointer is valid and points to urgent data that should be acted upon as soon as possible. The next flag is the A ( or ACK or "acknowledgment") flag. The ACK flag indicates that an acknowledgment number is valid, and acknowledges that data has been received. The next flag, the push (P) flag, tells the receiving end to push all |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 457 of 1100

| No. | '111 Patent Claim 16 | Chandrasekaran |
|---|---|---|
| | | buffered data to the receiving application. The reset (R) flag is the following flag, which terminates both ends of the TCP connection. Next, the S (or SYN for "synchronize") flag is set in the initial packet of a TCP connection where both ends have to synchronize their TCP buffers. Following the SYN flag is the F (for FIN or "finish") flag. This flag signifies that the sending end of the communication and the host will not send any more data but still may acknowledge data that is received.")<br><br>Copeland at [0083] ("Following the TCP flag bits is a 16-bit receive window size field that specifies the amount of space avail-able in the receive buffer for the TCP connection. The checksum of the TCP header is a 16-bit field. Following the checksum is a 16 bit urgent pointer that points to the urgent data. The TCP/IP datagram data follows the TCP header.")<br><br>Copeland at [0116] ("These steps generally require manipulations of quantities such as IP addresses, packet length, header length, start times, end times, port numbers, and other packet related information. Usually, though not necessarily, these quanti-ties take the form of electrical, magnetic, or optical signals capable of being stored, transferred, combined, compared, or otherwise manipulated. It is conventional for those skilled in the art to refer to these signals as bits, bytes, words, values, elements, symbols, characters, terms, numbers, points, records, objects, images, files or the like. It should be kept in mind, however, that these and similar terms should be associated with appropriate quantities for computer opera-tions and that these terms are merely conventional labels applied to quantities that exist within and during operation of the computer.")<br><br>As another example, Kempf discloses packet headers with flag bits.<br><br>Kempf at [0081] ("In one embodiment, OpenFlow is modified to pro-vide rules for GTP TEID Routing. FIG. 17 is a diagram of one embodiment of the OpenFlow flow table modification for GTP TEID routing. An OpenFlow switch that supports TEID routing matches on the 2 byte (16 bit) collection of header fields and the 4 byte (32 bit) GTP TEID, in addition to other OpenFlow header fields, in at least one flow table ( e.g., the first flow table). The GTP TEID flag can be wildcarded (i.e. matches are "don't care"). In one embodiment, the EPC pro-tocols do not assign any meaning to TEIDs other than as an endpoint identifier for tunnels, like ports in standard UDP/ TCP transport protocols. In other embodiments, the TEIDs can have a correlated meaning or semantics. The GTP header flags |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 458 of 1100

| No. | '111 Patent Claim 16 | Chandrasekaran |
|---|---|---|
| | | field can also be wildcarded, this can be partially matched by combining the following bitmasks: 0xFF00- Match the Message Type field; 0xe0-Match the Version field; 0xl0-Match the PT field; 0x04-Match the E field; 0x02- Match the S field; and 0x0l-Match the PN field.")<br><br>Kempf at [0082] ("In one embodiment, OpenFlow can be modified to support virtual ports for fast path GTP TEID encapsulation and decapsulation. An OpenFlow mobile gateway can be used to support GTP encapsulation and decapsulation with virtual ports. The GTP encapsulation and decapsulation virtual ports can be used for fast encapsulation and decapsulation of user data packets within GTP-U tunnels, and can be designed simply enough that they can be implemented in hardware or firmware. For this reason, GTP virtual ports may have the following restrictions on traffic they will handle: Protocol Type (PT) field= 1, where GTP encapsulation ports only sup-port GTP, not GTP' (PT field=0); Extension Header flag (E)=0, where no extension headers are supported, Sequence Number flag (S)=0, where no sequence numbers are sup-ported; N-PDU flag (PN)=0; and Message type=255, where Only G-PDU messages, i.e. tunneled user data, is supported in the fast path.")<br><br>Kempf at [0083] ("If a packet either needs encapsulation or arrives encapsulated with nonzero header flags, header extensions, and/or the GTP-U packet is not a G-PDU packet (i.e. it is a GTP-U control packet), the processing must proceed via the gateway's slow path (software) control plane. GTP-C and GTP' packets directed to the gateway's IP address are a result of mis-configuration and are in error. They must be sent to the OpenFlow controller, since these packets are handled by the S-GW-C and P-GW-C control plane entities in the cloud computing system or to the billing entity handling GTP' and not the S-GW-D and P-GW-D data plane switches.")<br><br>Kempf at [0088] ("To support slow path encapsulation, the software control plane on the switch maintains a hash table with keys calculated from the GTP-U TEID. The TEID hash keys are calculated using a suitable hash algorithm with low collision frequency, for example SHA-1. The flow table entries contain a record of how the packet header, including the GTP encap-sulation header, should be configured. This includes: the same header fields as for the hardware or firmware encapsu-lation table in FIG.18; values for the GTP header flags (PT, E, S, and PN); the sequence number and/or the N-PDU number if any; if the E |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 459 of 1100

| No. | '111 Patent Claim 16 | Chandrasekaran |
|---|---|---|
| | | flag is 1, then the flow table contains a list of the extension headers, including their types, which the slow path should insert into the GTP header.")<br><br>Kempf at [0092] ("In one embodiment, the system implements a GTP fast path decapsulation virtual port. When requested by the S-GW and P-GW control plane software running in the cloud computing system, the gateway switch installs rules and actions for routing GTP encapsulated packets out of GTP tunnels. The rules match the GTP header flags and the GTP TEID for the packet, in the modified OpenFlow flow table shown in FIG. 17 as follows: the IP destination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); the UDP destination port is the GTP-U destination port (2152); and the header fields and message type field is wildcarded with the flag 0XFFF0 and the upper two bytes of the field match the G-PDU message type (255) while the lower two bytes match 0x30, i.e. the packet is a GTP packet not a GTP' packet and the version number is 1.")<br><br>Kempf at [0094] ("In one embodiment, the system implements han-dling of GTP-U control packets. The OpenFlow controller programs the gateway switch flow tables with 5 rules for each gateway switch IP address used for GTP traffic. These rules contain specified values for the following fields: the IP des-tination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); the UDP destination port is the GTP-U destination port (2152); the GTP header flags and message type field is wildcarded with 0xFFF0; the value of the header flags field is 0x30, i.e. the version number is 1 and the PT field is 1; and the value of the message type field is one of 1 (Echo Request), 2 (Echo Response), 26 (Error Indication), 31 (Support for Extension Headers Notification), or 254 (End Marker).")<br><br>Kempf at [0098] ("The header flags and message type fields for the three rules are wildcarded with the following bitmasks and match as follows: bitmask 0xFFF4 and the upper two bytes match the G-PDU message type (255) while the lower two bytes are Ox34, indicating that the version number is 1, the packet is a GTP packet, and there is an extension header present; bitmask 0xFFFF2 and the upper two bytes match the G-PDU message type (255) while the lower two bytes are 0x32, indicating that the version number is 1, the packet is a GTP packet, and there is a sequence number present; and bitmask 0xFF0l and the upper two bytes match the G-PDU message type (255) while the lower two bytes are 0x31, |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 460 of 1100

| No. | '111 Patent Claim 16 | Chandrasekaran |
|---|---|---|
| | | indicating that the version number is 1, the packet is a GTP packet, and a N-PDU is present.")<br><br>Kempf at [0114] ("The gtp_type_n_flags field contains the GTP mes-sage type in the upper 8 bits and the GTP header flags in the lower 8 bits. The gtp_teid field contains the GTP TEID. The gtp_ wildcard field indicates whether the GTP type and flags and TEID should be matched. If the lower four bits are 1, the type and flags field should be ignored, while if the upper four bits are 1, the TEID should be ignored. If the lower bits are 0, the type and fields flag should be matched subject to the flags in the gtp_flag_mask field, while if the upper bits are 0 the TEID should be matched. The mask is combined with the message type and header field of the packet using logical AND; the result becomes the value of the match. Only those parts of the field in which the mask has a 1 value are matched.")<br><br>Kempf at [0117] ("The gtp_type_n_flags field contains the GTP mes-sage type in the upper 8 bits and the GTP header flags in the lower 8 bits. The gtp_<br>teid field contains the GRP TEID. When the value of the oxm_type ( oxm_class+oxm_field is GTP _ MATCH and the HM bit is zero, the flaw's GTP header must match these values exactly. If the HM flag is one, the value contains an ersmt_gtp_match field and an ermst_gtp_mask field, as specified by the OpenFlow 1.2 specification. We define ermst_gtp_mask field for selecting flows based on the settings of flag bits:<br><br>struct ermst_gtp_mask {<br>uint32_t gtp_wildcard;<br>uint16_t gtp_flag_mask;<br>};<br><br>Kempf at [0118] ("The gtp_ wildcard field indicates whether the TEID should be matched. If the value is 0xFFFFFFFF, the TEID should be matched and not the flags, if the value is 0x00000000, the flags should be matched and not the TEID. If the gtp_ wildcard indicates the flags should be matched, the gtp_flag_mask is combined with the message type and header field of the packet using logical AND, the result becomes the value of the match. Only those parts of the field in which the mask has a 1 value are matched.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 461 of 1100

| No. | '111 Patent Claim 16 | Chandrasekaran |
|---|---|---|
| 16[c] | wherein the packet-applicable criterion is that one or more of the flag bits is set. | Chandrasekaran discloses wherein the packet-applicable criterion is that one or more of the flag bits is set.<br><br>For example, Chandrasekaran discloses packets with header fields. A person of ordinary skill in the art would understand that the header could be comprised of one or more flag bits. A person of ordinary skill in the art would further understand that such flag bits in packet headers could be the packet specific information. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Chandrasekaran is found to not meet this limitation, wherein the packet applicable criterion is that one or more of the flag bits is set would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>Chandrasekaran at [0014] ("The term 'wireless controller' or 'controller' as used herein may refer to a wireless LAN (local area network) controller, mobility controller, wireless control device, wire-less control system, or any other network device operable to perform control functions for a wireless network. The net-work site may also include a wireless control system or other platform for centralized wireless LAN planning, configura-tion, and management. The wireless controller 12 enables system wide functions for wireless applications and may support any number of access points 14. Each access point 14 may serve any number of mobile devices 16 in the wireless network. The wireless controller 12 may be, for example, a standalone device or a rack-mounted appliance. In the example shown in FIG. 1, the wireless controller 12 and access points 14 are separate devices and may be located remote from one another. The wireless controller 12 may also be integrated with the access point 14 ( e.g., autonomous AP) or located at a switch, router, switch/router, or other network device. Thus, the wireless controller 12 may be a physical device located at a standalone device, access point, switch, router, or other network device. The wireless controller 12 may also be a virtual device located in a network or cloud, for example.")<br><br>Chandrasekaran at [0016] ("The wireless controller 12 includes a stateful appli-cation classifier 18 and the AP 14 includes a stateless appli-cation classifier 22. After the stateful classifier 18 identifies the application, the controller 12 transmits ( e.g., pushes) clas-sification information 26 to the AP 14 so that the AP can perform stateless classification and apply policies ( e.g., QoS or other policies) to traffic received from the |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 462 of 1100

| No. | '111 Patent Claim 16 | Chandrasekaran |
|---|---|---|
| | | mobile device 16. The controller 12 may also provide the classification information 26 to another AP 14 if the client 16 roams to a new AP, as shown in FIG. 1. Implementation of the stateful classifier 18 at the controller 12 and stateless classifier 22 at the AP 14 allows for policies to be applied for downstream traffic (packet 25) at the wireless controller 12, and for upstream traffic (packet 28) at the access point 14.")<br><br>Chandrasekaran at [0020] ("In one embodiment, the stateful classifier 18 is a classification engine configured for NBAR (Network Based Application Recognition) or other technology used to classify applications. The classifier 18 is operable to recognize a wide variety of applications, including Web-based and client/ server applications. The applications may include, for example, Skype, YouTube, Netflix, WebEx, Google Voice, BitTorrent, Citrix, virtual desktop, PCoIP, or any other appli-cation. The classification engine may be configured, for example, to identify generic protocols and perform heuristic analysis for encrypted protocols. The classifiers 18, 22 are configured to perform deep packet inspection (DPI), which provides the ability to look into the packet past basic header information so that the contents of a particular packet can be determined.")<br><br>Chandrasekaran at [0021] ("Once the application is recognized, QoS or other policies associated with the application can be applied to traffic so that the network can invoke services for that par-ticular application. For example, the application may have certain requirements and expectations from the network infrastructure, which may be specified in terms of bandwidth, delay, jitter, throughput, packet loss, or other performance attributes.")<br><br>Chandrasekaran at [0023] ("In one embodiment, the classification information 26 transmitted from the controller 12 to the AP 14 includes tuple information for a flow ( e.g., source IP address, destina-tion IP address, source port, destination port, and protocol), application identifier (ID), and stateless DPI information. Stateless DPI information includes classification and sub-classification information ( e.g., fixed or variable offset with a pattern or regular expression) and rules for applying policies on the sub-classified packets. The policies may include, for example, drop packet, mark a DSCP (Differentiated Services Code Point) value in the packet, or rate limit the traffic.")<br><br>Chandrasekaran at [0026] ("Memory 34 may be a volatile memory or non-vola-tile storage, which stores various applications, operating sys-tems, modules, and data for execution and |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 463 of 1100

| No. | '111 Patent Claim 16 | Chandrasekaran |
|---|---|---|
| | | use by the proces-sor 32. Memory 34 may include, for example, classification database 35. The classification database 35 may be any data structure configured for at least temporarily storing classifi-cation information including, for example, flow information, application ID, stateless DPI rules, and policies.")<br><br>Chandrasekaran at [0031] ("FIG. 3 is a flowchart illustrating an example of a process at the controller 12 for classification of traffic for application aware policies in a wireless network, in accor-dance with one embodiment. At step 40, the controller 12 receives packets belonging to a network flow. The controller 12 performs stateful classification to identify an application associated with the flow ( step 42). The controller 12 transmits classification information ( e.g., flow information, stateless DPI rule, and policy) to the AP 14 for use in stateless classi-fication at the AP (step 44). The controller 12 applies policies to downstream traffic (received at the controller and destined for the client 16) (step 46) and receives upstream traffic for which policies have been applied at the AP 14 (step 48). If the controller 12 determines ( e.g., receives an indication) that the client 16 has roamed, it transmits the classification informa-tion to the new AP 14 to which the client has roamed (steps 50 and 52).")<br><br>Chandrasekaran at [0033] ("The following describes an example of the above process for WebEx traffic that has different sub-classifications for voice and video traffic. Stateful classification is first performed by the controller 12 at the beginning of the flow. The controller 12 may need to process, for example, 10, 100, or any other number of packets to classify the flow as Web Ex traffic. Once the classification is performed, the controller 12 sends the stateless DPI rules and flow information to the AP 14 for stateless sub-classification to distinguish voice, video, or data within a WebEx flow. For example, after the controller 12 identifies the WebEx meeting traffic, it pushes the tuple, the stateless DPI rules (as shown below), and policies to the AP 14 for upstream traffic marking, dropping, or rate-limit-ing. If the client 16 roams, the controller 12 transmits the same classification information to the new AP to which the client has roamed.")<br><br>Chandrasekaran at [0035]-[0044] ("WebEx Video:<br>UDP Payload<br>First byte=0x06<br>Bytes [6-9]=Data length |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 464 of 1100

| No. | '111 Patent Claim 16 | Chandrasekaran |
|---|---|---|
| | | 10th byte=0x50<br><br>WebEx Voice:<br>UDP Payload<br>First byte=0x06<br>Bytes [6-9]=Data length<br>10th byte=0x48")<br><br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Chandrasekaran in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 16[c] of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references.<br><br>For example, Copeland discloses packet specific characteristics including flag bits that are set.<br><br>Copeland at [0081] ("In a TCP/IP datagram 210, the initial data of the IP datagram is the TCP header 230 information. The initial TCP header 230 information includes the 16-bit source and 16-bit destination port numbers. A 32-bit sequence number for the data in the packet follows the port numbers. Following the sequence number is a 32-bit acknowledgement number. If an ACK flag (discussed below) is set, this number is the next sequence number the sender of the packet expects to receive. Next is a 4-bit data offset, which is the number of 32-bit words in the TCP header. A 6-bit reserved field follows.")<br><br>Copeland at [0082] ("Following the reserved field, the next 6 bits are a series of one-bit flags, shown in FIG. 2 as flags U, A, P, R, S, F. The first flag is the urgent flag (U). If the U flag is set, it indicates that the urgent pointer is valid and points to urgent data that should be acted upon as soon as possible. The next flag is the A ( or ACK or "acknowledgment") flag. The ACK flag indicates that an acknowledgment number is valid, and acknowledges that data has been received. The next flag, the push (P) flag, tells the receiving end to push all buffered data to the receiving application. The reset (R) flag is the following flag, which terminates both ends of the TCP connection. Next, the S (or SYN for "synchronize") flag is set in the initial packet of a TCP connection where both ends have to synchronize their TCP |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 465 of 1100

| No. | '111 Patent Claim 16 | Chandrasekaran |
|---|---|---|
| | | buffers. Following the SYN flag is the F (for FIN or "finish") flag. This flag signifies that the sending end of the communication and the host will not send any more data but still may acknowledge data that is received.")<br><br>Copeland at [0083] ("Following the TCP flag bits is a 16-bit receive window size field that specifies the amount of space avail-able in the receive buffer for the TCP connection. The checksum of the TCP header is a 16-bit field. Following the checksum is a 16 bit urgent pointer that points to the urgent data. The TCP/IP datagram data follows the TCP header.")<br><br>Copeland at [0089] ("FIG. 3 illustrates an exemplary TCP/IP session 300. As discussed in reference to FIG. 2, the SYN flag is set whenever one host initiates a session with another host. In the initial packet, Hostl sends a message with only the SYN flag set. The SYN flag is designed to establish a TCP connection and allow both ends to synchronize their TCP buffers. Hostl provides the sequence of the first data packet it will send.")<br><br>Copeland at [0125] ("For purposes of the description, which follows, the IP address with the lower value, when considered as a 32-bit unsigned integer, is designated ip[0] and the corresponding port number is designated pt[0]. The higher IP address is designated ip[l] and the corresponding TCP or UDP port number is designated pt[l]. At some point, either pt[0] or pt[l] may be designated the "server" port by setting an appropriate bit in a bit map that is part of the flow record (record "state", bit 1 or 2 is set).")<br><br>Copeland at [0145] ("A list IP of addresses contacted or probed by each host can be maintained. When this list indicates that more than a threshold number of other hosts (e.g., 8) have been contacted in the same subnet, CI is added to the to the host and a bit in the host record is set to indicate that the host has received CI for "address scanning." Note that the number of hosts to designate a scan is not required to be a fixed value, but could be adjusted based on the sample rate or other means to enhance the accuracy making the number of hosts scanned "statistically significant". These and other values of concern index are shown for non-flow based events in FIG. 7.")<br><br>As another example, Kempf flow table matches in which the flag bits is set. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 466 of 1100

| No. | '111 Patent Claim 16 | Chandrasekaran |
|---|---|---|
| | | Kempf at [0081] ("In one embodiment, OpenFlow is modified to pro-vide rules for GTP TEID Routing. FIG. 17 is a diagram of one embodiment of the OpenFlow flow table modification for GTP TEID routing. An OpenFlow switch that supports TEID routing matches on the 2 byte (16 bit) collection of header fields and the 4 byte (32 bit) GTP TEID, in addition to other OpenFlow header fields, in at least one flow table ( e.g., the first flow table). The GTP TEID flag can be wildcarded (i.e. matches are "don't care"). In one embodiment, the EPC pro-tocols do not assign any meaning to TEIDs other than as an endpoint identifier for tunnels, like ports in standard UDP/ TCP transport protocols. In other embodiments, the TEIDs can have a correlated meaning or semantics. The GTP header flags field can also be wildcarded, this can be partially matched by combining the following bitmasks: 0xFF00- Match the Message Type field; 0xe0-Match the Version field; 0xl0-Match the PT field; 0x04-Match the E field; 0x02- Match the S field; and 0x0l-Match the PN field.")<br><br>Kempf at [0082] ("In one embodiment, OpenFlow can be modified to support virtual ports for fast path GTP TEID encapsulation and decapsulation. An OpenFlow mobile gateway can be used to support GTP encapsulation and decapsulation with virtual ports. The GTP encapsulation and decapsulation virtual ports can be used for fast encapsulation and decapsulation of user data packets within GTP-U tunnels, and can be designed simply enough that they can be implemented in hardware or firmware. For this reason, GTP virtual ports may have the following restrictions on traffic they will handle: Protocol Type (PT) field= 1, where GTP encapsulation ports only sup-port GTP, not GTP' (PT field=0); Extension Header flag (E)=0, where no extension headers are supported, Sequence Number flag (S)=0, where no sequence numbers are sup-ported; N-PDU flag (PN)=0; and Message type=255, where Only G-PDU messages, i.e. tunneled user data, is supported in the fast path.")<br><br>Kempf at [0083] ("If a packet either needs encapsulation or arrives encapsulated with nonzero header flags, header extensions, and/or the GTP-U packet is not a G-PDU packet (i.e. it is a GTP-U control packet), the processing must proceed via the gateway's slow path (software) control plane. GTP-C and GTP' packets directed to the gateway's IP address are a result of mis-configuration and are in error. They must be sent to the OpenFlow controller, since these packets are handled by the S-GW-C and P-GW-C control plane entities in the |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 467 of 1100

| No. | '111 Patent Claim 16 | Chandrasekaran |
|---|---|---|
| | | cloud computing system or to the billing entity handling GTP' and not the S-GW-D and P-GW-D data plane switches.")<br><br>Kempf at [0088] ("To support slow path encapsulation, the software control plane on the switch maintains a hash table with keys calculated from the GTP-U TEID. The TEID hash keys are calculated using a suitable hash algorithm with low collision frequency, for example SHA-1. The flow table entries contain a record of how the packet header, including the GTP encap-sulation header, should be configured. This includes: the same header fields as for the hardware or firmware encapsu-lation table in FIG.18; values for the GTP header flags (PT, E, S, and PN); the sequence number and/or the N-PDU number if any; if the E flag is 1, then the flow table contains a list of the extension headers, including their types, which the slow path should insert into the GTP header.")<br><br>Kempf at [0092] ("In one embodiment, the system implements a GTP fast path decapsulation virtual port. When requested by the S-GW and P-GW control plane software running in the cloud computing system, the gateway switch installs rules and actions for routing GTP encapsulated packets out of GTP tunnels. The rules match the GTP header flags and the GTP TEID for the packet, in the modified OpenFlow flow table shown in FIG. 17 as follows: the IP destination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); the UDP destination port is the GTP-U destination port (2152); and the header fields and message type field is wildcarded with the flag 0XFFF0 and the upper two bytes of the field match the G-PDU message type (255) while the lower two bytes match 0x30, i.e. the packet is a GTP packet not a GTP' packet and the version number is 1.")<br><br>Kempf at [0094] ("In one embodiment, the system implements han-dling of GTP-U control packets. The OpenFlow controller programs the gateway switch flow tables with 5 rules for each gateway switch IP address used for GTP traffic. These rules contain specified values for the following fields: the IP des-tination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); the UDP destination port is the GTP-U destination port (2152); the GTP header flags and message type field is wildcarded with 0xFFF0; the value of the header flags field is 0x30, i.e. the version number is 1 and the PT field is 1; and the value of the message type field is one of 1 (Echo Request), 2 (Echo |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 468 of 1100

| No. | '111 Patent Claim 16 | Chandrasekaran |
|---|---|---|
| | | Response), 26 (Error Indication), 31 (Support for Extension Headers Notification), or 254 (End Marker).")<br><br>Kempf at [0098] ("The header flags and message type fields for the three rules are wildcarded with the following bitmasks and match as follows: bitmask 0xFFF4 and the upper two bytes match the G-PDU message type (255) while the lower two bytes are Ox34, indicating that the version number is 1, the packet is a GTP packet, and there is an extension header present; bitmask 0xFFFF2 and the upper two bytes match the G-PDU message type (255) while the lower two bytes are 0x32, indicating that the version number is 1, the packet is a GTP packet, and there is a sequence number present; and bitmask 0xFF0l and the upper two bytes match the G-PDU message type (255) while the lower two bytes are 0x31, indicating that the version number is 1, the packet is a GTP packet, and a N-PDU is present.")<br><br>Kempf at [0114] ("The gtp_type_n_flags field contains the GTP mes-sage type in the upper 8 bits and the GTP header flags in the lower 8 bits. The gtp_teid field contains the GTP TEID. The gtp_ wildcard field indicates whether the GTP type and flags and TEID should be matched. If the lower four bits are 1, the type and flags field should be ignored, while if the upper four bits are 1, the TEID should be ignored. If the lower bits are 0, the type and fields flag should be matched subject to the flags in the gtp_flag_mask field, while if the upper bits are 0 the TEID should be matched. The mask is combined with the message type and header field of the packet using logical AND; the result becomes the value of the match. Only those parts of the field in which the mask has a 1 value are matched.")<br><br>Kempf at [0117] ("The gtp_type_n_flags field contains the GTP mes-sage type in the upper 8 bits and the GTP header flags in the lower 8 bits. The gtp_teid field contains the GRP TEID. When the value of the oxm_type ( oxm_class+oxm_field is GTP _ MATCH and the HM bit is zero, the flaw's GTP header must match these values exactly. If the HM flag is one, the value contains an ersmt_gtp_match field and an ermst_gtp_mask field, as specified by the OpenF!ow 1.2 specification. We define ermst_gtp_mask field for selecting flows based on the settings of flag bits: |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 469 of 1100

| No. | '111 Patent Claim 16 | Chandrasekaran |
|---|---|---|

```
struct ernst_gtp_mask {
  uint32_t gtp_wildcard;
  uint16_t gtp_flag_mask;
};
```

Kempf at [0118] ("The gtp_ wildcard field indicates whether the TEID should be matched. If the value is 0xFFFFFFFF, the TEID should be matched and not the flags, if the value is 0x00000000, the flags should be matched and not the TEID. If the gtp_ wildcard indicates the flags should be matched, the gtp_flag_mask is combined with the message type and header field of the packet using logical AND, the result becomes the value of the match. Only those parts of the field in which the mask has a 1 value are matched.")

Kempf at Figure 10

| Octets | Bits 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
|---|---|---|---|---|---|---|---|---|
| 1 | Version | | | PT | (*) | E | S | PN |
| 2 | Message Type | | | | | | | |
| 3 | Length (1st Octet) | | | | | | | |
| 4 | Length (2nd Octet) | | | | | | | |
| 5 | Tunnel Endpoint Identifier (1st Octet) | | | | | | | |
| 6 | Tunnel Endpoint Identifier (2nd Octet) | | | | | | | |
| 7 | Tunnel Endpoint Identifier (3rd Octet) | | | | | | | |
| 8 | Tunnel Endpoint Identifier (4th Octet) | | | | | | | |
| 9 | Sequence Number (1st Octet) | | | | | | | |
| 10 | Sequence Number (2nd Octet) | | | | | | | |
| 11 | N-PDU Number | | | | | | | |
| 12 | Next Extension Header Type | | | | | | | |

NOTE 0: (*) This bit is a spare bit. It shall be sent as '0'. The receiver shl not evaluate this bit,
NOTE 1: 1) This field shall only be evaluated when indicated by the S flag set to 1.
NOTE 2: 2) This field shall only be evaluated when indicated by the PN flag set to 1.
NOTE 3: 3) This field shall only be evaluated when indicated by the E flag set to 1.
NOTE 4: 4) This field shall be present if and only if any one or more of the S. PN and E flags are set.

FIG. 10



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 470 of 1100

| No. | '111 Patent Claim 16 | Chandrasekaran |
|---|---|---|
| | | |

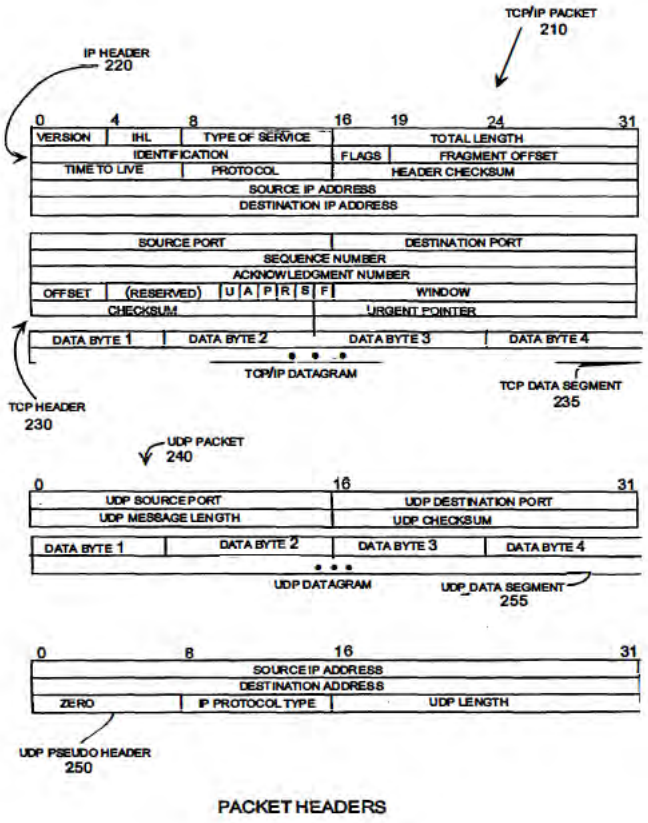
| No. | '111 Patent Claim 17 | Chandrasekaran |
|---|---|---|
| 17[a] | The method according to claim 16, wherein the packet is an Transmission Control Protocol (TCP) packet, and | Chandrasekaran discloses the method according to claim 16, wherein the packet is an Transmission Control Protocol (TCP) packet.<br><br>For example, Chandrasekaran discloses data traffic comprised of packets that may be a Transmission Control Protocol packet. A person of ordinary skill in the art would understand that the packets may be part of a number of protocols, including Transmission Control Protocol. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Chandrasekaran is found to not meet this limitation, wherein the packet is an Transmission Control Protocol (TCP) packet would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>*See supra* at Claim 16.<br><br>Chandrasekaran at [0012] ("Referring now to the drawings, and first to FIG.1, an example of a network in which embodiments described herein may be implemented is shown. For simplification, only a small number of network devices are shown. The network includes a wireless controller 12 in communication with a mobile device (client, wireless device, endpoint) 16 through an access point (AP) 14. In the example shown in FIG. 1, the controller 12 is in wired communication with two access points 14 for wireless communication with any number of mobile devices 16 via a wireless network ( e.g., WLAN (wire-less local area network)) at a network site. The wireless con-troller 12 may be in communication with one or more other networks (not shown) (e.g., Internet, intranet, local area net-work, wireless local area network, cellular network, metro-politan area network, wide area network, satellite network, radio access network, public switched network, virtual pri-vate network, or any other network or combination thereof). Communication paths between the wireless controller 12 and other networks or between the controller and access points 14 may include any number or type of intermediate nodes (e.g., routers, switches, |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 471 of 1100

| No. | '111 Patent Claim 17 | Chandrasekaran |
|---|---|---|
| | | gateways, or other network devices), which facilitate passage of data between network devices.")<br><br>Chandrasekaran at [0017] ("The stateful classifier 18 at the controller 12 classi-fies traffic based on multiple packets received from the begin-ning of a flow. Stateful classification uses rules which need information on states for a previous packet ( or packets) in a flow. Stateful classification may be based, for example, on packet pattern matching and decoding of protocols and their states. Stateful classification is also referred to as flow clas-sification since it looks at a data stream of related packets (flow, session).")<br><br>Chandrasekaran at [0018] ("The stateless classifier 22 at the AP 14 uses rules that can act on a per packet basis in the flow. Stateless classifica-tion (also referred to as packet classification) is based on individual packet inspection ( e.g., 5 tuple, pattern matching) without knowledge of any related stream of packets, flows, sessions, or protocols.")<br><br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Chandrasekaran in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 17(a) of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references.<br><br>For example, Copeland discloses TCP packets.<br><br>Copeland at Figure 2 |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 472 of 1100

| No. | '111 Patent Claim 17 | Chandrasekaran |
|---|---|---|
| | |  <br>FIG. 2<br><br>Copeland at [0076] ("FIG. 2 illustrates an exemplary TCP/IP packet or datagram 210 and an exemplary UDP datagram 240. In a typical TCP/IP packet like 210, each packet typically includes a header portion comprising an IP header 220 and a TCP header 230, followed by a data portion that contains the information to be communicated in the packet. The information in the IP header 220 contained in a TCP/IP packet 210, or any other IP packet, contains the IP addresses and assures that the packet is delivered to the right host. The transport layer protocol (TCP) header follows the Internet protocol header and specifies the port numbers for the associated service.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 473 of 1100

| No. | '111 Patent Claim 17 | Chandrasekaran |
|---|---|---|
| | | Copeland at [0077] ("The header portion in the typical TCP/IP datagram 210 is 40 bytes including 20 bytes of IP header 220 information and 20 bytes of TCP header 230 information. The data portion or segment associated with the packet 210 follows the header information.")<br><br>For example, Chua discloses packet traffic using TCP services.<br><br>Chua at 18:8-44 ("Based on the rule set for a specific device, including what traffic is allowed through, from what zones and expected targets, administrator 114 can infer the right type of test traffic that should be injected for both positive and negative testing of the device. Similarly, the inference can be made to deter-mine the appropriate type of flow troubleshooting that includes the appropriate IP ranges to watch on, and the TCP/ UDP services that the flowchain is meant to service.<br><br>In the case of traffic injection to test a path through SDN 106, path verification unit 136 of SDN controller 112 may perform any of the following tasks:<br><br>Infer from the flowchain the appropriate type of transmission control protocol (TCP) or uniform datagram proto-col (UDP) services that are relevant to the chain, as well as the range of source and target IP addresses for this chain<br><br>Create either UDP packet flows or TCP sessions that confirm to the traffic type inferred or discovered form the flowchain configuration<br><br>Using an SDN device ( e.g., a switch of SDN 106), inject these flows or TCP sessions by temporarily inserting a rule into the rules table on the switch that will take the flows from SDN controller 112 and send them across the devices of SDN 106 (starting from the ingress device in the chain)<br><br>Using an SDN device, insert flow rules to replicate (flow-span) these same traffic streams at each point in the flow chain. The replicated traffic may be directed (via SDN rules) to a collection device ( or to SDN controller 112) that can determine whether the test packets passed through each port on the switch that is part of the flowchain |


~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 474 of 1100

| No. | '111 Patent Claim 17 | Chandrasekaran |
|---|---|---|
| | | On the final exit device (prior to the end target), insert a flow rule that will drain all injected packets so the final devices (that is, devices external to SDN 106) do not see any of the test traffic.") |
| 17[b] | wherein the one or more flag bits comprises comprise a SYN flag bit, an ACK flag bit, a FIN flag bit, a RST flag bit, or any combination thereof. | Chandrasekaran discloses wherein the one or more flag bits comprises comprise a SYN flag bit, an ACK flag bit, a FIN flag bit, a RST flag bit, or any combination thereof.<br><br>For example, Chandrasekaran discloses packets with header fields. A person of ordinary skill in the art would understand that the header could be comprised of one or more flag bits that comprise a SYN flag bit, an ACK flag bit, a FIN flag bit, a RST flag bit, or any combination thereof. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Chandrasekaran is found to not meet this limitation, wherein the one or more flag bits comprises comprise a SYN flag bit, an ACK flag bit, a FIN flag bit, a RST flag bit, or any combination thereof would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>Chandrasekaran at [0014] ("The term 'wireless controller' or 'controller' as used herein may refer to a wireless LAN (local area network) controller, mobility controller, wireless control device, wire-less control system, or any other network device operable to perform control functions for a wireless network. The net-work site may also include a wireless control system or other platform for centralized wireless LAN planning, configura-tion, and management. The wireless controller 12 enables system wide functions for wireless applications and may support any number of access points 14. Each access point 14 may serve any number of mobile devices 16 in the wireless network. The wireless controller 12 may be, for example, a standalone device or a rack-mounted appliance. In the example shown in FIG. 1, the wireless controller 12 and access points 14 are separate devices and may be located remote from one another. The wireless controller 12 may also be integrated with the access point 14 ( e.g., autonomous AP) or located at a switch, router, switch/router, or other network device. Thus, the wireless controller 12 may be a physical device located at a standalone device, access point, switch, router, or other network device. The wireless controller 12 may also be a virtual device located in a network or cloud, for example.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 475 of 1100

| No. | '111 Patent Claim 17 | Chandrasekaran |
|---|---|---|
| | | Chandrasekaran at [0016] ("The wireless controller 12 includes a stateful appli-cation classifier 18 and the AP 14 includes a stateless appli-cation classifier 22. After the stateful classifier 18 identifies the application, the controller 12 transmits ( e.g., pushes) clas-sification information 26 to the AP 14 so that the AP can perform stateless classification and apply policies ( e.g., QoS or other policies) to traffic received from the mobile device 16. The controller 12 may also provide the classification information 26 to another AP 14 if the client 16 roams to a new AP, as shown in FIG. 1. Implementation of the stateful classifier 18 at the controller 12 and stateless classifier 22 at the AP 14 allows for policies to be applied for downstream traffic (packet 25) at the wireless controller 12, and for upstream traffic (packet 28) at the access point 14.")<br><br>Chandrasekaran at [0020] ("In one embodiment, the stateful classifier 18 is a classification engine configured for NBAR (Network Based Application Recognition) or other technology used to classify applications. The classifier 18 is operable to recognize a wide variety of applications, including Web-based and client/ server applications. The applications may include, for example, Skype, YouTube, Netflix, WebEx, Google Voice, BitTorrent, Citrix, virtual desktop, PCoIP, or any other appli-cation. The classification engine may be configured, for example, to identify generic protocols and perform heuristic analysis for encrypted protocols. The classifiers 18, 22 are configured to perform deep packet inspection (DPI), which provides the ability to look into the packet past basic header information so that the contents of a particular packet can be determined.")<br><br>Chandrasekaran at [0021] ("Once the application is recognized, QoS or other policies associated with the application can be applied to traffic so that the network can invoke services for that par-ticular application. For example, the application may have certain requirements and expectations from the network infrastructure, which may be specified in terms of bandwidth, delay, jitter, throughput, packet loss, or other performance attributes.")<br><br>Chandrasekaran at [0023] ("In one embodiment, the classification information 26 transmitted from the controller 12 to the AP 14 includes tuple information for a flow ( e.g., source IP address, destina-tion IP address, source port, destination port, and protocol), application identifier (ID), and stateless DPI information. Stateless DPI information includes classification and sub-classification information ( e.g., fixed or variable offset with |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 476 of 1100

| No. | '111 Patent Claim 17 | Chandrasekaran |
|---|---|---|
| | | a pattern or regular expression) and rules for applying policies on the sub-classified packets. The policies may include, for example, drop packet, mark a DSCP (Differentiated Services Code Point) value in the packet, or rate limit the traffic.")<br><br>Chandrasekaran at [0026] ("Memory 34 may be a volatile memory or non-vola-tile storage, which stores various applications, operating sys-tems, modules, and data for execution and use by the proces-sor 32. Memory 34 may include, for example, classification database 35. The classification database 35 may be any data structure configured for at least temporarily storing classifi-cation information including, for example, flow information, application ID, stateless DPI rules, and policies.")<br><br>Chandrasekaran at [0031] ("FIG. 3 is a flowchart illustrating an example of a process at the controller 12 for classification of traffic for application aware policies in a wireless network, in accor-dance with one embodiment. At step 40, the controller 12 receives packets belonging to a network flow. The controller 12 performs stateful classification to identify an application associated with the flow ( step 42). The controller 12 transmits classification information ( e.g., flow information, stateless DPI rule, and policy) to the AP 14 for use in stateless classi-fication at the AP (step 44). The controller 12 applies policies to downstream traffic (received at the controller and destined for the client 16) (step 46) and receives upstream traffic for which policies have been applied at the AP 14 (step 48). If the controller 12 determines ( e.g., receives an indication) that the client 16 has roamed, it transmits the classification informa-tion to the new AP 14 to which the client has roamed (steps 50 and 52).")<br><br>Chandrasekaran at [0033] ("The following describes an example of the above process for WebEx traffic that has different sub-classifications for voice and video traffic. Stateful classification is first performed by the controller 12 at the beginning of the flow. The controller 12 may need to process, for example, 10, 100, or any other number of packets to classify the flow as Web Ex traffic. Once the classification is performed, the controller 12 sends the stateless DPI rules and flow information to the AP 14 for stateless sub-classification to distinguish voice, video, or data within a WebEx flow. For example, after the controller 12 identifies the WebEx meeting traffic, it pushes the tuple, the stateless DPI rules (as shown below), and policies to the AP 14 for upstream traffic marking, dropping, or |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 477 of 1100

| No. | '111 Patent Claim 17 | Chandrasekaran |
|---|---|---|
| | | rate-limit-ing. If the client 16 roams, the controller 12 transmits the same classification information to the new AP to which the client has roamed.")<br><br>Chandrasekaran at [0035]-[0044] ("WebEx Video:<br>UDP Payload<br>First byte=0x06<br>Bytes [6-9]=Data length<br>10th byte=0x50<br><br>WebEx Voice:<br>UDP Payload<br>First byte=0x06<br>Bytes [6-9]=Data length<br>10th byte=0x48")<br><br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Chandrasekaran in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 17[b] of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references.<br><br>For example, Copeland discloses TCP packets with flag bits including SYN, ACK, FIN, and R flag bits.<br><br>Copeland at [0081] ("In a TCP/IP datagram 210, the initial data of the IP datagram is the TCP header 230 information. The initial TCP header 230 information includes the 16-bit source and 16-bit destination port numbers. A 32-bit sequence number for the data in the packet follows the port numbers. Following the sequence number is a 32-bit acknowledgement number. If an ACK flag (discussed below) is set, this number is the next sequence number the sender of the packet expects to receive. Next is a 4-bit data offset, which is the number of 32-bit words in the TCP header. A 6-bit reserved field follows.")<br><br>Copeland at [0082] ("Following the reserved field, the next 6 bits are a series of one-bit flags, shown in FIG. 2 as flags U, A, P, R, S, F. The first flag is the urgent flag (U). If the U |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 478 of 1100

| No. | '111 Patent Claim 17 | Chandrasekaran |
|---|---|---|
| | | flag is set, it indicates that the urgent pointer is valid and points to urgent data that should be acted upon as soon as possible. The next flag is the A ( or ACK or "acknowledgment") flag. The ACK flag indicates that an acknowledgment number is valid, and acknowledges that data has been received. The next flag, the push (P) flag, tells the receiving end to push all buffered data to the receiving application. The reset (R) flag is the following flag, which terminates both ends of the TCP connection. Next, the S (or SYN for "synchronize") flag is set in the initial packet of a TCP connection where both ends have to synchronize their TCP buffers. Following the SYN flag is the F (for FIN or "finish") flag. This flag signifies that the sending end of the communication and the host will not send any more data but still may acknowledge data that is received.")<br><br>Copeland at [0089] ("FIG. 3 illustrates an exemplary TCP/IP session 300. As discussed in reference to FIG. 2, the SYN flag is set whenever one host initiates a session with another host. In the initial packet, Hostl sends a message with only the SYN flag set. The SYN flag is designed to establish a TCP connection and allow both ends to synchronize their TCP buffers. Hostl provides the sequence of the first data packet it will send.")<br><br>Copeland at [0090] ("Host2 responds with a SYN-ACK packet. In this message, both the SYN flag and the ACK flag are set. Host2 provides the initial sequence number for its data to Hostl. Host2 also sends to Hostl the acknowledgment number that is the next sequence number Host2 expects to receive from host 1. In the SYN-ACK packet sent by Host2, the acknowl-edgment number is the initial sequence number of Hostl plus 1, which should be the next sequence number received.")<br><br>Copeland at [0091] ("Hostl responds to the SYN-ACK with a packet with just the ACK flag set. Hostl acknowledges that the next packet of information received from Host2 will be Host2's initial sequence number plus 1. The three-way handshake is complete and data is transferred.")<br><br>Copeland at [0092] ("Host2 responds to ACK packet with its own ACK packet. Host2 acknowledges the data it has received from Hostl by sending an acknowledgment number one greater than its last received data sequence number. Both hosts send packets with the ACK flag set until the session is to end although the P and U flags may also be set, if warranted.") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 479 of 1100

| No. | '111 Patent Claim 17 | Chandrasekaran |
|---|---|---|
| | | Copeland at [0093] ("As illustrated, when Hostl terminates its end of the session, it sends a packet with the FIN and ACK flags set. The FIN flag informs Host2 that Hostl will send no more data. The ACK flag acknowledges the last data received by Hostl by informing Host2 of the next sequence number it expects to receive.")<br><br>Copeland at [0094] ("Host2 acknowledges the FIN packet by sending its own ACK packet. The ACK packet has the acknowledge-ment number one greater than the sequence number of Hostl's FIN-ACK packet. ACK packets are still delivered between the two hosts, except that HOSTl's packets have no data appended to the TCP/IP end of the headers.")<br><br>Copeland at [0095] ("When Host 2 is ready to terminate the session, it sends its own packet with the FIN and ACK flags set. Hostl responds that it has received the final packet with an ACK packet providing to Host2 an acknowledgment number one greater than the sequence number provided in the FIN-ACK packet of Host2.")<br><br>Uchida discloses wherein the one or more flag bits comprises a SYN flag bit, an ACK flag bit, a FIN flag bit, a RST flag bit, or any combination thereof.<br><br>As another example, Uchida discloses the TCP (Transmission Control Protocol) FIN flag, RST flag, and SYN flag *i.e.,* the one or more flag bits comprises comprise a SYN flag bit, an ACK flag bit, a FIN flag bit, a RST flag bit.<br><br>Uchida at [0040] ("A flow end can be detected by various methods as below. For example, in one method, a protocol end message is checked. For example, in the TCP (Transmission Control Protocol), a FIN flag is checked. In this way, the end of communication, that is, the end of a flow using communica-tion, can be detected. In practice, after a FIN flag, communi-cation with an ACK packet is generated in a reverse-direction flow (a flow in which the source and the destination are reversed). Thus, by detecting the ACK flag in the reverse-direction flow after the FIN packet, a flow end can be deter-mined. Further, since the TCP is used in bidirectional com-munication, the forward- and reverse-direction flows can be used as a pair to determine a flow end. Namely, if the end of a flow is detected, a process rule corresponding to the reverse-direction flow of the flow can also be determined |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 480 of 1100

| No. | '111 Patent Claim 17 | Chandrasekaran |
|---|---|---|
| | | to be unnec-essary. Alternatively, a communication end can also be deter-mined when a predetermined time elapses after reception of a SYN packet and a timeout is determined. Still alternatively, a communication end can be determined by reception of a RST packet. These methods will be described in more detail later as specific examples.")<br><br>Uchida at [0050] ("The flow end check unit can use at least one of a TCP (Transmission Control Protocol) FIN flag, RST flag, and SYN flag extracted by the end determination information extraction unit to determine a flow end.")<br><br>Uchida at [0055] ("In the process rule update method, a flow end can be determined by at least one of a TCP (Transmission Control Protocol) FIN flag, RST flag, and SYN flag.")<br><br>Uchida at [0102] ("Next, specific examples 1 to 3 will be described. In the examples 1 to 3, a flow end is determined by combining features of the above individual exemplary embodiments and using TCP (Transmission Control Protocol) flags.")<br><br>Uchida at [0103] ("FIG. 6 is a state transition diagram of TCP connec-tion. "CLOSED" at the top of FIG. 6 represents the end of TCP communication, and portions connected thereto repre-sent states prior to the end of TCP communication. Approxi-mately 2MSL (MSL: Maximum Segment Lifetime) is the maximum amount of time required to reach the above "CLOSED," that is, if the packet forwarding apparatus stands by for approximately 2MSL after both FINs flow, the above "CLOSED" is reached. Thus, after a FIN is confirmed in either direction, if this 2MSL elapses, basically, a communi-cation end can be determined. Even if the state does not change smoothly because of packet loss or the like (for example, even if an ACK packet does not arrive after "CLOS-ING"), a retransmitted packet is forwarded immediately after this 2MSL. Thus, the end of TCP communication can be determined if a new FIN packet is not received within the time corresponding to the 2MSL and a margin (2MSL+a) at long-est.")<br><br>Uchida at [0104] ("Hereinafter, the description will be made, assuming that a packet forwarding apparatus Cl according to the present invention relays TCP communication between a com-puter (client) Dl 0 and a server D20 that use network configu-rations illustrated in FIG. 7. In the example of FIG. 7, the computer Dl0 belongs to a network represented by 192.168. 0./24 and is set by 192.168.0.10. The server D20 belongs to a |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 481 of 1100

| No. | '111 Patent Claim 17 | Chandrasekaran |
|---|---|---|
| | | network represented by 192.168.1./24 and is set by 192.168. 1.10. As in the case of the OpenFlow controller described in Non-Patent Documents 1 and 2, a control apparatus ( control-ler) Dl is connected to the packet forwarding apparatus Cl via a dedicated channel and manages connection between the two networks. In the following description, the control appa-ratus (controller) Dl controls the packet forwarding appara-tus Cl so that connection from other networks appears as communication from network number 1 (192.168.1.1) of the respective networks (see process rule actions in FIG. 19). In addition, in the present specific example, since FIN packets are monitored, the end determination information extraction unit Cl 7 monitors a protocol stack, including: fields in which the TCP is determined; and the FIN flag in the TCP header.")<br><br>Uchida at [0105] ("FIG. 8 is a flow chart of a flow end determination process using FIN flags. In FIG. 8, steps relating to a timeout determination are added to steps Slll to S116 in the flow chart in FIG. 3. Thus, the flow chart in FIG. 8 includes more detailed steps than the flow chart of FIG. 3. Hereinafter, operations will be described with reference to FIGS. 3, 6, and 8 and FIGS. 9 to 13. In practice, prior to TCP/IP communi-cation, ARP (Address Resolution Protocol) communication is executed, and a process rule may be set in that stage. However, for ease of description, description of the ARP communication will be omitted. The following description will be made based on communication at the TCP/IP level.")<br><br>Uchida at [0106] ("First, the computer Dl0 starts communication with the server D20. For an initial establishment of communica-tion, a packet (SYN) is inputted to the packet forwarding apparatus Cl (start of ACTIVE OPEN through SYN forward-ing in FIG. 6). The packet reception unit Cl0 receives and stores this first packet in the packet storage unit Cll (steps SlOl to S102 in FIG. 3).")<br><br>Uchida at [0107] ("The packet reception unit C10 notifies the packet process information extraction unit C12 and the end determination information extraction unit C17 of reception of the packet. The packet process information extraction unit C12 refers to the packet storage unit C11 and extracts information such as IP source and destination information that is necessary to search for a process rule (step S103 in FIG. 3). Hereinafter, a process corresponding to steps S103 to S110 in FIG. 3 will be executed.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 482 of 1100

| No. | '111 Patent Claim 17 | Chandrasekaran |
|---|---|---|
| | | Uchida at [0115] ("Upon receiving a notification that the packet has been received by the packet reception unit Cl 0, the end deter-mination information extraction unit Cl 7 refers to the packet storage unit Cll, monitors a TCP FIN flag, and finds a FIN flag (step S201 in FIG. 8).")<br><br>Uchida at [0116] ("Since a FIN flag is set, the end determination infor-mation extraction unit Cl 7 determines that the packet includes information necessary for determining a flow end. Thus, the end determination information extraction unit Cl 7 extracts information for identifying a process rule to be deleted (the ingress port is 1; the source address is 192.168. 0.10; the destination is 192.168.1.10; and the protocol is TCP (the type is Ox0006)) and stands by until forwarding of the packet. Upon receiving a notification that the packet has been transmitted by the packet forwarding unit C16, the end deter-mination information extraction unit Cl 7 further extracts information for identifying a process rule to be deleted from the packet storage unit Cll. Since the IP address is replaced, the extracted information for identifying a process rule to be deleted represents that the source address is 192.168.1.1; the destination is 192.168.1.1 0; and the protocol is TCP (the type is 0x0006). The information is used for marking of the reverse flow. The end determination information extraction unit Cl 7 notifies the flow end check unit C18 of the notification that the FIN packet has been received and these items of information (step S202 in FIG. 8).")<br><br>Uchida at [0117] ("Upon receiving the above information from the end determination information extraction unit Cl 7, the flow end check unit C18 checks whether or not a FIN flag is set in a predetermined packet header position (step S203). These steps correspond to steps Slll to S114 in FIG. 3.")<br><br>Uchida at [0121] ("Next, after an ACK reply in response to the FIN packet from the computer DlO is forwarded from the server D20 in the same way as the above normal packet (start of PASSIVE CLOSE in FIG. 6), the server D20 transmits a FIN packet to the computer DlO. When this FIN packet is inputted to the packet forwarding apparatus Cl, the flow end determi-nation process from steps Slll to S116 is started, as in the case of the above start of ACTIVE CLOSE.")<br><br>Uchida at [0122] ("Upon receiving a notification that the packet has been received from the packet reception unit Cl0, the end determination information extraction unit Cl 7 refers to |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 483 of 1100

| No. | '111 Patent Claim 17 | Chandrasekaran |
|---|---|---|
| | | the packet storage unit Cll, monitors a TCP FIN flag, and finds a FIN packet (step S201 in FIG. 8).")<br><br>Uchida at [0123] ("Since a FIN flag is set, the end determination infor-mation extraction unit Cl 7 determines that the packet includes information necessary for determining a flow end. Thus, the end determination information extraction unit Cl 7 extracts information for identifying a process rule to be deleted (the ingress port is 2; the source address is 192.168. 1.10; the destination is 192.168.1.1; and the protocol is TCP (the type is Ox.0006)) and stands by until the packet is trans-mitted. Upon receiving a notification that the packet has been transmitted from the packet forwarding unit C16, the end determination information extraction unit Cl 7 further extracts information for identifying a modified process rule from the packet storage unit Cll. Since the IP address is replaced, the extracted information for identifying a modified process rule represents that the source address is 192.168.1. 10; the destination is 192.168.0.10; and the protocol is TCP (the type is 0x0006). The information is used for marking of the reverse flow. The end determination information extrac-tion unit Cl 7 notifies the flow end check unit C18 of the notification that the FIN packet has been received and these items of information (step S202 in FIG. 8).")<br><br>Uchida at [0124] ("Upon receiving the above information from the end determination information extraction unit Cl 7, the flow end check unit C18 checks whether or not a FIN flag is set in a predetermined packet header position (step S203 in FIG. 8). These steps correspond to steps Slll to S114 in FIG. 3.")<br><br>Uchida at [0125] ("At this point, since a FIN packet has been transmit-ted, the flow end check unit C18 uses the information for identifying a process rule to be deleted as a key, extracts the process rule (process rule corresponding to ingress port 2 in FIG. 11) from the process rule storage unit C13, and marks a FIN packet reception flag (steps S204 to S205 in FIG. 8). This process corresponds to the internal state update process in step S115 in FIG. 3.")<br><br>Uchida at [0134] ("Referring back to the state transition diagram of TCP connection in FIG. 6, there are two cases where "CLOSED" at the top of FIG. 6 is reached without a state transition involving FIN flags. One case arises when the ses-sion is closed from SYN_SENT, which is reached when a SYN packet in which a SYN flag is marked is |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 484 of 1100

| No. | '111 Patent Claim 17 | Chandrasekaran |
|---|---|---|
| | | transmitted. The other case arises when a timeout is generated. In such case, while the packet forwarding apparatus cannot monitor the closed session, the packet forwarding apparatus can con-firm a timeout in the following way. In the present specific example, a flow end is determined by this timeout.")<br><br>Uchida at [0135] ("n the present specific example, if a SYN/ ACK packet does not flow in a direction opposite to the SYN packet flow direction within a predetermined time (from "SYN_ RCVD" to "SYN_SENT" in FIG. 6), a timeout is determined.")<br><br>Uchida at [0136] ("FIG. 14 is a flow chart illustrating a flow end deter-mination process using a SYN flag. Since the basic operations are the same as those of the above specific example 1, the following description will be made with a focus on the dif-ference.")<br><br>Uchida at [0137] ("In FIG. 14, upon receiving a notification that the packet has been received by the packet reception unit ClO, the end determination information extraction unit Cl 7 refers to the packet storage, unit Cll, monitors a TCP SYN flag, and finds a SYN packet (step S301 in FIG. 14).")<br><br>Uchida at [0138] ("Since a SYN flag is set, the end determination infor-mation extraction unit Cl 7 determines that the packet includes information necessary for determining a flow end. Thus, the end determination information extraction unit Cl 7 extracts information for identifying a process rule to be deleted (the ingress port is 2; the source address is 192.168. 1.10; the destination is 192.168.1.1; and the protocol is TCP (the type is Ox.0006)) and stands by until the packet is trans-mitted. Upon receiving a notification that the packet has been transmitted by the packet forwarding unit C16, the end deter-mination information extraction unit Cl 7 further extracts information for identifying a modified process rule from the packet storage unit Cll. Since the IP address is replaced, the extracted information for identifying a process rule repre-sents that the source address is 192.168.1.10; the destination is 192.168.0.10; and the protocol is TCP (the type is 0x0006). The information is used for marking of the reverse flow. The end determination information extraction unit Cl 7 notifies the flow end check unit C18 of the notification that the SYN packet has been received and these items of information (step S302 in FIG. 14).") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 485 of 1100

| No. | '111 Patent Claim 17 | Chandrasekaran |
|---|---|---|
| | | Uchida at [0139] ("Upon receiving the above information from the end determination information extraction unit Cl 7, the flow end check unit C18 checks whether a SYN flag is set in a prede-termined packet header position and an ACK flag is not marked (step S303 in FIG. 14). These steps correspond to steps Slll to S114 in FIG. 3.")<br><br>Uchida at [0148] (" Next, a third specific example in which a flow end determination is executed by using a TCP RST (reset) flag will be described.")<br><br>Uchida at [0149] ("Referring back to the state transition diagram of TCP connection in FIG. 6, there is a transition from "SYN_ RCVD," which is a communication establishment standby state, to "LISTEN," which is a communication standby state. A TCP RST (reset) flag signifies release of connection and retry of communication. Namely, since a RST packet in which this RST flag is set signifies invalidation of communi-cation, by detecting this RST flag, a flow end can be deter-mined.")<br><br>Uchida at [0150] ("FIG. 16 is a first flow chart illustrating a flow end determination process using a RST flag. Since the basic operations are the same as those of the above specific example 1, the following description will be made with a focus on the difference.")<br><br>Uchida at [0151] ("In FIG. 16, upon receiving a notification that the packet has been received by the packet reception unit ClO, the end determination information extraction unit Cl 7 refers to the packet storage unit Cll, monitors a TCP RST flag, and finds a RST packet (step S401 in FIG. 16).")<br><br>Uchida at [0152] ("Since a RST flag is set, the end determination infor-mation extraction unit Cl 7 determines that the packet includes information necessary for determining a flow end. Thus, the end determination information extraction unit Cl 7 extracts information for identifying a process rule to be deleted (the ingress port is 2; the source address is 192.168. 1.10; the destination is 192.168.1.1; and the protocol is TCP (the type is Ox0006)) and stands by until the packet is trans-mitted. Upon receiving a notification that the packet has been transmitted from the packet forwarding unit C16, the end determination information extraction unit Cl 7 notifies the flow end check unit C18 of the notification that the RST packet has been received and these items of information ( step S402 in FIG. 16).") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 486 of 1100

| No. | '111 Patent Claim 17 | Chandrasekaran |
|---|---|---|
| | | Uchida at [0164] ("For example, in a specific example of the present invention, certain TCP flags are monitored. A single packet forwarding apparatus can monitor these flags in a parallel fashion. For example, after a packet that triggers a flow end is detected, the above process may be allowed to branch to the above FIGS. 8, 14, and 16 (17) to realize parallel monitoring.") |

| No. | '111 Patent Claim 18 | Chandrasekaran |
|---|---|---|
| 18[a] | The method according to claim 1, wherein the packet comprises distinct header and payload fields, | Chandrasekaran discloses the method according to claim 1, wherein the packet comprises distinct header and payload fields.<br><br>*See supra* at Claim 1, 15[a]. |
| 18[b] | the header comprises at least the first and second entities addresses in the packet network, and | Chandrasekaran discloses the header comprises at least the first and second entities addresses in the packet network.<br><br>For example, Chandrasekaran discloses packet headers that may include source and destination addresses.<br><br>Chandrasekaran at [0014] ("The term 'wireless controller' or 'controller' as used herein may refer to a wireless LAN (local area network) controller, mobility controller, wireless control device, wire-less control system, or any other network device operable to perform control functions for a wireless network. The net-work site may also include a wireless control system or other platform for centralized wireless LAN planning, configura-tion, and management. The wireless controller 12 enables system wide functions for wireless applications and may support any number of access points 14. Each access point 14 may serve any number of mobile devices 16 in the wireless network. The wireless controller 12 may be, for example, a standalone device or a rack-mounted appliance. In the example shown in FIG. 1, the wireless controller 12 and access points 14 are separate devices and may be located remote from one another. The wireless controller 12 may also be integrated with the access point 14 ( e.g., autonomous AP) or located at a switch, router, switch/router, or other network device. Thus, the wireless controller 12 may be a physical device located at |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 487 of 1100

| No. | '111 Patent Claim 18 | Chandrasekaran |
|---|---|---|
| | | a standalone device, access point, switch, router, or other network device. The wireless controller 12 may also be a virtual device located in a network or cloud, for example.")<br><br>Chandrasekaran at [0016] ("The wireless controller 12 includes a stateful appli-cation classifier 18 and the AP 14 includes a stateless appli-cation classifier 22. After the stateful classifier 18 identifies the application, the controller 12 transmits ( e.g., pushes) clas-sification information 26 to the AP 14 so that the AP can perform stateless classification and apply policies ( e.g., QoS or other policies) to traffic received from the mobile device 16. The controller 12 may also provide the classification information 26 to another AP 14 if the client 16 roams to a new AP, as shown in FIG. 1. Implementation of the stateful classifier 18 at the controller 12 and stateless classifier 22 at the AP 14 allows for policies to be applied for downstream traffic (packet 25) at the wireless controller 12, and for upstream traffic (packet 28) at the access point 14.")<br><br>Chandrasekaran at [0020] ("In one embodiment, the stateful classifier 18 is a classification engine configured for NBAR (Network Based Application Recognition) or other technology used to classify applications. The classifier 18 is operable to recognize a wide variety of applications, including Web-based and client/ server applications. The applications may include, for example, Skype, YouTube, Netflix, WebEx, Google Voice, BitTorrent, Citrix, virtual desktop, PCoIP, or any other appli-cation. The classification engine may be configured, for example, to identify generic protocols and perform heuristic analysis for encrypted protocols. The classifiers 18, 22 are configured to perform deep packet inspection (DPI), which provides the ability to look into the packet past basic header information so that the contents of a particular packet can be determined.")<br><br>Chandrasekaran at [0021] ("Once the application is recognized, QoS or other policies associated with the application can be applied to traffic so that the network can invoke services for that par-ticular application. For example, the application may have certain requirements and expectations from the network infrastructure, which may be specified in terms of bandwidth, delay, jitter, throughput, packet loss, or other performance attributes.")<br><br>Chandrasekaran at [0023] ("In one embodiment, the classification information 26 transmitted from the controller 12 to the AP 14 includes tuple information for a flow ( e.g., source IP address, destina-tion IP address, source port, destination port, and protocol), |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 488 of 1100

| No. | '111 Patent Claim 18 | Chandrasekaran |
|---|---|---|
| | | application identifier (ID), and stateless DPI information. Stateless DPI information includes classification and sub-classification information ( e.g., fixed or variable offset with a pattern or regular expression) and rules for applying policies on the sub-classified packets. The policies may include, for example, drop packet, mark a DSCP (Differentiated Services Code Point) value in the packet, or rate limit the traffic.")<br><br>Chandrasekaran at [0026] ("Memory 34 may be a volatile memory or non-vola-tile storage, which stores various applications, operating sys-tems, modules, and data for execution and use by the proces-sor 32. Memory 34 may include, for example, classification database 35. The classification database 35 may be any data structure configured for at least temporarily storing classifi-cation information including, for example, flow information, application ID, stateless DPI rules, and policies.")<br><br>Chandrasekaran at [0031] ("FIG. 3 is a flowchart illustrating an example of a process at the controller 12 for classification of traffic for application aware policies in a wireless network, in accor-dance with one embodiment. At step 40, the controller 12 receives packets belonging to a network flow. The controller 12 performs stateful classification to identify an application associated with the flow ( step 42). The controller 12 transmits classification information ( e.g., flow information, stateless DPI rule, and policy) to the AP 14 for use in stateless classi-fication at the AP (step 44). The controller 12 applies policies to downstream traffic (received at the controller and destined for the client 16) (step 46) and receives upstream traffic for which policies have been applied at the AP 14 (step 48). If the controller 12 determines ( e.g., receives an indication) that the client 16 has roamed, it transmits the classification informa-tion to the new AP 14 to which the client has roamed (steps 50 and 52).")<br><br>Chandrasekaran at [0033] ("The following describes an example of the above process for WebEx traffic that has different sub-classifications for voice and video traffic. Stateful classification is first performed by the controller 12 at the beginning of the flow. The controller 12 may need to process, for example, 10, 100, or any other number of packets to classify the flow as Web Ex traffic. Once the classification is performed, the controller 12 sends the stateless DPI rules and flow information to the AP 14 for stateless sub-classification to distinguish voice, video, or data within a WebEx flow. For example, after the controller 12 identifies the WebEx meeting traffic, it pushes the tuple, the stateless DPI |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 489 of 1100

| No. | '111 Patent Claim 18 | Chandrasekaran |
|---|---|---|
| | | rules (as shown below), and policies to the AP 14 for upstream traffic marking, dropping, or rate-limit-ing. If the client 16 roams, the controller 12 transmits the same classification information to the new AP to which the client has roamed.") |
| 18[c] | wherein the packet-applicable criterion is that the first entity address, the second entity address, or both match a predetermined address or addresses. | Chandrasekaran discloses wherein the packet-applicable criterion is that the first entity address, the second entity address, or both match a predetermined address or addresses.<br><br>For example, Chandrasekaran discloses the classification information, including source and destination addresses and other tuple information.<br><br>Chandrasekaran at [0014] ("The term 'wireless controller' or 'controller' as used herein may refer to a wireless LAN (local area network) controller, mobility controller, wireless control device, wire-less control system, or any other network device operable to perform control functions for a wireless network. The net-work site may also include a wireless control system or other platform for centralized wireless LAN planning, configura-tion, and management. The wireless controller 12 enables system wide functions for wireless applications and may support any number of access points 14. Each access point 14 may serve any number of mobile devices 16 in the wireless network. The wireless controller 12 may be, for example, a standalone device or a rack-mounted appliance. In the example shown in FIG. 1, the wireless controller 12 and access points 14 are separate devices and may be located remote from one another. The wireless controller 12 may also be integrated with the access point 14 ( e.g., autonomous AP) or located at a switch, router, switch/router, or other network device. Thus, the wireless controller 12 may be a physical device located at a standalone device, access point, switch, router, or other network device. The wireless controller 12 may also be a virtual device located in a network or cloud, for example.")<br><br>Chandrasekaran at [0016] ("The wireless controller 12 includes a stateful appli-cation classifier 18 and the AP 14 includes a stateless appli-cation classifier 22. After the stateful classifier 18 identifies the application, the controller 12 transmits ( e.g., pushes) clas-sification information 26 to the AP 14 so that the AP can perform stateless classification and apply policies ( e.g., QoS or other policies) to traffic received from the mobile device 16. The controller 12 may also provide the classification information 26 to another AP 14 if the client 16 roams to a new AP, as shown in FIG. 1. Implementation of |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 490 of 1100

| No. | '111 Patent Claim 18 | Chandrasekaran |
|---|---|---|
| | | the stateful classifier 18 at the controller 12 and stateless classifier 22 at the AP 14 allows for policies to be applied for downstream traffic (packet 25) at the wireless controller 12, and for upstream traffic (packet 28) at the access point 14.")<br><br>Chandrasekaran at [0020] ("In one embodiment, the stateful classifier 18 is a classification engine configured for NBAR (Network Based Application Recognition) or other technology used to classify applications. The classifier 18 is operable to recognize a wide variety of applications, including Web-based and client/ server applications. The applications may include, for example, Skype, YouTube, Netflix, WebEx, Google Voice, BitTorrent, Citrix, virtual desktop, PCoIP, or any other appli-cation. The classification engine may be configured, for example, to identify generic protocols and perform heuristic analysis for encrypted protocols. The classifiers 18, 22 are configured to perform deep packet inspection (DPI), which provides the ability to look into the packet past basic header information so that the contents of a particular packet can be determined.")<br><br>Chandrasekaran at [0021] ("Once the application is recognized, QoS or other policies associated with the application can be applied to traffic so that the network can invoke services for that par-ticular application. For example, the application may have certain requirements and expectations from the network infrastructure, which may be specified in terms of bandwidth, delay, jitter, throughput, packet loss, or other performance attributes.")<br><br>Chandrasekaran at [0023] ("In one embodiment, the classification information 26 transmitted from the controller 12 to the AP 14 includes tuple information for a flow ( e.g., source IP address, destina-tion IP address, source port, destination port, and protocol), application identifier (ID), and stateless DPI information. Stateless DPI information includes classification and sub-classification information ( e.g., fixed or variable offset with a pattern or regular expression) and rules for applying policies on the sub-classified packets. The policies may include, for example, drop packet, mark a DSCP (Differentiated Services Code Point) value in the packet, or rate limit the traffic.")<br><br>Chandrasekaran at [0026] ("Memory 34 may be a volatile memory or non-vola-tile storage, which stores various applications, operating sys-tems, modules, and data for execution and use by the proces-sor 32. Memory 34 may include, for example, classification database 35. The classification database 35 may be any data structure configured for at least temporarily |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 491 of 1100

| No. | '111 Patent Claim 18 | Chandrasekaran |
|---|---|---|
| | | storing classifi-cation information including, for example, flow information, application ID, stateless DPI rules, and policies.")<br><br>Chandrasekaran at [0031] ("FIG. 3 is a flowchart illustrating an example of a process at the controller 12 for classification of traffic for application aware policies in a wireless network, in accor-dance with one embodiment. At step 40, the controller 12 receives packets belonging to a network flow. The controller 12 performs stateful classification to identify an application associated with the flow ( step 42). The controller 12 transmits classification information ( e.g., flow information, stateless DPI rule, and policy) to the AP 14 for use in stateless classi-fication at the AP (step 44). The controller 12 applies policies to downstream traffic (received at the controller and destined for the client 16) (step 46) and receives upstream traffic for which policies have been applied at the AP 14 (step 48). If the controller 12 determines ( e.g., receives an indication) that the client 16 has roamed, it transmits the classification informa-tion to the new AP 14 to which the client has roamed (steps 50 and 52).")<br><br>Chandrasekaran at [0033] ("The following describes an example of the above process for WebEx traffic that has different sub-classifications for voice and video traffic. Stateful classification is first performed by the controller 12 at the beginning of the flow. The controller 12 may need to process, for example, 10, 100, or any other number of packets to classify the flow as Web Ex traffic. Once the classification is performed, the controller 12 sends the stateless DPI rules and flow information to the AP 14 for stateless sub-classification to distinguish voice, video, or data within a WebEx flow. For example, after the controller 12 identifies the WebEx meeting traffic, it pushes the tuple, the stateless DPI rules (as shown below), and policies to the AP 14 for upstream traffic marking, dropping, or rate-limit-ing. If the client 16 roams, the controller 12 transmits the same classification information to the new AP to which the client has roamed.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 492 of 1100

| No. | '111 Patent Claim 19 | Chandrasekaran |
|---|---|---|
| 19 | The method according to claim 18, wherein the addresses are Internet Protocol (IP) addresses. | Chandrasekaran discloses the method according to claim 18, wherein the addresses are Internet Protocol (IP) addresses.<br><br>For example, Chandrasekaran discloses source and destination IP addresses.<br><br>*See supra* at Claim 18.<br><br>Chandrasekaran at [0012] ("Referring now to the drawings, and first to FIG.1, an example of a network in which embodiments described herein may be implemented is shown. For simplification, only a small number of network devices are shown. The network includes a wireless controller 12 in communication with a mobile device (client, wireless device, endpoint) 16 through an access point (AP) 14. In the example shown in FIG. 1, the controller 12 is in wired communication with two access points 14 for wireless communication with any number of mobile devices 16 via a wireless network ( e.g., WLAN (wire-less local area network)) at a network site. The wireless con-troller 12 may be in communication with one or more other networks (not shown) (e.g., Internet, intranet, local area net-work, wireless local area network, cellular network, metro-politan area network, wide area network, satellite network, radio access network, public switched network, virtual pri-vate network, or any other network or combination thereof). Communication paths between the wireless controller 12 and other networks or between the controller and access points 14 may include any number or type of intermediate nodes (e.g., routers, switches, gateways, or other network devices), which facilitate passage of data between network devices.")<br><br>Chandrasekaran at [0017] ("The stateful classifier 18 at the controller 12 classi-fies traffic based on multiple packets received from the begin-ning of a flow. Stateful classification uses rules which need information on states for a previous packet ( or packets) in a flow. Stateful classification may be based, for example, on packet pattern matching and decoding of protocols and their states. Stateful classification is also referred to as flow clas-sification since it looks at a data stream of related packets (flow, session).")<br><br>Chandrasekaran at [0018] ("The stateless classifier 22 at the AP 14 uses rules that can act on a per packet basis in the flow. Stateless classifica-tion (also referred to as packet |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 493 of 1100

| No. | '111 Patent Claim 19 | Chandrasekaran |
|---|---|---|
| | | classification) is based on individual packet inspection ( e.g., 5 tuple, pattern matching) without knowledge of any related stream of packets, flows, sessions, or protocols.")<br><br>Chandrasekaran at [0023] ("In one embodiment, the classification information 26 transmitted from the controller 12 to the AP 14 includes tuple information for a flow ( e.g., source IP address, destina-tion IP address, source port, destination port, and protocol), application identifier (ID), and stateless DPI information. Stateless DPI information includes classification and sub-classification information ( e.g., fixed or variable offset with a pattern or regular expression) and rules for applying policies on the sub-classified packets. The policies may include, for example, drop packet, mark a DSCP (Differentiated Services Code Point) value in the packet, or rate limit the traffic.") |

| No. | '111 Patent Claim 20 | Chandrasekaran |
|---|---|---|
| 20[a] | The method according to claim 1, wherein the packet is an Transmission Control Protocol (TCP) packet that comprises source and destination TCP ports, a TCP sequence number, and a TCP sequence mask fields, and | Chandrasekaran discloses the method according to claim 1, wherein the packet is an Transmission Control Protocol (TCP) packet that comprises source and destination TCP ports, a TCP sequence number, and a TCP sequence mask fields.<br><br>*See supra* at Claim 1, 17[a]. |
| 20[b] | wherein the packet-applicable criterion is that the source TCP port, the destination TCP port, the TCP sequence number, the TCP sequence mask, or any combination thereof, matches a | Chandrasekaran discloses wherein the packet-applicable criterion is that the source TCP port, the destination TCP port, the TCP sequence number, the TCP sequence mask, or any combination thereof, matches a predetermined value or values.<br><br>For example, Chandrasekaran discloses classification information that can include different identifiers of a packet. A person of ordinary skill in the art would understand that the classification information transmitted could be the source TCP port, the destination TCP port, the TCP sequence number, the TCP sequence mask, or any combination thereof, which are applied to policy rules. Thus, at least under the apparent claim scope alleged by Orckit's |

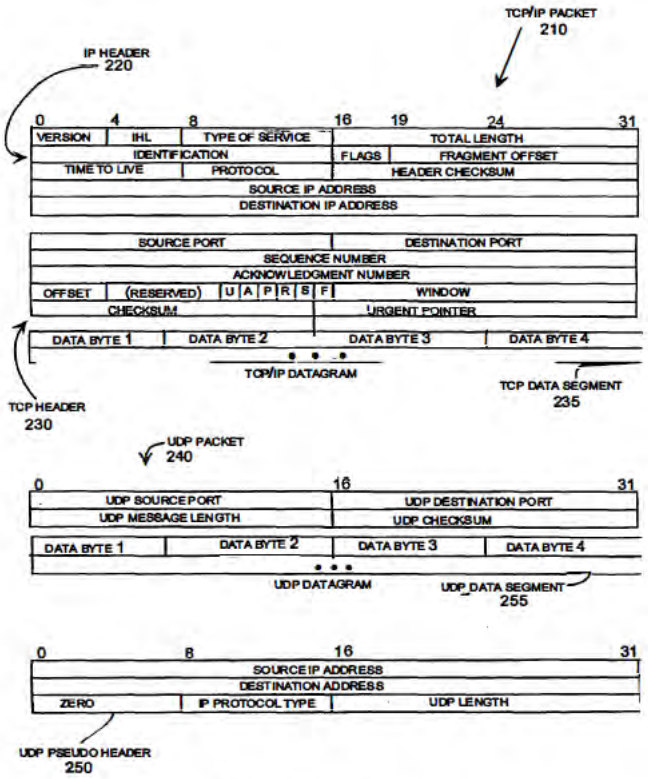


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 494 of 1100

| No. | '111 Patent Claim 20 | Chandrasekaran |
|---|---|---|
| | predetermined value or values. | Infringement Disclosures, this limitation is met. To the extent that the Chandrasekaran is found to not meet this limitation, wherein the packet-applicable criterion is that the source TCP port, the destination TCP port, the TCP sequence number, the TCP sequence mask, or any combination thereof, matches a predetermined value or values would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>Chandrasekaran at [0023] ("In one embodiment, the classification information 26 transmitted from the controller 12 to the AP 14 includes tuple information for a flow ( e.g., source IP address, destina-tion IP address, source port, destination port, and protocol), application identifier (ID), and stateless DPI information. Stateless DPI information includes classification and sub-classification information ( e.g., fixed or variable offset with a pattern or regular expression) and rules for applying policies on the sub-classified packets. The policies may include, for example, drop packet, mark a DSCP (Differentiated Services Code Point) value in the packet, or rate limit the traffic.")<br><br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Chandrasekaran in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 17(a) of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references.<br><br>For example, Copeland discloses TCP packets with TCP port information.<br><br>Copeland at Figure 2 |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 495 of 1100

| No. | '111 Patent Claim 20 | Chandrasekaran |
|---|---|---|
| | |  <br>FIG. 2<br><br>Copeland at [0076] ("FIG. 2 illustrates an exemplary TCP/IP packet or datagram 210 and an exemplary UDP datagram 240. In a typical TCP/IP packet like 210, each packet typically includes a header portion comprising an IP header 220 and a TCP header 230, followed by a data portion that contains the information to be communicated in the packet. The information in the IP header 220 contained in a TCP/IP packet 210, or any other IP packet, contains the IP addresses and assures that the packet is delivered to the right host. The transport layer protocol (TCP) header follows the Internet protocol header and specifies the port numbers for the associated service.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 496 of 1100

| No. | '111 Patent Claim 20 | Chandrasekaran |
|---|---|---|
| | | Copeland at [0077] ("The header portion in the typical TCP/IP datagram 210 is 40 bytes including 20 bytes of IP header 220 information and 20 bytes of TCP header 230 information. The data portion or segment associated with the packet 210 follows the header information.")<br><br>For example, Chua discloses packet traffic using TCP services with TCP port information.<br><br>Chua at 18:8-44 ("Based on the rule set for a specific device, including what traffic is allowed through, from what zones and expected targets, administrator 114 can infer the right type of test traffic that should be injected for both positive and negative testing of the device. Similarly, the inference can be made to deter-mine the appropriate type of flow troubleshooting that includes the appropriate IP ranges to watch on, and the TCP/ UDP services that the flowchain is meant to service.<br><br>In the case of traffic injection to test a path through SDN 106, path verification unit 136 of SDN controller 112 may perform any of the following tasks:<br><br>Infer from the flowchain the appropriate type of transmission control protocol (TCP) or uniform datagram proto-col (UDP) services that are relevant to the chain, as well as the range of source and target IP addresses for this chain<br><br>Create either UDP packet flows or TCP sessions that confirm to the traffic type inferred or discovered form the flowchain configuration<br><br>Using an SDN device ( e.g., a switch of SDN 106), inject these flows or TCP sessions by temporarily inserting a rule into the rules table on the switch that will take the flows from SDN controller 112 and send them across the devices of SDN 106 (starting from the ingress device in the chain)<br><br>Using an SDN device, insert flow rules to replicate (flow-span) these same traffic streams at each point in the flow chain. The replicated traffic may be directed (via SDN rules) to a collection device ( or to SDN controller 112) that can determine whether the test packets passed through each port on the switch that is part of the flowchain |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 497 of 1100

| No. | '111 Patent Claim 20 | Chandrasekaran |
|---|---|---|
| | | On the final exit device (prior to the end target), insert a flow rule that will drain all injected packets so the final devices (that is, devices external to SDN 106) do not see any of the test traffic.") |

| No. | '111 Patent Claim 21 | Chandrasekaran |
|---|---|---|
| 21 | The method according to claim 1, wherein the packet network comprises a Wide Area Network (WAN), Local Area Network (LAN), the Internet, Metropolitan Area Network (MAN), Internet Service Provider (ISP) backbone datacenter network, or inter - datacenter network. | Chandrasekaran discloses the method according to claim 1, wherein the packet network comprises a Wide Area Network (WAN), Local Area Network (LAN), the Internet, Metropolitan Area Network (MAN), Internet Service Provider (ISP) backbone datacenter network, or inter - datacenter network.<br><br>For example, Chandrasekaran discloses a packet network comprising a Internet, intranet, local area net-work, wireless local area network, cellular network, metro-politan area network, wide area network, satellite network, radio access network, public switched network, virtual pri-vate network, or any other network or combination thereof<br>.<br>*See supra* at Claim 1.<br><br>Chandrasekaran at [0012] ("Referring now to the drawings, and first to FIG.1, an example of a network in which embodiments described herein may be implemented is shown. For simplification, only a small number of network devices are shown. The network includes a wireless controller 12 in communication with a mobile device (client, wireless device, endpoint) 16 through an access point (AP) 14. In the example shown in FIG. 1, the controller 12 is in wired communication with two access points 14 for wireless communication with any number of mobile devices 16 via a wireless network ( e.g., WLAN (wire-less local area network)) at a network site. The wireless con-troller 12 may be in communication with one or more other networks (not shown) (e.g., Internet, intranet, local area net-work, wireless local area network, cellular network, metro-politan area network, wide area network, satellite network, radio access network, public switched network, virtual pri-vate network, or any other network or combination thereof). Communication paths between the wireless controller 12 and other networks or between the controller and access points 14 may include any number or type of intermediate nodes (e.g., routers, switches, |


Orckit-Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 498 of 1100

| No. | '111 Patent Claim 21 | Chandrasekaran |
|---|---|---|
| | | gateways, or other network devices), which facilitate passage of data between network devices.")<br><br>Chandrasekaran at [0014] ("The term 'wireless controller' or 'controller' as used herein may refer to a wireless LAN (local area network) controller, mobility controller, wireless control device, wire-less control system, or any other network device operable to perform control functions for a wireless network. The net-work site may also include a wireless control system or other platform for centralized wireless LAN planning, configura-tion, and management. The wireless controller 12 enables system wide functions for wireless applications and may support any number of access points 14. Each access point 14 may serve any number of mobile devices 16 in the wireless network. The wireless controller 12 may be, for example, a standalone device or a rack-mounted appliance. In the example shown in FIG. 1, the wireless controller 12 and access points 14 are separate devices and may be located remote from one another. The wireless controller 12 may also be integrated with the access point 14 ( e.g., autonomous AP) or located at a switch, router, switch/router, or other network device. Thus, the wireless controller 12 may be a physical device located at a standalone device, access point, switch, router, or other network device. The wireless controller 12 may also be a virtual device located in a network or cloud, for example.") |

| No. | '111 Patent Claim 22 | Chandrasekaran |
|---|---|---|
| 22 | The method according to claim 1, wherein the first entity is a server device and the second entity is a client device, or wherein the first entity is a client device and the second entity is a server device.. | Chandrasekaran discloses the method according to claim 1, wherein the first entity is a server device and the second entity is a client device, or wherein the first entity is a client device and the second entity is a server device.<br><br>For example, Chandrasekaran discloses mobile devices that may be a client, wireless device, endpoint, host, user device, etc, that employ client/server applications. A person of ordinary skill in the art would understand that a first mobile device may be either a server device or client device and a second mobile may be either a server device or client device.<br><br>*See supra* at Claim 1. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 499 of 1100

| No. | '111 Patent Claim 22 | Chandrasekaran |
|---|---|---|
| | | Chandrasekaran at [0012] ("Referring now to the drawings, and first to FIG.1, an example of a network in which embodiments described herein may be implemented is shown. For simplification, only a small number of network devices are shown. The network includes a wireless controller 12 in communication with a mobile device (client, wireless device, endpoint) 16 through an access point (AP) 14. In the example shown in FIG. 1, the controller 12 is in wired communication with two access points 14 for wireless communication with any number of mobile devices 16 via a wireless network (e.g., WLAN (wire-less local area network)) at a network site. The wireless con-troller 12 may be in communication with one or more other networks (not shown) (e.g., Internet, intranet, local area net-work, wireless local area network, cellular network, metro-politan area network, wide area network, satellite network, radio access network, public switched network, virtual pri-vate network, or any other network or combination thereof). Communication paths between the wireless controller 12 and other networks or between the controller and access points 14 may include any number or type of intermediate nodes (e.g., routers, switches, gateways, or other network devices), which facilitate passage of data between network devices.")<br><br>Chandrasekaran at [0013] ("In one example, the wireless controller 12 receives upstream traffic transmitted from the mobile device 16 and destined for another endpoint ( e.g., host, user device), and transmits downstream traffic received from the endpoint to the mobile device in a communication session. As used herein, the term 'downstream' refers to traffic transmitted from the controller 12 towards the mobile device 16, and the term 'upstream' refers to traffic transmitted from the mobile device towards the controller.")<br><br>Chandrasekaran at [0014] ("The term 'wireless controller' or 'controller' as used herein may refer to a wireless LAN (local area network) controller, mobility controller, wireless control device, wire-less control system, or any other network device operable to perform control functions for a wireless network. The net-work site may also include a wireless control system or other platform for centralized wireless LAN planning, configura-tion, and management. The wireless controller 12 enables system wide functions for wireless applications and may support any number of access points 14. Each access point 14 may serve any number of mobile devices 16 in the wireless network. The wireless controller 12 may be, for example, a standalone device or a rack-mounted appliance. In the example shown in FIG. 1, the wireless controller 12 and access points 14 are separate devices and |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 500 of 1100

| No. | '111 Patent Claim 22 | Chandrasekaran |
|---|---|---|
| | | may be located remote from one another. The wireless controller 12 may also be integrated with the access point 14 ( e.g., autonomous AP) or located at a switch, router, switch/router, or other network device. Thus, the wireless controller 12 may be a physical device located at a standalone device, access point, switch, router, or other network device. The wireless controller 12 may also be a virtual device located in a network or cloud, for example.")<br><br>Chandrasekaran at [0015] ("The mobile device 16 may be any suitable equip-ment that supports wireless communication, including for example, a mobile phone, personal digital assistant, portable computing device, laptop, tablet, multimedia device, or any other wireless device. The mobile device 16 and access point 14 are configured to perform wireless communication according to a wireless network communication protocol such as IEEE 802.11/Wi-Fi.")<br><br>Chandrasekaran at [0016] ("The wireless controller 12 includes a stateful appli-cation classifier 18 and the AP 14 includes a stateless appli-cation classifier 22. After the stateful classifier 18 identifies the application, the controller 12 transmits ( e.g., pushes) clas-sification information 26 to the AP 14 so that the AP can perform stateless classification and apply policies ( e.g., QoS or other policies) to traffic received from the mobile device 16. The controller 12 may also provide the classification information 26 to another AP 14 if the client 16 roams to a new AP, as shown in FIG. 1. Implementation of the stateful classifier 18 at the controller 12 and stateless classifier 22 at the AP 14 allows for policies to be applied for downstream traffic (packet 25) at the wireless controller 12, and for upstream traffic (packet 28) at the access point 14.")<br><br>Chandrasekaran at [0020] ("In one embodiment, the stateful classifier 18 is a classification engine configured for NBAR (Network Based Application Recognition) or other technology used to classify applications. The classifier 18 is operable to recognize a wide variety of applications, including Web-based and client/ server applications. The applications may include, for example, Skype, YouTube, Netflix, WebEx, Google Voice, BitTorrent, Citrix, virtual desktop, PCoIP, or any other appli-cation. The classification engine may be configured, for example, to identify generic protocols and perform heuristic analysis for encrypted protocols. The classifiers 18, 22 are configured to perform deep packet inspection (DPI), which provides the ability to look into the packet past basic header information so that the contents of a particular packet can be determined.") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 501 of 1100

| No. | '111 Patent Claim 22 | Chandrasekaran |
|---|---|---|
| | | |

| No. | '111 Patent Claim 23 | Chandrasekaran |
|---|---|---|
| 23[a] | The method according to claim 22, wherein the server device comprises a web server, and | Chandrasekaran discloses the method according to claim 22, wherein the server device comprises a web server.<br><br>For example, Chandrasekaran discloses a mobile device that may be a server device, further comprising a web device. A person of ordinary skill in the art would understand that a mobile device that may be a server device, may further comprise a web device.<br><br>*See supra* at Claim 22.<br><br>Chandrasekaran at [0012] ("Referring now to the drawings, and first to FIG.1, an example of a network in which embodiments described herein may be implemented is shown. For simplification, only a small number of network devices are shown. The network includes a wireless controller 12 in communication with a mobile device (client, wireless device, endpoint) 16 through an access point (AP) 14. In the example shown in FIG. 1, the controller 12 is in wired communication with two access points 14 for wireless communication with any number of mobile devices 16 via a wireless network ( e.g., WLAN (wire-less local area network)) at a network site. The wireless con-troller 12 may be in communication with one or more other networks (not shown) (e.g., Internet, intranet, local area net-work, wireless local area network, cellular network, metro-politan area network, wide area network, satellite network, radio access network, public switched network, virtual pri-vate network, or any other network or combination thereof). Communication paths between the wireless controller 12 and other networks or between the controller and access points 14 may include any number or type of intermediate nodes (e.g., routers, switches, gateways, or other network devices), which facilitate passage of data between network devices.") |


~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 502 of 1100

| No. | '111 Patent Claim 23 | Chandrasekaran |
|---|---|---|
| | | Chandrasekaran at [0013] ("In one example, the wireless controller 12 receives upstream traffic transmitted from the mobile device 16 and destined for another endpoint ( e.g., host, user device), and transmits downstream traffic received from the endpoint to the mobile device in a communication session. As used herein, the term 'downstream' refers to traffic transmitted from the controller 12 towards the mobile device 16, and the term 'upstream' refers to traffic transmitted from the mobile device towards the controller.")<br><br>Chandrasekaran at [0014] ("The term 'wireless controller' or 'controller' as used herein may refer to a wireless LAN (local area network) controller, mobility controller, wireless control device, wire-less control system, or any other network device operable to perform control functions for a wireless network. The net-work site may also include a wireless control system or other platform for centralized wireless LAN planning, configura-tion, and management. The wireless controller 12 enables system wide functions for wireless applications and may support any number of access points 14. Each access point 14 may serve any number of mobile devices 16 in the wireless network. The wireless controller 12 may be, for example, a standalone device or a rack-mounted appliance. In the example shown in FIG. 1, the wireless controller 12 and access points 14 are separate devices and may be located remote from one another. The wireless controller 12 may also be integrated with the access point 14 ( e.g., autonomous AP) or located at a switch, router, switch/router, or other network device. Thus, the wireless controller 12 may be a physical device located at a standalone device, access point, switch, router, or other network device. The wireless controller 12 may also be a virtual device located in a network or cloud, for example.")<br><br>Chandrasekaran at [0015] ("The mobile device 16 may be any suitable equip-ment that supports wireless communication, including for example, a mobile phone, personal digital assistant, portable computing device, laptop, tablet, multimedia device, or any other wireless device. The mobile device 16 and access point 14 are configured to perform wireless communication according to a wireless network communication protocol such as IEEE 802.11/Wi-Fi.")<br><br>Chandrasekaran at [0016] ("The wireless controller 12 includes a stateful appli-cation classifier 18 and the AP 14 includes a stateless appli-cation classifier 22. After the stateful classifier 18 identifies the application, the controller 12 transmits ( e.g., pushes) |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 503 of 1100

| No. | '111 Patent Claim 23 | Chandrasekaran |
|---|---|---|
| | | clas-sification information 26 to the AP 14 so that the AP can perform stateless classification and apply policies ( e.g., QoS or other policies) to traffic received from the mobile device 16. The controller 12 may also provide the classification information 26 to another AP 14 if the client 16 roams to a new AP, as shown in FIG. 1. Implementation of the stateful classifier 18 at the controller 12 and stateless classifier 22 at the AP 14 allows for policies to be applied for downstream traffic (packet 25) at the wireless controller 12, and for upstream traffic (packet 28) at the access point 14.")<br><br>Chandrasekaran at [0020] ("In one embodiment, the stateful classifier 18 is a classification engine configured for NBAR (Network Based Application Recognition) or other technology used to classify applications. The classifier 18 is operable to recognize a wide variety of applications, including Web-based and client/ server applications. The applications may include, for example, Skype, YouTube, Netflix, WebEx, Google Voice, BitTorrent, Citrix, virtual desktop, PCoIP, or any other appli-cation. The classification engine may be configured, for example, to identify generic protocols and perform heuristic analysis for encrypted protocols. The classifiers 18, 22 are configured to perform deep packet inspection (DPI), which provides the ability to look into the packet past basic header information so that the contents of a particular packet can be determined.") |
| 23[b] | wherein the client device comprises a smartphone, a tablet computer, a personal computer, a laptop computer, or a wearable computing device. | Chandrasekaran discloses wherein the client device comprises a smartphone, a tablet computer, a personal computer, a laptop computer, or a wearable computing device.<br><br>For example, Chandrasekaran discloses a mobile device that may be a client device, wireless device, or endpoint, which may be any of a smartphone, a tablet computer, a personal computer, a laptop computer, or a wearable computing device<br><br>Chandrasekaran at [0012] ("Referring now to the drawings, and first to FIG.1, an example of a network in which embodiments described herein may be implemented is shown. For simplification, only a small number of network devices are shown. The network includes a wireless controller 12 in communication with a mobile device (client, wireless device, endpoint) 16 through an access point (AP) 14. In the example shown in FIG. 1, the controller 12 is in wired communication with two access points 14 for wireless communication with any number of mobile devices 16 via a wireless network ( e.g., WLAN |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 504 of 1100

| No. | '111 Patent Claim 23 | Chandrasekaran |
|---|---|---|
| | | (wire-less local area network)) at a network site. The wireless con-troller 12 may be in communication with one or more other networks (not shown) (e.g., Internet, intranet, local area net-work, wireless local area network, cellular network, metro-politan area network, wide area network, satellite network, radio access network, public switched network, virtual pri-vate network, or any other network or combination thereof). Communication paths between the wireless controller 12 and other networks or between the controller and access points 14 may include any number or type of intermediate nodes (e.g., routers, switches, gateways, or other network devices), which facilitate passage of data between network devices.")<br><br>Chandrasekaran at [0013] ("In one example, the wireless controller 12 receives upstream traffic transmitted from the mobile device 16 and destined for another endpoint ( e.g., host, user device), and transmits downstream traffic received from the endpoint to the mobile device in a communication session. As used herein, the term 'downstream' refers to traffic transmitted from the controller 12 towards the mobile device 16, and the term 'upstream' refers to traffic transmitted from the mobile device towards the controller.")<br><br>Chandrasekaran at [0014] ("The term 'wireless controller' or 'controller' as used herein may refer to a wireless LAN (local area network) controller, mobility controller, wireless control device, wire-less control system, or any other network device operable to perform control functions for a wireless network. The net-work site may also include a wireless control system or other platform for centralized wireless LAN planning, configura-tion, and management. The wireless controller 12 enables system wide functions for wireless applications and may support any number of access points 14. Each access point 14 may serve any number of mobile devices 16 in the wireless network. The wireless controller 12 may be, for example, a standalone device or a rack-mounted appliance. In the example shown in FIG. 1, the wireless controller 12 and access points 14 are separate devices and may be located remote from one another. The wireless controller 12 may also be integrated with the access point 14 ( e.g., autonomous AP) or located at a switch, router, switch/router, or other network device. Thus, the wireless controller 12 may be a physical device located at a standalone device, access point, switch, router, or other network device. The wireless controller 12 may also be a virtual device located in a network or cloud, for example.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 505 of 1100

| No. | '111 Patent Claim 23 | Chandrasekaran |
|---|---|---|
| | | Chandrasekaran at [0015] ("The mobile device 16 may be any suitable equip-ment that supports wireless communication, including for example, a mobile phone, personal digital assistant, portable computing device, laptop, tablet, multimedia device, or any other wireless device. The mobile device 16 and access point 14 are configured to perform wireless communication according to a wireless network communication protocol such as IEEE 802.11/Wi-Fi.")<br><br>Chandrasekaran at [0016] ("The wireless controller 12 includes a stateful appli-cation classifier 18 and the AP 14 includes a stateless appli-cation classifier 22. After the stateful classifier 18 identifies the application, the controller 12 transmits ( e.g., pushes) clas-sification information 26 to the AP 14 so that the AP can perform stateless classification and apply policies ( e.g., QoS or other policies) to traffic received from the mobile device 16. The controller 12 may also provide the classification information 26 to another AP 14 if the client 16 roams to a new AP, as shown in FIG. 1. Implementation of the stateful classifier 18 at the controller 12 and stateless classifier 22 at the AP 14 allows for policies to be applied for downstream traffic (packet 25) at the wireless controller 12, and for upstream traffic (packet 28) at the access point 14.")<br><br>Chandrasekaran at [0020] ("In one embodiment, the stateful classifier 18 is a classification engine configured for NBAR (Network Based Application Recognition) or other technology used to classify applications. The classifier 18 is operable to recognize a wide variety of applications, including Web-based and client/ server applications. The applications may include, for example, Skype, YouTube, Netflix, WebEx, Google Voice, BitTorrent, Citrix, virtual desktop, PCoIP, or any other appli-cation. The classification engine may be configured, for example, to identify generic protocols and perform heuristic analysis for encrypted protocols. The classifiers 18, 22 are configured to perform deep packet inspection (DPI), which provides the ability to look into the packet past basic header information so that the contents of a particular packet can be determined.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 506 of 1100

| No. | '111 Patent Claim 24 | Chandrasekaran |
|---|---|---|
| 24 | The method according to claim 22, wherein the communication between the network node and the controller is based on, or uses, a standard protocol. | Chandrasekaran discloses the method according to claim 22, wherein the communication between the network node and the controller is based on, or uses, a standard protocol.<br><br>For example, Chandrasekaran discloses a wireless network where communication between the controller and access point is based on Internet, intranet, local area net-work, wireless local area network, cellular network, metro-politan area network, wide area network, satellite network, radio access network, public switched network, virtual private network, or any other network or combination thereof, or any other standard protocol. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Chandrasekaran is found to not meet this limitation, wherein the communication between the network node and the controller is based on, or uses, a standard protocol would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>*See supra* at Claim 22.<br><br>Chandrasekaran at [0012] ("Referring now to the drawings, and first to FIG.1, an example of a network in which embodiments described herein may be implemented is shown. For simplification, only a small number of network devices are shown. The network includes a wireless controller 12 in communication with a mobile device (client, wireless device, endpoint) 16 through an access point (AP) 14. In the example shown in FIG. 1, the controller 12 is in wired communication with two access points 14 for wireless communication with any number of mobile devices 16 via a wireless network ( e.g., WLAN (wire-less local area network)) at a network site. The wireless con-troller 12 may be in communication with one or more other networks (not shown) (e.g., Internet, intranet, local area net-work, wireless local area network, cellular network, metro-politan area network, wide area network, satellite network, radio access network, public switched network, virtual pri-vate network, or any other network or combination thereof). Communication paths between the wireless controller 12 and other networks or between the controller and access points 14 may include any number or type of intermediate nodes (e.g., routers, switches, gateways, or other network devices), which facilitate passage of data between network devices.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 507 of 1100

| No. | '111 Patent Claim 24 | Chandrasekaran |
|---|---|---|
| | | Chandrasekaran at [0017] ("The stateful classifier 18 at the controller 12 classi-fies traffic based on multiple packets received from the begin-ning of a flow. Stateful classification uses rules which need information on states for a previous packet ( or packets) in a flow. Stateful classification may be based, for example, on packet pattern matching and decoding of protocols and their states. Stateful classification is also referred to as flow clas-sification since it looks at a data stream of related packets (flow, session).")<br><br>Chandrasekaran at [0018] ("The stateless classifier 22 at the AP 14 uses rules that can act on a per packet basis in the flow. Stateless classifica-tion (also referred to as packet classification) is based on individual packet inspection ( e.g., 5 tuple, pattern matching) without knowledge of any related stream of packets, flows, sessions, or protocols.")<br><br>Chandrasekaran at [0023] ("In one embodiment, the classification information 26 transmitted from the controller 12 to the AP 14 includes tuple information for a flow ( e.g., source IP address, destina-tion IP address, source port, destination port, and protocol), application identifier (ID), and stateless DPI information. Stateless DPI information includes classification and sub-classification information ( e.g., fixed or variable offset with a pattern or regular expression) and rules for applying policies on the sub-classified packets. The policies may include, for example, drop packet, mark a DSCP (Differentiated Services Code Point) value in the packet, or rate limit the traffic.")<br><br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Chandrasekaran in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 24 of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references.<br><br>For example, Kempf discloses a packet network using the OpenFlow protocol, which is used in Software Defined Networks for communication between network device and a controller.<br><br>Kempf at [0004] ("The GPRS tunneling protocol (GTP) is an important communication protocol utilized within the GPRS core net-work. GTP enables end user devices ( e.g., cellular phones) in a GSM network to move from place to place while continuing to connect |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 508 of 1100

| No. | '111 Patent Claim 24 | Chandrasekaran |
|---|---|---|
| | | to the Internet. The end user devices are connected to other devices through a gateway GPRS support node (GGSN). The GGSN tracks the end user device's data from the end user device's serving GPRS support node (GGSN) that is handling the session originating from the end user device.")<br><br>Kempf at [0006] ("A method implements a control plane of an evolved packet core (EPC) of a third generation partnership project (3GPP) long term evolution (LTE) network in a cloud com-puting system. The cloud computing system includes a cloud manager and a controller. The controller executes a plurality of control plane modules. The control plane communicates with the data plane of the EPC implemented in a plurality of network elements of the 3GPP LTE network through a control protocol. The EPC with the control plane implemented in the cloud computing system utilizes resources more efficiently than an architecture with the control plane implemented in the plurality of network elements of the 3GPP LTE network. The method comprises the steps of initializing the plurality of control plane modules of the EPC within the controller. Each control plane module in the plurality of control plane modules is initialized as a separate virtual machine by the cloud man-ager. Each control plane module provides a set of control plane functions for managing the data plane. The cloud man-ager monitors resource utilization of each control plane mod-ule and the control plane traffic handled by each control plane module. The cloud manager detects a threshold level of resource utilization or traffic load for one of the plurality of control plane modules of the EPC. A new control plane mod-ule is initialized as a separate virtual machine by the cloud manager in response to detecting the threshold level. The new control plane module shares the load of the one of the plural-ity of control plane modules and signals the plurality of net-work elements in the data plane to establish flow rules and actions to establish differential routing of flows in the data plane using the control protocol, wherein the control protocol is an OpenFlow protocol, and wherein flow matches are encoded using an extensible match structure in which the flow match is encoded as a type-length-value (TLV).")<br><br>Kempf at [0007] ("A cloud computer system implements a control plane of an evolved packet core (EPC) of a third generation partnership project (3GPP) long term evolution (LTE) net-work. The control plane communicates with the data plane of the EPC that is implemented in a plurality of network ele-ments of the 3GPP LTE network through a control protocol. The EPC with the control plane implemented in the cloud computing |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 509 of 1100

| No. | '111 Patent Claim 24 | Chandrasekaran |
|---|---|---|
| | | system utilizes resources more efficiently than an architecture with the control plane implemented in the plu-rality of network elements of the 3GPP LTE network. The cloud computing system, comprises a controller configured to execute a plurality of control plane modules of the EPC, each control plane module configured to provide a set of control plane functions for managing the data plane and to signal the plurality of network elements in the data plane to establish flow rules and actions to establish differential rout-ing of flows in the data plane using the control protocol, wherein the control protocol is an OpenFlow protocol, and wherein flow matches are encoded using an extensible match structure in which the flow match is encoded as a type-length-value (TLV) and a cloud manager communicatively coupled to the controller. The cloud manager is configured to initialize each of the plurality of control plane modules within the controller as a separate virtual machine, monitor resource utilization of each control plane module and the control plane traffic handled by each control plane module, detect whether a threshold level ofresource utilization or traffic load has been reached by any of the plurality of control plane modules of the EPC, and initialize a new control plane module as a separate virtual machine in response to detecting the threshold level, the new control plane module to share the load of the one of the plurality of control plane modules that exceeded the threshold level.")<br><br>Kempf at [0038] ("Implementing the control plane of an EPC in a cloud computing facility and the data plane of the EPC using a set of OpenFlow switches, as well as managing communication between the control plane and the dataplane using the Open-Flow protocol (e.g., OpenFlow 1.1), creates a problem that the OpenFlow protocol does not support GTP or GTP tunnel endpoint identifier (TEID) routing, which is necessary for implementing the dataplane of the EPC")<br><br>Kempf at [0039] ("The embodiments of the invention overcome these disadvantages of the prior art. The disadvantages of the prior art are avoided by splitting the control plane and the data plane for the EPC architecture and to implement the control plane by deploying the EPC control plane entities in a cloud computing facility, while the data plane is implemented by a distributed collection of OpenFlow switches. The OpenFlow protocol is used to connect the two, with enhancements to support GTP routing. While the EPC architecture already has a split between the control plane and the data plane, in the sense that the serving gateway (S-GW) and the PDN gateway (P-GW) are data plane entities |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 510 of 1100

| No. | '111 Patent Claim 24 | Chandrasekaran |
|---|---|---|
| | | while the MME, PCRF, and home subscriber server (HSS) are control plane entities, this split was made at the level of the mobility management pro-tocol, GTP.")<br><br>Kempf at [0040] ("The standard EPC architecture assumes a standard routed IP network for transport on top of which the mobile network entities and protocols are implemented. The enhanced EPC architecture described herein is instead at the level ofIP routing and media access control (MAC) switch-ing. Instead of using L2 routing and L3 internal gateway protocols to distribute IP routing and managing Ethernet and IP routing as a collection of distributed control entities, L2 and L3 routing management is centralized in a cloud facility and the routing is controlled from the cloud facility using the OpenFlow protocol. As used herein, the "OpenFlow proto-col" refers to the OpenFlow network protocol and switching specification defined in the OpenFlow Switch Specification at www.openflowswitch.org a web site hosted by Stanford Uni-versity. As used herein, an "OpenFlow switch" refers to a network element implementing the OpenFlow protocol.)<br><br>Kempf at [0044] ("FIG. 1 is a diagram of one embodiment of an example network with an OpenFlow switch, conforming to the OpenFlow 1.0 specification. The OpenFlow 1.0 protocol enables a controller 101 to connect to an OpenFlow 1.0 enabled switch 109 using a secure channel 103 and control a single forwarding table 107 in the switch 109. The controller 101 is an external software component executed by a remote computing device that enables a user to configure the Open-Flow 1.0 switch 109. The secure channel 103 can be provided by any type of network including a local area network (LAN) or a wide area network (WAN), such as the Internet.")<br><br>As another example, OpenFlow is a standard protocol used in SDNs to communicate between an OpenFlow switch and controller.<br><br>OpenFlow at 6-7 |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 511 of 1100

| No. | '111 Patent Claim 24 | Chandrasekaran |
|---|---|---|

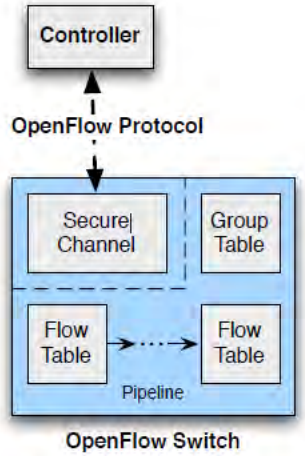



Figure 1: Main components of an OpenFlow switch.

## 2 Switch Components

An OpenFlow Switch consists of one or more *flow tables* and a *group table*, which perform packet lookups and forwarding, and an *OpenFlow channel* to an external controller (Figure 1). The switch communicates with the controller and the controller manages the switch via the OpenFlow protocol.

Using the OpenFlow protocol, the controller can add, update, and delete *flow entries* in flow tables, both reactively (in response to packets) and proactively. Each flow table in the switch contains a set of flow entries; each flow entry consists of *match fields*, *counters*, and a set of *instructions* to apply to matching packets (see 5.2).

Matching starts at the first flow table and may continue to additional flow tables (see 5.1). Flow entries match packets in priority order, with the first matching entry in each table being used (see 5.3). If a matching entry is found, the instructions associated with the specific flow entry are executed. If no match is found in a flow table, the outcome depends on configuration of the table-miss flow entry: for example, the packet may be forwarded to the controller over the OpenFlow channel, dropped, or may continue to the next flow table (see 5.4).

Instructions associated with each flow entry either contain actions or modify pipeline processing (see 5.9). Actions included in instructions describe packet forwarding, packet modification and group table



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 512 of 1100

| No. | '111 Patent Claim 24 | Chandrasekaran |
|---|---|---|
| | | processing. Pipeline processing instructions allow packets to be sent to subsequent tables for further processing and allow information, in the form of metadata, to be communicated between tables. Table pipeline processing stops when the instruction set associated with a matching flow entry does not specify a next table; at this point the packet is usually modified and forwarded (see 5.10).<br><br>Flow entries may forward to a *port*. This is usually a physical port, but it may also be a logical port defined by the switch or a reserved port defined by this specification (see 4.1). Reserved ports may specify generic forwarding actions such as sending to the controller, flooding, or forwarding using non-OpenFlow methods, such as "normal" switch processing (see 4.5), while switch-defined logical ports may specify link aggregation groups, tunnels or loopback interfaces (see 4.4).<br><br>Actions associated with flow entries may also direct packets to a group, which specifies additional processing (see 5.6). Groups represent sets of actions for flooding, as well as more complex forwarding semantics (e.g. multipath, fast reroute, and link aggregation). As a general layer of indirection, groups also enable multiple flow entries to forward to a single identifier (e.g. IP forwarding to a common next hop). This abstraction allows common output actions across flow entries to be changed efficiently.<br><br>The group table contains group entries; each group entry contains a list of *action buckets* with specific semantics dependent on group type (see 5.6.1). The actions in one or more action buckets are applied to packets sent to the group.<br><br>Switch designers are free to implement the internals in any way convenient, provided that correct match and instruction semantics are preserved. For example, while a flow entry may use an all group to forward to multiple ports, a switch designer may choose to implement this as a single bitmask within the hardware forwarding table. Another example is matching; the pipeline exposed by an OpenFlow switch may be physically implemented with a different number of hardware tables. |

| No. | '111 Patent Claim 27 | Chandrasekaran |
|---|---|---|
| 27 | The method according to claim 1, wherein the network node comprises a router, a switch, or a bridge. | Chandrasekaran discloses the method according to claim 1, wherein the network node comprises a router, a switch, or a bridge.<br><br>For example, Chandrasekaran discloses an access point and other nodes that may be routers, switches, gateways, or other network devices.<br><br>*See supra* at Claim 1.<br><br>Chandrasekaran at [0012] ("Referring now to the drawings, and first to FIG.1, an example of a network in which embodiments described herein may be implemented is shown. For |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 513 of 1100

| No. | '111 Patent Claim 27 | Chandrasekaran |
|---|---|---|
| | | simplification, only a small number of network devices are shown. The network includes a wireless controller 12 in communication with a mobile device (client, wireless device, endpoint) 16 through an access point (AP) 14. In the example shown in FIG. 1, the controller 12 is in wired communication with two access points 14 for wireless communication with any number of mobile devices 16 via a wireless network ( e.g., WLAN (wire-less local area network)) at a network site. The wireless con-troller 12 may be in communication with one or more other networks (not shown) (e.g., Internet, intranet, local area net-work, wireless local area network, cellular network, metro-politan area network, wide area network, satellite network, radio access network, public switched network, virtual pri-vate network, or any other network or combination thereof). Communication paths between the wireless controller 12 and other networks or between the controller and access points 14 may include any number or type of intermediate nodes (e.g., routers, switches, gateways, or other network devices), which facilitate passage of data between network devices.")<br><br>Chandrasekaran at [0014] ("The term 'wireless controller' or 'controller' as used herein may refer to a wireless LAN (local area network) controller, mobility controller, wireless control device, wire-less control system, or any other network device operable to perform control functions for a wireless network. The net-work site may also include a wireless control system or other platform for centralized wireless LAN planning, configura-tion, and management. The wireless controller 12 enables system wide functions for wireless applications and may support any number of access points 14. Each access point 14 may serve any number of mobile devices 16 in the wireless network. The wireless controller 12 may be, for example, a standalone device or a rack-mounted appliance. In the example shown in FIG. 1, the wireless controller 12 and access points 14 are separate devices and may be located remote from one another. The wireless controller 12 may also be integrated with the access point 14 ( e.g., autonomous AP) or located at a switch, router, switch/router, or other network device. Thus, the wireless controller 12 may be a physical device located at a standalone device, access point, switch, router, or other network device. The wireless controller 12 may also be a virtual device located in a network or cloud, for example.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 514 of 1100

| No. | '111 Patent Claim 28 | Chandrasekaran |
|---|---|---|
| 28 | The method according to claim 1, wherein the packet network is an Internet Protocol (IP) network, and the packet is an IP packet. | Chandrasekaran discloses the method according to claim 1, wherein the packet network is an Internet Protocol (IP) network, and the packet is an IP packet.<br><br>For example, Chandrasekaran discloses a network and packet that may belong to the Internet, intranet, local area net-work, wireless local area network, cellular network, metro-politan area network, wide area network, satellite network, radio access network, public switched network, virtual pri-vate network, or any other network or combination thereof. A person of ordinary skill in the art would understand that the network may be an Internet Protocol network and that the packet may belong to an Internet Protocol network and be an IP packet.<br><br>*See supra* at Claim 1.<br><br>Chandrasekaran at [0012] ("Referring now to the drawings, and first to FIG.1, an example of a network in which embodiments described herein may be implemented is shown. For simplification, only a small number of network devices are shown. The network includes a wireless controller 12 in communication with a mobile device (client, wireless device, endpoint) 16 through an access point (AP) 14. In the example shown in FIG. 1, the controller 12 is in wired communication with two access points 14 for wireless communication with any number of mobile devices 16 via a wireless network ( e.g., WLAN (wire-less local area network)) at a network site. The wireless con-troller 12 may be in communication with one or more other networks (not shown) (e.g., Internet, intranet, local area net-work, wireless local area network, cellular network, metro-politan area network, wide area network, satellite network, radio access network, public switched network, virtual pri-vate network, or any other network or combination thereof). Communication paths between the wireless controller 12 and other networks or between the controller and access points 14 may include any number or type of intermediate nodes (e.g., routers, switches, gateways, or other network devices), which facilitate passage of data between network devices.")<br><br>Chandrasekaran at [0017] ("The stateful classifier 18 at the controller 12 classi-fies traffic based on multiple packets received from the begin-ning of a flow. Stateful classification uses rules which need information on states for a previous packet ( or packets) in a flow. Stateful classification may be based, for example, on packet pattern matching and decoding |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 515 of 1100

| No. | '111 Patent Claim 28 | Chandrasekaran |
|---|---|---|
| | | of protocols and their states. Stateful classification is also referred to as flow clas-sification since it looks at a data stream of related packets (flow, session).")<br><br>Chandrasekaran at [0018] ("The stateless classifier 22 at the AP 14 uses rules that can act on a per packet basis in the flow. Stateless classifica-tion (also referred to as packet classification) is based on individual packet inspection ( e.g., 5 tuple, pattern matching) without knowledge of any related stream of packets, flows, sessions, or protocols.")<br><br>Chandrasekaran at [0023] ("In one embodiment, the classification information 26 transmitted from the controller 12 to the AP 14 includes tuple information for a flow ( e.g., source IP address, destina-tion IP address, source port, destination port, and protocol), application identifier (ID), and stateless DPI information. Stateless DPI information includes classification and sub-classification information ( e.g., fixed or variable offset with a pattern or regular expression) and rules for applying policies on the sub-classified packets. The policies may include, for example, drop packet, mark a DSCP (Differentiated Services Code Point) value in the packet, or rate limit the traffic.") |

| No. | '111 Patent Claim 29 | Chandrasekaran |
|---|---|---|
| 29 | The method according to claim 28, wherein the packet network is an Transmission Control Protocol (TCP) network, and the packet is an TCP packet. | Chandrasekaran discloses the method according to claim 28, wherein the packet network is an Transmission Control Protocol (TCP) network, and the packet is an TCP packet.<br><br>For example, Chandrasekaran discloses a network and packet that may belong to the Internet, intranet, local area net-work, wireless local area network, cellular network, metro-politan area network, wide area network, satellite network, radio access network, public switched network, virtual pri-vate network, or any other network or combination thereof. A person of ordinary skill in the art would understand that the network may be an Transmission Control Protocol network and that the packet may belong to an Transmission Control Protocol network and be a TCP packet. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Chandrasekaran is found to not meet this limitation, wherein the packet network is an Transmission Control Protocol (TCP) network, and the packet is an TCP packet would have been obvious to a person having ordinary skill in the art, as explained below. |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 516 of 1100

| No. | '111 Patent Claim 29 | Chandrasekaran |
|---|---|---|
| | | *See supra* at Claim 28.<br><br>Chandrasekaran at [0012] ("Referring now to the drawings, and first to FIG.1, an example of a network in which embodiments described herein may be implemented is shown. For simplification, only a small number of network devices are shown. The network includes a wireless controller 12 in communication with a mobile device (client, wireless device, endpoint) 16 through an access point (AP) 14. In the example shown in FIG. 1, the controller 12 is in wired communication with two access points 14 for wireless communication with any number of mobile devices 16 via a wireless network ( e.g., WLAN (wire-less local area network)) at a network site. The wireless con-troller 12 may be in communication with one or more other networks (not shown) (e.g., Internet, intranet, local area net-work, wireless local area network, cellular network, metro-politan area network, wide area network, satellite network, radio access network, public switched network, virtual pri-vate network, or any other network or combination thereof). Communication paths between the wireless controller 12 and other networks or between the controller and access points 14 may include any number or type of intermediate nodes (e.g., routers, switches, gateways, or other network devices), which facilitate passage of data between network devices.")<br><br>Chandrasekaran at [0017] ("The stateful classifier 18 at the controller 12 classi-fies traffic based on multiple packets received from the begin-ning of a flow. Stateful classification uses rules which need information on states for a previous packet ( or packets) in a flow. Stateful classification may be based, for example, on packet pattern matching and decoding of protocols and their states. Stateful classification is also referred to as flow clas-sification since it looks at a data stream of related packets (flow, session).")<br><br>Chandrasekaran at [0018] ("The stateless classifier 22 at the AP 14 uses rules that can act on a per packet basis in the flow. Stateless classifica-tion (also referred to as packet classification) is based on individual packet inspection ( e.g., 5 tuple, pattern matching) without knowledge of any related stream of packets, flows, sessions, or protocols.")<br><br>Chandrasekaran at [0023] ("In one embodiment, the classification information 26 transmitted from the controller 12 to the AP 14 includes tuple information for a flow ( e.g., |

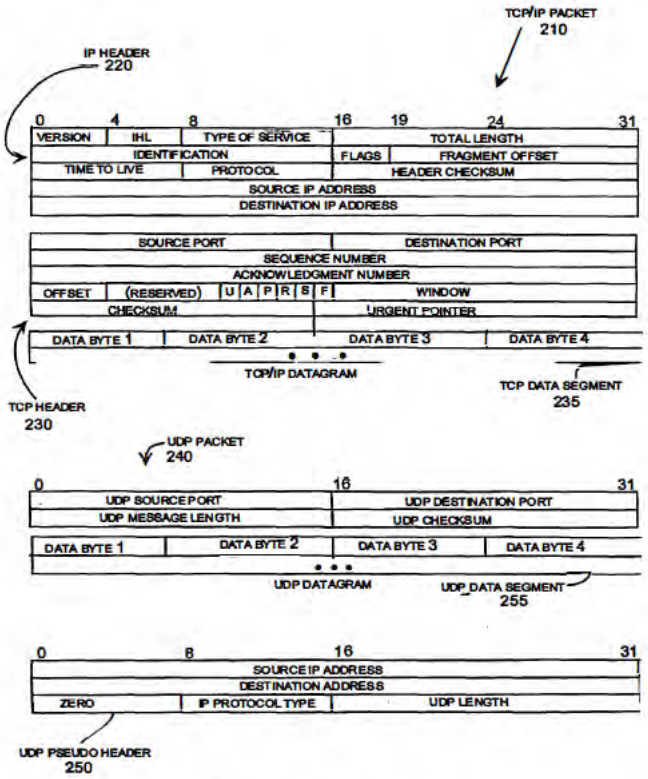


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 517 of 1100

| No. | '111 Patent Claim 29 | Chandrasekaran |
|---|---|---|
| | | source IP address, destina-tion IP address, source port, destination port, and protocol), application identifier (ID), and stateless DPI information. Stateless DPI information includes classification and sub-classification information ( e.g., fixed or variable offset with a pattern or regular expression) and rules for applying policies on the sub-classified packets. The policies may include, for example, drop packet, mark a DSCP (Differentiated Services Code Point) value in the packet, or rate limit the traffic.")<br><br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Chandrasekaran in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 29 of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references.<br><br>For example, Copeland discloses TCP packets.<br><br>Copeland at Figure 2 |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 518 of 1100

| No. | '111 Patent Claim 29 | Chandrasekaran |
|---|---|---|
| | |  <br>FIG. 2<br><br>Copeland at [0076] ("FIG. 2 illustrates an exemplary TCP/IP packet or datagram 210 and an exemplary UDP datagram 240. In a typical TCP/IP packet like 210, each packet typically includes a header portion comprising an IP header 220 and a TCP header 230, followed by a data portion that contains the information to be communicated in the packet. The information in the IP header 220 contained in a TCP/IP packet 210, or any other IP packet, contains the IP addresses and assures that the packet is delivered to the right host. The transport layer protocol (TCP) header follows the Internet protocol header and specifies the port numbers for the associated service.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 519 of 1100

| No. | '111 Patent Claim 29 | Chandrasekaran |
|---|---|---|
| | | Copeland at [0077] ("The header portion in the typical TCP/IP datagram 210 is 40 bytes including 20 bytes of IP header 220 information and 20 bytes of TCP header 230 information. The data portion or segment associated with the packet 210 follows the header information.")<br><br>For example, Chua discloses packet traffic using TCP services.<br><br>Chua at 18:8-44 ("Based on the rule set for a specific device, including what traffic is allowed through, from what zones and expected targets, administrator 114 can infer the right type of test traffic that should be injected for both positive and negative testing of the device. Similarly, the inference can be made to deter-mine the appropriate type of flow troubleshooting that includes the appropriate IP ranges to watch on, and the TCP/ UDP services that the flowchain is meant to service.<br><br>In the case of traffic injection to test a path through SDN 106, path verification unit 136 of SDN controller 112 may perform any of the following tasks:<br><br>Infer from the flowchain the appropriate type of transmission control protocol (TCP) or uniform datagram proto-col (UDP) services that are relevant to the chain, as well as the range of source and target IP addresses for this chain<br><br>Create either UDP packet flows or TCP sessions that confirm to the traffic type inferred or discovered form the flowchain configuration<br><br>Using an SDN device ( e.g., a switch of SDN 106), inject these flows or TCP sessions by temporarily inserting a rule into the rules table on the switch that will take the flows from SDN controller 112 and send them across the devices of SDN 106 (starting from the ingress device in the chain)<br><br>Using an SDN device, insert flow rules to replicate (flow-span) these same traffic streams at each point in the flow chain. The replicated traffic may be directed (via SDN rules) to a collection device ( or to SDN controller 112) that can determine whether the test packets passed through each port on the switch that is part of the flowchain |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 520 of 1100

| No. | '111 Patent Claim 29 | Chandrasekaran |
|---|---|---|
| | | On the final exit device (prior to the end target), insert a flow rule that will drain all injected packets so the final devices (that is, devices external to SDN 106) do not see any of the test traffic.") |

| No. | '111 Patent Claim 30 | Chandrasekaran |
|---|---|---|
| 30[a] | The method according to claim 1, further comprising: receiving, by the network node from the first entity over the packet network, one or more additional packets; | Chandrasekaran discloses the method according to claim 1, further comprising: receiving, by the network node from the first entity over the packet network, one or more additional packets.<br><br>For example, Chandrasekaran discloses data traffic sent from a mobile device over a network to the access point second and additional packets in a flow.<br><br>*See supra at* Claim 1, 1[c].<br><br>Chandrasekaran at [0010] ("In order to provide end-to-end Quality of Service (QoS), policies should be applied to both upstream and down-stream traffic. In wireless networks, this would involve apply-ing policies at both a controller and an access point. Applica-tion classification is needed if the policies are application dependent. However, when a client roams between access points, it may interrupt classification performed at the access point, since classification of the application is based on mul-tiple packets and with roaming, the first packet of the flow may arrive on one access point and the second on another access point.")<br><br>Chandrasekaran at [0019] ("As noted above, stateful classification uses rules which need information on states for previous packets in a flow. When the client 16 roams (as shown in FIG. 1), the first packet of the flow may be received on one AP 14 and the second packet on another AP. Stateful classification is there-fore performed at the controller 12 rather than the AP 14 so that stateful packet inspection is not broken when the client 16 roams. As described below, when the client 16 roams, the controller 12 pushes the same classification rules and policies that it previously sent to the original AP to the new AP.") |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 521 of 1100

| No. | '111 Patent Claim 30 | Chandrasekaran |
|---|---|---|
| | | |
| 30[b] | checking, by the network node, if any one of the one or more additional packets satisfies the criterion; | Chandrasekaran discloses checking, by the network node, if any one of the one or more additional packets satisfies the criterion.<br><br>*See supra at* Claim 1[d], 30[a]. |
| 30[c] | responsive to an additional packet not satisfying the criterion, sending, by the network node over the packet network, the additional packet to the second entity; and | Chandrasekaran discloses responsive to an additional packet not satisfying the criterion, sending, by the network node over the packet network, the additional packet to the second entity.<br><br>*See supra at* Claim 1[e], 30[a]. |
| 30[d] | responsive to the additional packet satisfying the criterion, sending the additional packet, by the network node over the packet network, in response to the instruction. | Chandrasekaran discloses responsive to the additional packet satisfying the criterion, sending the additional packet, by the network node over the packet network, in response to the instruction.<br><br>*See supra at* Claim 1[f], 30[a]. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 522 of 1100

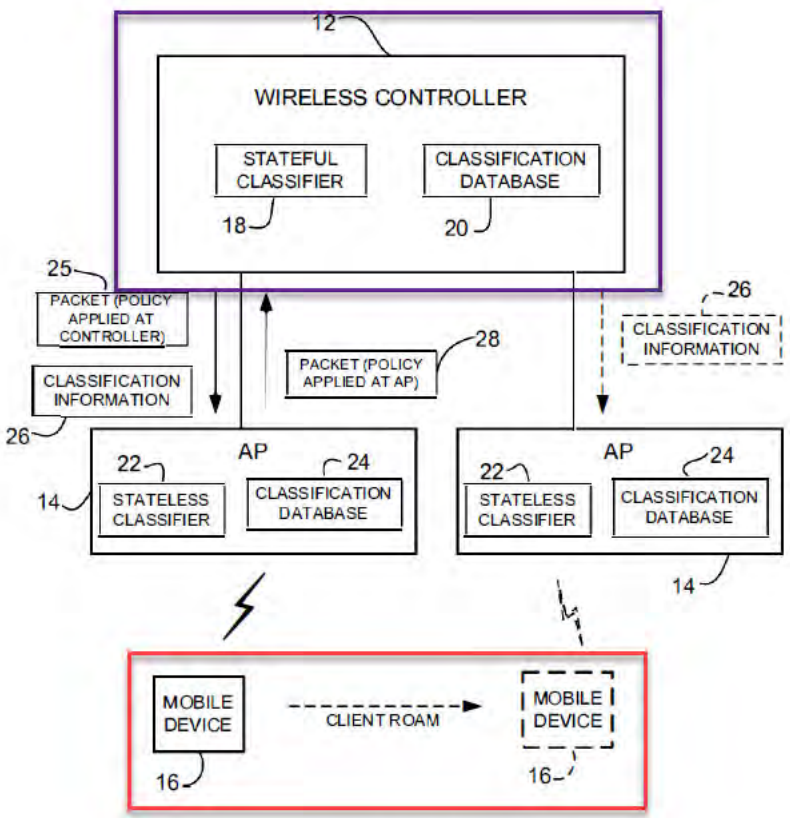
| No. | '111 Patent Claim 31 | Chandrasekaran |
|---|---|---|
| 31[a] | The method according to claim 1, wherein the packet network is a Software Defined Network (SDN), | Chandrasekaran discloses the method according to claim 1, wherein the packet network is a Software Defined Network (SDN).<br><br>For example, Chandrasekaran discloses a wireless controller and access point are configured to communicate to transmit network traffic. A person of ordinary skill in the art would understand the communication between the controller and access point to be a software defined network. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Chandrasekaran is found to not meet this limitation, wherein the packet network is a Software Defined Network (SDN) would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>*See supra* at Claim 1.<br><br>Chandrasekaran at Abstract ("In one embodiment, a method includes performing stateful application classification on packets received at a controller and transmitting classification information to an access point. The classification information includes flow information and stateless rules for applying policies. The access point is con-figured to use the classification information to perform state-less application classification and apply policies to packets received from a mobile device. An apparatus and logic are also disclosed herein.")<br><br>Chandrasekaran at [0007] ("In one embodiment, a method generally comprises performing stateful application classification on packets received at a controller and transmitting classification infor-mation to an access point. The classification information comprises flow information and stateless rules for applying policies. The access point is configured to use the classifica-tion information to perform stateless application classifica-tion and apply policies to packets received from a mobile device.")<br><br>Chandrasekaran at [0012] ("Referring now to the drawings, and first to FIG.1, an example of a network in which embodiments described herein may be implemented is shown. For simplification, only a small number of network devices are shown. The network includes a wireless controller 12 in communication with a mobile device (client, wireless device, endpoint) 16 through an access point (AP) 14. In the example shown in FIG. 1, the controller 12 is in wired communication with two access points 14 for wireless |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 523 of 1100

| No. | '111 Patent Claim 31 | Chandrasekaran |
|---|---|---|
| | | communication with any number of mobile devices 16 via a wireless network ( e.g., WLAN (wire-less local area network)) at a network site. The wireless con-troller 12 may be in communication with one or more other networks (not shown) (e.g., Internet, intranet, local area net-work, wireless local area network, cellular network, metro-politan area network, wide area network, satellite network, radio access network, public switched network, virtual pri-vate network, or any other network or combination thereof). Communication paths between the wireless controller 12 and other networks or between the controller and access points 14 may include any number or type of intermediate nodes (e.g., routers, switches, gateways, or other network devices), which facilitate passage of data between network devices.")<br><br>Chandrasekaran at [0013] ("In one example, the wireless controller 12 receives upstream traffic transmitted from the mobile device 16 and destined for another endpoint ( e.g., host, user device), and transmits downstream traffic received from the endpoint to the mobile device in a communication session. As used herein, the term 'downstream' refers to traffic transmitted from the controller 12 towards the mobile device 16, and the term 'upstream' refers to traffic transmitted from the mobile device towards the controller.")<br><br>Chandrasekaran at [0014] ("The term 'wireless controller' or 'controller' as used herein may refer to a wireless LAN (local area network) controller, mobility controller, wireless control device, wire-less control system, or any other network device operable to perform control functions for a wireless network. The net-work site may also include a wireless control system or other platform for centralized wireless LAN planning, configura-tion, and management. The wireless controller 12 enables system wide functions for wireless applications and may support any number of access points 14. Each access point 14 may serve any number of mobile devices 16 in the wireless network. The wireless controller 12 may be, for example, a standalone device or a rack-mounted appliance. In the example shown in FIG. 1, the wireless controller 12 and access points 14 are separate devices and may be located remote from one another. The wireless controller 12 may also be integrated with the access point 14 ( e.g., autonomous AP) or located at a switch, router, switch/router, or other network device. Thus, the wireless controller 12 may be a physical device located at a standalone device, access point, switch, router, or other network device. The wireless controller 12 may also be a virtual device located in a network or cloud, for example.") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 524 of 1100

| No. | '111 Patent Claim 31 | Chandrasekaran |
|---|---|---|
| | | Chandrasekaran at Figure (annotations added)<br>  <br><br>FIGURE 1<br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Chadrasekaran in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 525 of 1100

| No. | '111 Patent Claim 31 | Chandrasekaran |
|---|---|---|
| | | together) of the references identified in element 31[a] of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references.<br><br>For example, Kempf discloses a packet network using the OpenFlow protocol which is used in Software Defined Networks.<br><br>Kempf at [0004] ("The GPRS tunneling protocol (GTP) is an important communication protocol utilized within the GPRS core net-work. GTP enables end user devices ( e.g., cellular phones) in a GSM network to move from place to place while continuing to connect to the Internet. The end user devices are connected to other devices through a gateway GPRS support node (GGSN). The GGSN tracks the end user device's data from the end user device's serving GPRS support node (GGSN) that is handling the session originating from the end user device.")<br><br>Kempf at [0006] ("A method implements a control plane of an evolved packet core (EPC) of a third generation partnership project (3GPP) long term evolution (LTE) network in a cloud com-puting system. The cloud computing system includes a cloud manager and a controller. The controller executes a plurality of control plane modules. The control plane communicates with the data plane of the EPC implemented in a plurality of network elements of the 3GPP LTE network through a control protocol. The EPC with the control plane implemented in the cloud computing system utilizes resources more efficiently than an architecture with the control plane implemented in the plurality of network elements of the 3GPP LTE network. The method comprises the steps of initializing the plurality of control plane modules of the EPC within the controller. Each control plane module in the plurality of control plane modules is initialized as a separate virtual machine by the cloud man-ager. Each control plane module provides a set of control plane functions for managing the data plane. The cloud man-ager monitors resource utilization of each control plane mod-ule and the control plane traffic handled by each control plane module. The cloud manager detects a threshold level of resource utilization or traffic load for one of the plurality of control plane modules of the EPC. A new control plane mod-ule is initialized as a separate virtual machine by the cloud manager in response to detecting the threshold level. The new control plane module shares the load of the one of the plural-ity of control plane modules and signals the plurality of net-work elements in the data plane to establish flow rules and actions to establish differential routing of flows in the data plane using the control |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 526 of 1100

| No. | '111 Patent Claim 31 | Chandrasekaran |
|---|---|---|
| | | protocol, wherein the control protocol is an OpenFlow protocol, and wherein flow matches are encoded using an extensible match structure in which the flow match is encoded as a type-length-value (TLV).")<br><br>Kempf at [0007] ("A cloud computer system implements a control plane of an evolved packet core (EPC) of a third generation partnership project (3GPP) long term evolution (LTE) net-work. The control plane communicates with the data plane of the EPC that is implemented in a plurality of network ele-ments of the 3GPP LTE network through a control protocol. The EPC with the control plane implemented in the cloud computing system utilizes resources more efficiently than an architecture with the control plane implemented in the plu-rality of network elements of the 3GPP LTE network. The cloud computing system, comprises a controller configured to execute a plurality of control plane modules of the EPC, each control plane module configured to provide a set of control plane functions for managing the data plane and to signal the plurality of network elements in the data plane to establish flow rules and actions to establish differential rout-ing of flows in the data plane using the control protocol, wherein the control protocol is an OpenFlow protocol, and wherein flow matches are encoded using an extensible match structure in which the flow match is encoded as a type-length-value (TLV) and a cloud manager communicatively coupled to the controller. The cloud manager is configured to initialize each of the plurality of control plane modules within the controller as a separate virtual machine, monitor resource utilization of each control plane module and the control plane traffic handled by each control plane module, detect whether a threshold level ofresource utilization or traffic load has been reached by any of the plurality of control plane modules of the EPC, and initialize a new control plane module as a separate virtual machine in response to detecting the threshold level, the new control plane module to share the load of the one of the plurality of control plane modules that exceeded the threshold level.")<br><br>Kempf at [0038] ("Implementing the control plane of an EPC in a cloud computing facility and the data plane of the EPC using a set of OpenFlow switches, as well as managing communication between the control plane and the dataplane using the Open-Flow protocol (e.g., OpenFlow 1.1), creates a problem that the OpenFlow protocol does not support GTP or GTP tunnel endpoint identifier (TEID) routing, which is necessary for implementing the dataplane of the EPC") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 527 of 1100

| No. | '111 Patent Claim 31 | Chandrasekaran |
|---|---|---|
| | | Kempf at [0039] ("The embodiments of the invention overcome these disadvantages of the prior art. The disadvantages of the prior art are avoided by splitting the control plane and the data plane for the EPC architecture and to implement the control plane by deploying the EPC control plane entities in a cloud computing facility, while the data plane is implemented by a distributed collection of OpenFlow switches. The OpenFlow protocol is used to connect the two, with enhancements to support GTP routing. While the EPC architecture already has a split between the control plane and the data plane, in the sense that the serving gateway (S-GW) and the PDN gateway (P-GW) are data plane entities while the MME, PCRF, and home subscriber server (HSS) are control plane entities, this split was made at the level of the mobility management pro-tocol, GTP.")<br><br>Kempf at [0040] ("The standard EPC architecture assumes a standard routed IP network for transport on top of which the mobile network entities and protocols are implemented. The enhanced EPC architecture described herein is instead at the level ofIP routing and media access control (MAC) switch-ing. Instead of using L2 routing and L3 internal gateway protocols to distribute IP routing and managing Ethernet and IP routing as a collection of distributed control entities, L2 and L3 routing management is centralized in a cloud facility and the routing is controlled from the cloud facility using the OpenFlow protocol. As used herein, the "OpenFlow proto-col" refers to the OpenFlow network protocol and switching specification defined in the OpenFlow Switch Specification at www.openflowswitch.org a web site hosted by Stanford Uni-versity. As used herein, an "OpenFlow switch" refers to a network element implementing the OpenFlow protocol.)<br><br>Kempf at [0044] ("FIG. 1 is a diagram of one embodiment of an example network with an OpenFlow switch, conforming to the OpenFlow 1.0 specification. The OpenFlow 1.0 protocol enables a controller 101 to connect to an OpenFlow 1.0 enabled switch 109 using a secure channel 103 and control a single forwarding table 107 in the switch 109. The controller 101 is an external software component executed by a remote computing device that enables a user to configure the Open-Flow 1.0 switch 109. The secure channel 103 can be provided by any type of network including a local area network (LAN) or a wide area network (WAN), such as the Internet.")<br><br>As another example, Chua discloses techniques and methods related to software defined networks (SDNs). |


~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 528 of 1100

| No. | '111 Patent Claim 31 | Chandrasekaran |
|---|---|---|
| | | Chua at 1:45-55 ("In general, this disclosure describes techniques related to controlling software defined networks (SDNs). A software defined network is generally a network of interconnected computing devices having forwarding planes or data planes that can be programmed remotely by one or more controller devices. In this manner, the control plane can be physically separate from the data plane ( or forwarding plane) for an SDN. These computing devices can have either physical instantiation or virtual (software-only) instantiation without the presence of a hardware appliance. This disclosure describes various techniques related to controlling SDNs.")<br><br>Chua at 1:56-63 ("In one example, a method includes determining, by a con-troller device for a software defined network, connections between network devices in the software defined network, determining, by the controller device, one or more paths for network traffic between the network devices based on the determination of the connections, and programming, by the controller device, the network devices to direct network traf-fic along the one or more paths.")<br><br>Chua at 2:14-20 ("In another example, a method includes programming, by a controller device for a software defined network (SDN), a first network device of the SDN to send packets of a packet flow to a service device, and programming, by the controller device, one or more network devices of the SDN to perform a programmed action on packets of the packet flow based on data received from the service device for the packet flow.")<br><br>Chua at 2:38-48 ("In another example, a method includes programming, by a controller device for a software defined network (SDN), a set of network devices of the SDN to form a path through the SDN and to send data representative of packets sent along the path to the controller device, sending, by the controller device, packets of a packet flow corresponding to the path to one of the set of network devices, determining, by the controller device, whether the set of network devices is properly forwarding the packets of the packet flow along the path based on data received from the set of network devices, and present-ing a report representative of the determination.")<br><br>Chua at 5:50-6:5 ("SDN 106 generally serves to interconnect various endpoint devices, such as client device 102 and server device 104. In addition, SDN 106 may provide services to network traffic flowing between client device 102 and server device 104. Alternatively, |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 529 of 1100

| No. | '111 Patent Claim 31 | Chandrasekaran |
|---|---|---|
| | | SDN 106 may provide services to client device 102, without further directing traffic to server device 106. For example, administrator 114 may use SDN controller 112 to program network devices of SDN 106 to direct network traffic for client device 102 to one or more of service devices 116. Service devices 116 may include, for example, intrusion detection service (IDS) devices, intrusion prevention system (IPS) devices, web proxies, web servers, web-application firewalls and the like. In other examples, service devices 116 may, additionally or alternatively, include devices for provid-ing services such as, for example, denial of service (DoS) protection, distributed denial of service (DDoS) protection, traffic filtering, wide area network (WAN) acceleration, or other such services. Service devices 116 may also addition-ally or alternatively include malware detection devices, net-work anti-virus devices, network packet capture and analysis devices, honeypot devices, reflector net devices, tar pit devices, domain name service (DNS) and global DNS server devices, mail proxies, and anti-spam devices.") |
| 31[b] | the packet is routed as part of a data plane and | Chandrasekaran discloses the packet is routed as part of a data plane.<br><br>For example, Chandrasekaran routing packets in a network utilizing a controller and access point. A person of ordinary skill in the art would understand that the packet is routed over a data plane. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Chandrasekaran is found to not meet this limitation, the packet is routed as part of a data plane would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>*See supra* at Claim 1.<br><br>Chandrasekaran at Abstract ("In one embodiment, a method includes performing stateful application classification on packets received at a controller and transmitting classification information to an access point. The classification information includes flow information and stateless rules for applying policies. The access point is con-figured to use the classification information to perform state-less application classification and apply policies to packets received from a mobile device. An apparatus and logic are also disclosed herein.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 530 of 1100

| No. | '111 Patent Claim 31 | Chandrasekaran |
|---|---|---|
| | | Chandrasekaran at [0007] ("In one embodiment, a method generally comprises performing stateful application classification on packets received at a controller and transmitting classification infor-mation to an access point. The classification information comprises flow information and stateless rules for applying policies. The access point is configured to use the classifica-tion information to perform stateless application classifica-tion and apply policies to packets received from a mobile device.")<br><br>Chandrasekaran at [0012] ("Referring now to the drawings, and first to FIG.1, an example of a network in which embodiments described herein may be implemented is shown. For simplification, only a small number of network devices are shown. The network includes a wireless controller 12 in communication with a mobile device (client, wireless device, endpoint) 16 through an access point (AP) 14. In the example shown in FIG. 1, the controller 12 is in wired communication with two access points 14 for wireless communication with any number of mobile devices 16 via a wireless network ( e.g., WLAN (wire-less local area network)) at a network site. The wireless con-troller 12 may be in communication with one or more other networks (not shown) (e.g., Internet, intranet, local area net-work, wireless local area network, cellular network, metro-politan area network, wide area network, satellite network, radio access network, public switched network, virtual pri-vate network, or any other network or combination thereof). Communication paths between the wireless controller 12 and other networks or between the controller and access points 14 may include any number or type of intermediate nodes (e.g., routers, switches, gateways, or other network devices), which facilitate passage of data between network devices.")<br><br>Chandrasekaran at [0013] ("In one example, the wireless controller 12 receives upstream traffic transmitted from the mobile device 16 and destined for another endpoint ( e.g., host, user device), and transmits downstream traffic received from the endpoint to the mobile device in a communication session. As used herein, the term 'downstream' refers to traffic transmitted from the controller 12 towards the mobile device 16, and the term 'upstream' refers to traffic transmitted from the mobile device towards the controller.")<br><br>Chandrasekaran at [0014] ("The term 'wireless controller' or 'controller' as used herein may refer to a wireless LAN (local area network) controller, mobility controller, wireless control device, wire-less control system, or any other network device operable to perform control |

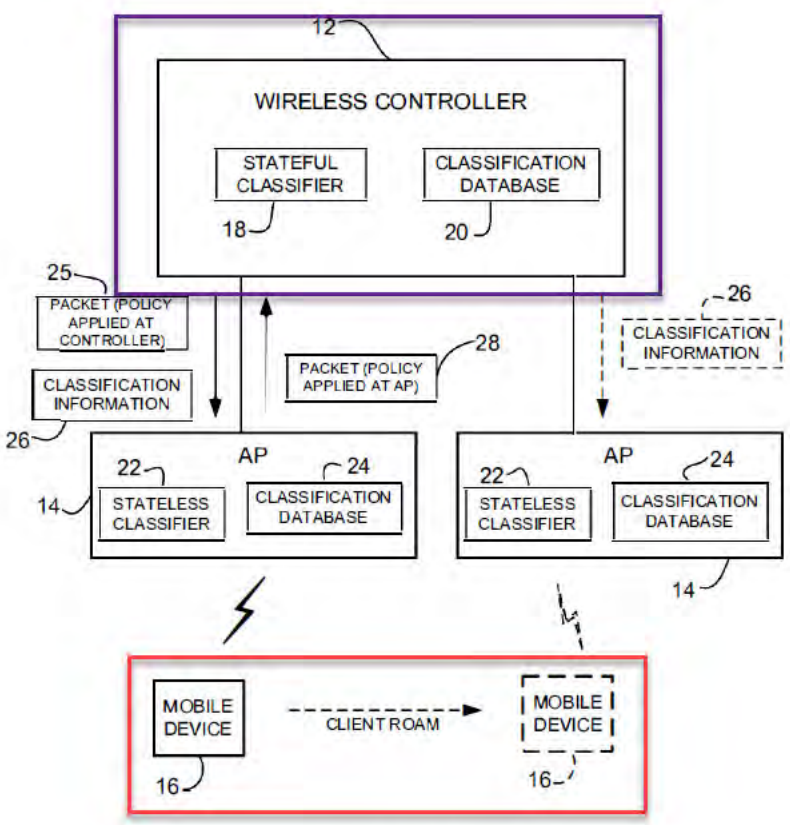


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 531 of 1100

| No. | '111 Patent Claim 31 | Chandrasekaran |
|---|---|---|
| | | functions for a wireless network. The net-work site may also include a wireless control system or other platform for centralized wireless LAN planning, configura-tion, and management. The wireless controller 12 enables system wide functions for wireless applications and may support any number of access points 14. Each access point 14 may serve any number of mobile devices 16 in the wireless network. The wireless controller 12 may be, for example, a standalone device or a rack-mounted appliance. In the example shown in FIG. 1, the wireless controller 12 and access points 14 are separate devices and may be located remote from one another. The wireless controller 12 may also be integrated with the access point 14 ( e.g., autonomous AP) or located at a switch, router, switch/router, or other network device. Thus, the wireless controller 12 may be a physical device located at a standalone device, access point, switch, router, or other network device. The wireless controller 12 may also be a virtual device located in a network or cloud, for example.")<br><br>Chandrasekaran at Figure (annotations added) |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 532 of 1100

| No. | '111 Patent Claim 31 | Chandrasekaran |
|---|---|---|
| | |  <br>FIGURE 1<br><br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Chandrasekaran in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 31[b] of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 533 of 1100

| No. | '111 Patent Claim 31 | Chandrasekaran |
|---|---|---|
| | | For example, Kempf discloses routing packets on a data plane using a control protocol.<br><br>Kempf at [0006] ("A method implements a control plane of an evolved packet core (EPC) of a third generation partnership project (3GPP) long term evolution (LTE) network in a cloud com-puting system. The cloud computing system includes a cloud manager and a controller. The controller executes a plurality of control plane modules. The control plane communicates with the data plane of the EPC implemented in a plurality of network elements of the 3GPP LTE network through a control protocol. The EPC with the control plane implemented in the cloud computing system utilizes resources more efficiently than an architecture with the control plane implemented in the plurality of network elements of the 3GPP LTE network. The method comprises the steps of initializing the plurality of control plane modules of the EPC within the controller. Each control plane module in the plurality of control plane modules is initialized as a separate virtual machine by the cloud man-ager. Each control plane module provides a set of control plane functions for managing the data plane. The cloud man-ager monitors resource utilization of each control plane mod-ule and the control plane traffic handled by each control plane module. The cloud manager detects a threshold level of resource utilization or traffic load for one of the plurality of control plane modules of the EPC. A new control plane mod-ule is initialized as a separate virtual machine by the cloud manager in response to detecting the threshold level. The new control plane module shares the load of the one of the plural-ity of control plane modules and signals the plurality of net-work elements in the data plane to establish flow rules and actions to establish differential routing of flows in the data plane using the control protocol, wherein the control protocol is an OpenFlow protocol, and wherein flow matches are encoded using an extensible match structure in which the flow match is encoded as a type-length-value (TLV).")<br><br>Kempf at [0007] ("A cloud computer system implements a control plane of an evolved packet core (EPC) of a third generation partnership project (3GPP) long term evolution (LTE) net-work. The control plane communicates with the data plane of the EPC that is implemented in a plurality of network ele-ments of the 3GPP LTE network through a control protocol. The EPC with the control plane implemented in the cloud computing system utilizes resources more efficiently than an architecture with the control plane implemented in the plu-rality of network elements of the 3GPP LTE network. The cloud |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 534 of 1100

| No. | '111 Patent Claim 31 | Chandrasekaran |
|---|---|---|
| | | computing system, comprises a controller configured to execute a plurality of control plane modules of the EPC, each control plane module configured to provide a set of control plane functions for managing the data plane and to signal the plurality of network elements in the data plane to establish flow rules and actions to establish differential rout-ing of flows in the data plane using the control protocol, wherein the control protocol is an OpenFlow protocol, and wherein flow matches are encoded using an extensible match structure in which the flow match is encoded as a type-length-value (TLV) and a cloud manager communicatively coupled to the controller. The cloud manager is configured to initialize each of the plurality of control plane modules within the controller as a separate virtual machine, monitor resource utilization of each control plane module and the control plane traffic handled by each control plane module, detect whether a threshold level ofresource utilization or traffic load has been reached by any of the plurality of control plane modules of the EPC, and initialize a new control plane module as a separate virtual machine in response to detecting the threshold level, the new control plane module to share the load of the one of the plurality of control plane modules that exceeded the threshold level.")<br><br>Kempf at [0038] ("Implementing the control plane of an EPC in a cloud computing facility and the data plane of the EPC using a set of OpenFlow switches, as well as managing communication between the control plane and the dataplane using the Open-Flow protocol (e.g., OpenFlow 1.1), creates a problem that the OpenFlow protocol does not support GTP or GTP tunnel endpoint identifier (TEID) routing, which is necessary for implementing the dataplane of the EPC")<br><br>Kempf at [0039] ("The embodiments of the invention overcome these disadvantages of the prior art. The disadvantages of the prior art are avoided by splitting the control plane and the data plane for the EPC architecture and to implement the control plane by deploying the EPC control plane entities in a cloud computing facility, while the data plane is implemented by a distributed collection of OpenFlow switches. The OpenFlow protocol is used to connect the two, with enhancements to support GTP routing. While the EPC architecture already has a split between the control plane and the data plane, in the sense that the serving gateway (S-GW) and the PDN gateway (P-GW) are data plane entities while the MME, PCRF, and home subscriber server (HSS) are control plane entities, this split was made at the level of the mobility management pro-tocol, GTP.") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 535 of 1100

| No. | '111 Patent Claim 31 | Chandrasekaran |
|---|---|---|
| | | Kempf at [0040] ("The standard EPC architecture assumes a standard routed IP network for transport on top of which the mobile network entities and protocols are implemented. The enhanced EPC architecture described herein is instead at the level ofIP routing and media access control (MAC) switch-ing. Instead of using L2 routing and L3 internal gateway protocols to distribute IP routing and managing Ethernet and IP routing as a collection of distributed control entities, L2 and L3 routing management is centralized in a cloud facility and the routing is controlled from the cloud facility using the OpenFlow protocol. As used herein, the "OpenFlow proto-col" refers to the OpenFlow network protocol and switching specification defined in the OpenFlow Switch Specification at www.openflowswitch.org a web site hosted by Stanford Uni-versity. As used herein, an "OpenFlow switch" refers to a network element implementing the OpenFlow protocol.)<br><br>Kempf at [0041] ("The standard EPC control plane entities-the MME, PCRF, and HSS-are likewise deployed in the cloud, along with the control plane parts of the S-GW and P-GW, namely, the S-GW-C and the P-GW-C. The data plane con-sists of standard OpenFlow switches with enhancements as needed for routing GTP packets, rather than IP routers and Ethernet switches. At a minimum, the data plane parts of the S-GW and P-GW, namely, the S-GW-Dand the P-GW-D, and the packet routing part of the E-NodeB in the E-UTRAN require OpenFlow enhancements for GTP routing. Addi-tional enhancements for GTP routing may be needed on other switches within the EPC architecture depending on how much fine grained control over the routing an operator requires.")<br><br>Kempf at [0078] ("FIG. 15 is a diagram of one embodiment of how the EPC in the cloud computing system enables a managed ser-vices company to manage multiple operator networks out of a single data center. The managed services cloud computing facility 1501 runs separate instances of the EPC control plane for every mobile operator with which the managed services company has a contract. Each EPC instance is in a VPC 1503A,B that isolates the mobile operator's traffic from other tenants in the cloud computing facility 1501 of the data cen-ter. The EPC control plane instance for a mobile operator is connected to the mobile operator's geographically distributed EPC OpenFlow data plane switching fabric 1507 A,B and the mobile operator's base stations through a virtual edge router 1509A,B. The virtual edge router 1509A,B routes traffic from the data center to and from the appropriate mobile operator EPC data plane switching fabric 1507 A,B. In some cases, the mobile operators may even share base stations and EPC switching fabrics, though the |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 536 of 1100

| No. | '111 Patent Claim 31 | Chandrasekaran |
|---|---|---|
| | | example embodiment in FIG. 15 shows a case where the two mobile operators have separate switching fabrics.")<br><br>Kempf at [0087] ("In one embodiment, slow path support for GTP is implemented with an OpenFlow gateway switch. An Open-Flow mobile gateway switch also contains support on the software control plane for slow path packet processing. This path is taken by G-PDU (message type 255) packets with nonzero header fields or extension headers, and user data plane packets requiring encapsulation with such fields or addition of extension headers, and by G TP-U control packets. For this purpose, the switch supports three local ports in the software control plane: LOCAL_GTP _CONTROL-the switch fast path forwards GTP encapsulated packets directed to the gateway IP address that contain GTP-U control mes-sages and the local switch software control plane initiates local control plane actions depending on the GTP-U control message; LOCAL_GTP _U_DECAP-the switch fast path forwards G-PDU packets to this port that have nonzero header fields or extension headers (i.e. E!=0, S!=0, or PN!=0). These packets require specialized handling. The local switch software slow path processes the packets and performs the specialized handling; and LOCAL_GTP _U_ENCAP-the switch fast path forwards user data plane packets to this port that require encapsulation in a GTP tunnel with nonzero header fields or extension headers (i.e. E!=0, S!=0, or PN!=0). These packets require specialized handling. The local switch software slow path encapsulates the packets and performs the specialized handling. In addition to forwarding the packet, the switch fast path makes the OpenFlow metadata field avail-able to the slow path software.")<br><br>Kempf at [0093] ("The virtual port simply removes the GTP tunnel header and forwards the enclosed user data plane packet out the bound physical port.")<br><br>Kempf at [0101] ("In one embodiment, the system implements han-dling of user data plane packets requiring GTP-U encapsula-tion with extension headers, sequence numbers, and N-PDU numbers. User data plane packets that require extension head-ers, sequence numbers, or N-PDU numbers during GTP encapsulation require special handling by the software slow path. For these packets, the OpenFlow controller programs a rule matching the 4 tuple: IP source address; IP destination address; UDP/TCP/SCTP source port; and UDP/TCP/SCTP destination port. The instructions for matching packets are: |


~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 537 of 1100

| No. | '111 Patent Claim 31 | Chandrasekaran |
|---|---|---|
| | | Write-Metadata ( GTP-TEID, 0x FFFFFFFF)<br>Apply-Actions (Set-Output-Port LOCAL_GTP _U_ENCAP )")<br><br>Kempf at [0145] ("In other embodiments, other control protocols can be utilized in place of OpenFlow as described herein. The use of OpenFlow is presented by way of example and not limita-tion. Other control protocols can also be utilized to manage the communication between the control plane and data plane and configuration of the data plane of the split EPC architec-ture. An example of such a protocol is FORCES, an IETF standard protocol for splitting the control plane and forward-ing plane in networks. The FORCES protocol specification is described in RFC 5810. RFC 5812 describes the architecture of a FORCES forwarding element, the equivalent of an Open-Flow switch. The FORCES protocol itself does not directly support programming routes into the forwarding element, it is, instead, a framework for handling the interaction between the FORCES controller and a FORCES forwarding element. The forwarding element architecture describes how to design the protocol that actually allows a FORCES controller to program a FORCES forwarding element. One skilled in the art would understand that a FORCES based system could include features described herein above in relation to the OpenFlow embodiment, such as the GTP OpenFlow exten-sion, to allow the controller to program the switches for GTP TEID routing.")<br><br>As another example, Chua discloses forwarding packets over a data plane to various network destinations.<br><br>Chua at 1:45-55 ("In general, this disclosure describes techniques related to controlling software defined networks (SDNs). A software defined network is generally a network of interconnected computing devices having forwarding planes or data planes that can be programmed remotely by one or more controller devices. In this manner, the control plane can be physically separate from the data plane ( or forwarding plane) for an SDN. These computing devices can have either physical instantiation or virtual (software-only) instantiation without the presence of a hardware appliance. This disclosure describes various techniques related to controlling SDNs.")<br><br>Chua at 1:56-63 ("In one example, a method includes determining, by a con-troller device for a software defined network, connections between network devices in the software defined |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 538 of 1100

| No. | '111 Patent Claim 31 | Chandrasekaran |
|---|---|---|
| | | network, determining, by the controller device, one or more paths for network traffic between the network devices based on the determination of the connections, and programming, by the controller device, the network devices to direct network traf-fic along the one or more paths.")<br><br>Chua at 23:22-34 ("FIG. 4 illustrates various devices and services organized according to the "control plane" and the "data plane." In general, devices and services of the control plane manage devices of the data plane to cause the devices of the data plane to forward data traffic between various network destinations. In conventional routers, each router includes functionality for both the control plane and the data plane, and the same is true for conventional switches. However, in accordance with the techniques of this disclosure, the control plane can be entirely separated from the data plane, such that an SDN controller, such as SDN controller 112, can program devices of the data plane, such as network switches, to perform the techniques of this disclosure.")<br><br>Chua at 23:35:45 ("FIG. 4 is a conceptual diagram illustrating an example flow management system 250 including various components that may operate in accordance with the techniques of this disclo-sure. Flow management system 250 (also referred to as "sys-tem 250") includes control plane 252 and data plane 280. In general, control plane 252 includes components that relate to control information, e.g., routing information relating to packet flows and paths through an SDN. Data plane 280 generally includes components that send, forward, and/or receive data in accordance with control information from components of control plane 252.")<br><br>Chua at 24:20-36 ("In accordance with the techniques of this disclosure, flow management server 256 programs network switches 282, based on connections between network switches 282, to form paths through an SDN. For example, flow management server 256 may program network switches 282 to establish a path between TCP client 284 and server 288, and/or a path between TCP client 284 and multicast source 286. In some examples, flow management server 256 may program net-work switches 282 to define multiple paths, e.g., a primary path and one or more backup paths, as discussed above. Likewise, flow management server 256 may send test traffic through network switches 282 to test one or more of the paths. Data plane 280 may include one or more service devices (such as web proxy devices, IDS devices, and/or web serv-ers), to which network switches 282 may direct network packets. Server 288 may represent a service device of an SDN controlled by control plane 252, in some examples.") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 539 of 1100

| No. | '111 Patent Claim 31 | Chandrasekaran |
|---|---|---|
| | | |
| 31[c] | the network node communication with the controller serves as a control plane. | Chandrasekaran discloses the network node communication with the controller serves as a control plane.<br><br>For example, Chandrasekaran discloses a wireless controller and access point are configured to communicate to transmit network traffic. A person of ordinary skill in the art would understand the communication between the controller and access point occurs over a control plane. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Chandrasekaran is found to not meet this limitation, the network node communication with the controller serves as a control plane would have been obvious to a person having ordinary skill in the art, as explained below<br><br>Chandrasekaran at Abstract ("In one embodiment, a method includes performing stateful application classification on packets received at a controller and transmitting classification information to an access point. The classification information includes flow information and stateless rules for applying policies. The access point is con-figured to use the classification information to perform state-less application classification and apply policies to packets received from a mobile device. An apparatus and logic are also disclosed herein.")<br><br>Chandrasekaran at [0007] ("In one embodiment, a method generally comprises performing stateful application classification on packets received at a controller and transmitting classification infor-mation to an access point. The classification information comprises flow information and stateless rules for applying policies. The access point is configured to use the classifica-tion information to perform stateless application classifica-tion and apply policies to packets received from a mobile device.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 540 of 1100

| No. | '111 Patent Claim 31 | Chandrasekaran |
|---|---|---|
| | | Chandrasekaran at [0012] ("Referring now to the drawings, and first to FIG.1, an example of a network in which embodiments described herein may be implemented is shown. For simplification, only a small number of network devices are shown. The network includes a wireless controller 12 in communication with a mobile device (client, wireless device, endpoint) 16 through an access point (AP) 14. In the example shown in FIG. 1, the controller 12 is in wired communication with two access points 14 for wireless communication with any number of mobile devices 16 via a wireless network ( e.g., WLAN (wire-less local area network)) at a network site. The wireless con-troller 12 may be in communication with one or more other networks (not shown) (e.g., Internet, intranet, local area net-work, wireless local area network, cellular network, metro-politan area network, wide area network, satellite network, radio access network, public switched network, virtual pri-vate network, or any other network or combination thereof). Communication paths between the wireless controller 12 and other networks or between the controller and access points 14 may include any number or type of intermediate nodes (e.g., routers, switches, gateways, or other network devices), which facilitate passage of data between network devices.")<br><br>Chandrasekaran at [0013] ("In one example, the wireless controller 12 receives upstream traffic transmitted from the mobile device 16 and destined for another endpoint ( e.g., host, user device), and transmits downstream traffic received from the endpoint to the mobile device in a communication session. As used herein, the term 'downstream' refers to traffic transmitted from the controller 12 towards the mobile device 16, and the term 'upstream' refers to traffic transmitted from the mobile device towards the controller.")<br><br>Chandrasekaran at [0014] ("The term 'wireless controller' or 'controller' as used herein may refer to a wireless LAN (local area network) controller, mobility controller, wireless control device, wire-less control system, or any other network device operable to perform control functions for a wireless network. The net-work site may also include a wireless control system or other platform for centralized wireless LAN planning, configura-tion, and management. The wireless controller 12 enables system wide functions for wireless applications and may support any number of access points 14. Each access point 14 may serve any number of mobile devices 16 in the wireless network. The wireless controller 12 may be, for example, a standalone device or a rack-mounted appliance. In the example, |

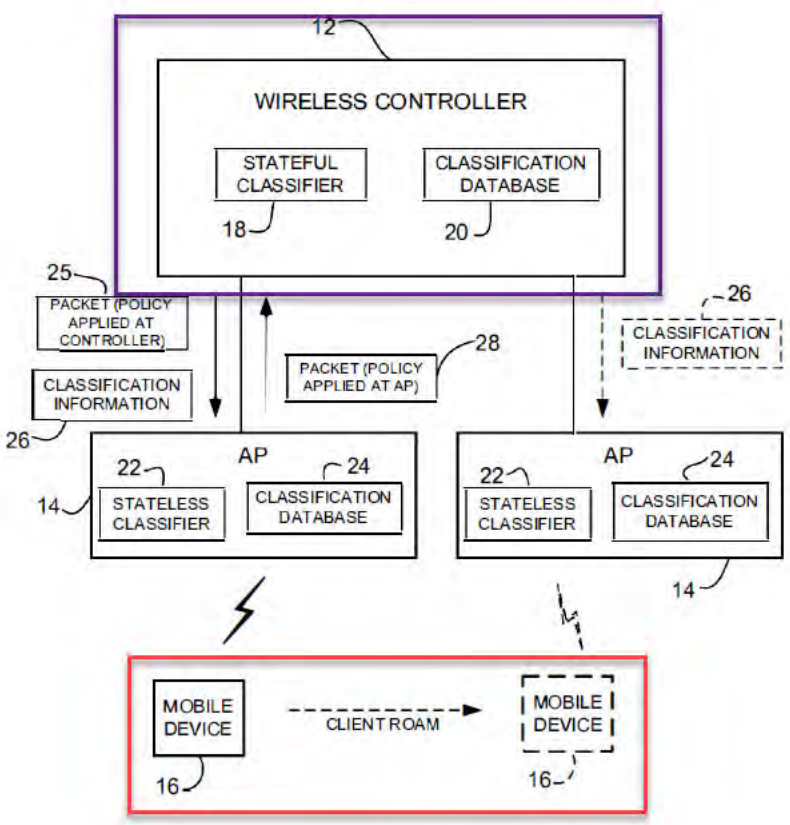


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 541 of 1100

| No. | '111 Patent Claim 31 | Chandrasekaran |
|---|---|---|
| | | shown in FIG. 1, the wireless controller 12 and access points 14 are separate devices and may be located remote from one another. The wireless controller 12 may also be integrated with the access point 14 ( e.g., autonomous AP) or located at a switch, router, switch/router, or other network device. Thus, the wireless controller 12 may be a physical device located at a standalone device, access point, switch, router, or other network device. The wireless controller 12 may also be a virtual device located in a network or cloud, for example.")<br><br>Chandrasekaran at Figure (annotations added) |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 542 of 1100

| No. | '111 Patent Claim 31 | Chandrasekaran |
|---|---|---|
| | |  <br>FIGURE 1<br><br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Chandrasekaran in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 31[c] of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 543 of 1100

| No. | '111 Patent Claim 31 | Chandrasekaran |
|---|---|---|
| | | For example, Kempf discloses communication between network elements and an OpenFlow controller over a control plane.<br><br>Kempf at [0006] ("A method implements a control plane of an evolved packet core (EPC) of a third generation partnership project (3GPP) long term evolution (LTE) network in a cloud com-puting system. The cloud computing system includes a cloud manager and a controller. The controller executes a plurality of control plane modules. The control plane communicates with the data plane of the EPC implemented in a plurality of network elements of the 3GPP LTE network through a control protocol. The EPC with the control plane implemented in the cloud computing system utilizes resources more efficiently than an architecture with the control plane implemented in the plurality of network elements of the 3GPP LTE network. The method comprises the steps of initializing the plurality of control plane modules of the EPC within the controller. Each control plane module in the plurality of control plane modules is initialized as a separate virtual machine by the cloud man-ager. Each control plane module provides a set of control plane functions for managing the data plane. The cloud man-ager monitors resource utilization of each control plane mod-ule and the control plane traffic handled by each control plane module. The cloud manager detects a threshold level of resource utilization or traffic load for one of the plurality of control plane modules of the EPC. A new control plane mod-ule is initialized as a separate virtual machine by the cloud manager in response to detecting the threshold level. The new control plane module shares the load of the one of the plural-ity of control plane modules and signals the plurality of net-work elements in the data plane to establish flow rules and actions to establish differential routing of flows in the data plane using the control protocol, wherein the control protocol is an OpenFlow protocol, and wherein flow matches are encoded using an extensible match structure in which the flow match is encoded as a type-length-value (TLV).")<br><br>Kempf at [0007] ("A cloud computer system implements a control plane of an evolved packet core (EPC) of a third generation partnership project (3GPP) long term evolution (LTE) net-work. The control plane communicates with the data plane of the EPC that is implemented in a plurality of network ele-ments of the 3GPP LTE network through a control protocol. The EPC with the control plane implemented in the cloud computing system utilizes resources more efficiently than an architecture with the control plane |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 544 of 1100

| No. | '111 Patent Claim 31 | Chandrasekaran |
|---|---|---|
| | | implemented in the plu-rality of network elements of the 3GPP LTE network. The cloud computing system, comprises a controller configured to execute a plurality of control plane modules of the EPC, each control plane module configured to provide a set of control plane functions for managing the data plane and to signal the plurality of network elements in the data plane to establish flow rules and actions to establish differential rout-ing of flows in the data plane using the control protocol, wherein the control protocol is an OpenFlow protocol, and wherein flow matches are encoded using an extensible match structure in which the flow match is encoded as a type-length-value (TLV) and a cloud manager communicatively coupled to the controller. The cloud manager is configured to initialize each of the plurality of control plane modules within the controller as a separate virtual machine, monitor resource utilization of each control plane module and the control plane traffic handled by each control plane module, detect whether a threshold level ofresource utilization or traffic load has been reached by any of the plurality of control plane modules of the EPC, and initialize a new control plane module as a separate virtual machine in response to detecting the threshold level, the new control plane module to share the load of the one of the plurality of control plane modules that exceeded the threshold level.")<br><br>Kempf at [0038] ("Implementing the control plane of an EPC in a cloud computing facility and the data plane of the EPC using a set of OpenFlow switches, as well as managing communication between the control plane and the dataplane using the Open-Flow protocol (e.g., OpenFlow 1.1), creates a problem that the OpenFlow protocol does not support GTP or GTP tunnel endpoint identifier (TEID) routing, which is necessary for implementing the dataplane of the EPC")<br><br>Kempf at [0039] ("The embodiments of the invention overcome these disadvantages of the prior art. The disadvantages of the prior art are avoided by splitting the control plane and the data plane for the EPC architecture and to implement the control plane by deploying the EPC control plane entities in a cloud computing facility, while the data plane is implemented by a distributed collection of OpenFlow switches. The OpenFlow protocol is used to connect the two, with enhancements to support GTP routing. While the EPC architecture already has a split between the control plane and the data plane, in the sense that the serving gateway (S-GW) and the PDN gateway (P-GW) are data plane entities while the MME, PCRF, and home subscriber server (HSS) are control plane entities, this split was made at the level of the mobility management pro-tocol, GTP.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 545 of 1100

| No. | '111 Patent Claim 31 | Chandrasekaran |
|---|---|---|
| | | Kempf at [0040] ("The standard EPC architecture assumes a standard routed IP network for transport on top of which the mobile network entities and protocols are implemented. The enhanced EPC architecture described herein is instead at the level ofIP routing and media access control (MAC) switch-ing. Instead of using L2 routing and L3 internal gateway protocols to distribute IP routing and managing Ethernet and IP routing as a collection of distributed control entities, L2 and L3 routing management is centralized in a cloud facility and the routing is controlled from the cloud facility using the OpenFlow protocol. As used herein, the "OpenFlow proto-col" refers to the OpenFlow network protocol and switching specification defined in the OpenFlow Switch Specification at www.openflowswitch.org a web site hosted by Stanford Uni-versity. As used herein, an "OpenFlow switch" refers to a network element implementing the OpenFlow protocol.)<br><br>Kempf at [0041] ("The standard EPC control plane entities-the MME, PCRF, and HSS-are likewise deployed in the cloud, along with the control plane parts of the S-GW and P-GW, namely, the S-GW-C and the P-GW-C. The data plane con-sists of standard OpenFlow switches with enhancements as needed for routing GTP packets, rather than IP routers and Ethernet switches. At a minimum, the data plane parts of the S-GW and P-GW, namely, the S-GW-Dand the P-GW-D, and the packet routing part of the E-NodeB in the E-UTRAN require OpenFlow enhancements for GTP routing. Addi-tional enhancements for GTP routing may be needed on other switches within the EPC architecture depending on how much fine grained control over the routing an operator requires.")<br><br>Kempf at [0078] ("FIG. 15 is a diagram of one embodiment of how the EPC in the cloud computing system enables a managed ser-vices company to manage multiple operator networks out of a single data center. The managed services cloud computing facility 1501 runs separate instances of the EPC control plane for every mobile operator with which the managed services company has a contract. Each EPC instance is in a VPC 1503A,B that isolates the mobile operator's traffic from other tenants in the cloud computing facility 1501 of the data cen-ter. The EPC control plane instance for a mobile operator is connected to the mobile operator's geographically distributed EPC OpenFlow data plane switching fabric 1507 A,B and the mobile operator's base stations through a virtual edge router 1509A,B. The virtual edge router 1509A,B routes traffic from the data center to and from the appropriate mobile operator EPC data plane switching fabric 1507 A,B. In some cases, the |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 546 of 1100

| No. | '111 Patent Claim 31 | Chandrasekaran |
|---|---|---|
| | | mobile operators may even share base stations and EPC switching fabrics, though the example embodiment in FIG. 15 shows a case where the two mobile operators have separate switching fabrics.")<br><br>Kempf at [0087] ("In one embodiment, slow path support for GTP is implemented with an OpenFlow gateway switch. An Open-Flow mobile gateway switch also contains support on the software control plane for slow path packet processing. This path is taken by G-PDU (message type 255) packets with nonzero header fields or extension headers, and user data plane packets requiring encapsulation with such fields or addition of extension headers, and by G TP-U control packets. For this purpose, the switch supports three local ports in the software control plane: LOCAL_GTP _CONTROL-the switch fast path forwards GTP encapsulated packets directed to the gateway IP address that contain GTP-U control mes-sages and the local switch software control plane initiates local control plane actions depending on the GTP-U control message; LOCAL_GTP _U_DECAP-the switch fast path forwards G-PDU packets to this port that have nonzero header fields or extension headers (i.e. E!=0, S!=0, or PN!=0). These packets require specialized handling. The local switch software slow path processes the packets and performs the specialized handling; and LOCAL_GTP _U_ENCAP-the switch fast path forwards user data plane packets to this port that require encapsulation in a GTP tunnel with nonzero header fields or extension headers (i.e. E!=0, S!=0, or PN!=0). These packets require specialized handling. The local switch software slow path encapsulates the packets and performs the specialized handling. In addition to forwarding the packet, the switch fast path makes the OpenFlow metadata field avail-able to the slow path software.")<br><br>Kempf at [0093] ("The virtual port simply removes the GTP tunnel header and forwards the enclosed user data plane packet out the bound physical port.")<br><br>Kempf at [0101] ("In one embodiment, the system implements han-dling of user data plane packets requiring GTP-U encapsula-tion with extension headers, sequence numbers, and N-PDU numbers. User data plane packets that require extension head-ers, sequence numbers, or N-PDU numbers during GTP encapsulation require special handling by the software slow path. For these packets, the OpenFlow controller programs a rule matching the 4 tuple: IP source address; IP destination address; UDP/TCP/SCTP source port; and UDP/TCP/SCTP destination port. The instructions for matching packets are: |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 547 of 1100

| No. | '111 Patent Claim 31 | Chandrasekaran |
|---|---|---|
| | | Write-Metadata ( GTP-TEID, 0x FFFFFFFF)<br>Apply-Actions (Set-Output-Port LOCAL_GTP _U_ENCAP )")<br><br>Kempf at [0145] ("In other embodiments, other control protocols can be utilized in place of OpenFlow as described herein. The use of OpenFlow is presented by way of example and not limita-tion. Other control protocols can also be utilized to manage the communication between the control plane and data plane and configuration of the data plane of the split EPC architec-ture. An example of such a protocol is FORCES, an IETF standard protocol for splitting the control plane and forward-ing plane in networks. The FORCES protocol specification is described in RFC 5810. RFC 5812 describes the architecture of a FORCES forwarding element, the equivalent of an Open-Flow switch. The FORCES protocol itself does not directly support programming routes into the forwarding element, it is, instead, a framework for handling the interaction between the FORCES controller and a FORCES forwarding element. The forwarding element architecture describes how to design the protocol that actually allows a FORCES controller to program a FORCES forwarding element. One skilled in the art would understand that a FORCES based system could include features described herein above in relation to the OpenFlow embodiment, such as the GTP OpenFlow exten-sion, to allow the controller to program the switches for GTP TEID routing.")<br><br>As another example, Chua discloses the network device's communication with the SDN controller over the control plane as controlling and programming the network devices to direct network traffic along one or more paths.<br><br>Chua at 1:64-2:5 ("In another example, a controller device for a software defined network includes one or more interfaces for commu-nicating with network devices in the software defined net-work, and one or more processors configured to determine connections between the network devices, determine one or more paths for network traffic between the network devices based on the determination of the connections, and program the network devices to direct network traffic along the one or more paths.")<br><br>Chua at 2:21-29 ("In another example, a controller device for a software defined network (SDN) includes one or more network inter-faces configured to communicate with network |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 548 of 1100

| No. | '111 Patent Claim 31 | Chandrasekaran |
|---|---|---|
| | | devices of the SDN, and one or more processors configured to program a first network device of the SDN to send packets of a packet flow to a service device, and program one or more network devices of the SDN to perform a programmed action on packets of the packet flow based on data received from the service device for the packet flow.")<br><br>Chua at 2:49-61 ("In another example, a controller device for a software defined network (SDN) includes one or more network interfaces configured to communicate with network devices of the SDN, and one or more processors configured to program a set of network devices of the SDN to form a path through the SDN and to send data representative of packets sent along the path to the controller device, send, via one of the network interfaces, packets of a packet flow corresponding to the path to one of the set of network devices, determine whether the set of network devices is properly forwarding the packets of the packet flow along the path based on data received from the set of network devices, and present a report representative of the determination.")<br><br>Chua at 23:62-24:4 ("OpenFlow is an example of an SDN protocol. That is, in some examples, SDN controller 270 may conform to the OpenFlow protocol. However, it should be understood that other protocols may be used in conjunction with a software defined network. In general, any protocol that gives access to the forwarding plane or data plane of a networking (e.g., a switch or router) to a remote device over a network may be used in accordance with the techniques of this disclo-sure, other example protocols include XMPP, RESTful APis, Cisco OnePK, IETF I2RS (Interface to Routing Systems).") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 549 of 1100

---

**Chart for U.S. Patent 10,652,111 ("the '111 Patent")**
**U.S. Patent No. 9,264,400 to Lin et al. ("Lin '400")**

As shown in the chart below, all Asserted Claims of the '111 Patent are invalid under (1) AIA-35 U.S.C. § 102 (a) because Lin '400 meets each element of those claims, and/or (2) 35 U.S.C. § 103 because Lin '400 renders those claims obvious either alone, or in combination with the knowledge of a person having ordinary skill in the art, and in further combination with the references specifically identified below and in the following claim chart and/or one or more references identified in Defendant's Preliminary Invalidity Contentions. The following quotations and diagrams come from Lin '400 titled "Software Defined Networking Pipe For Network Traffic Inspection", which was filed on Dec. 2, 2013, and issued on February 16, 2016.

Motivations to combine the disclosures in Lin '400 with disclosures in other publications known in the art, as explained in this chart, include at least the similarity in subject matter between the references to the extent they concern methods relating to routing certain network traffic to entities for further analysis and inspection. Insofar as the references cite other patents or publications, or suggest additional changes, one of ordinary skill in the art would look beyond a single reference to other references in the field.

These invalidity contentions are based on Defendant's present understanding of the Asserted Claims, and Orckit's apparent construction of the claims in its November 3, 2022 Disclosure of Asserted Claims and Infringement Contentions Pursuant to P.R. 3-1, and Orckit's January 19, 2023 First Amended Disclosure of Asserted Claims and Infringement Contentions Pursuant to P.R. 3-1 (Orckit's "Infringement Disclosures"), which is deficient at least insofar as it fails to cite any documents or identify accused structures, acts, or materials in the Accused Products with particularity. Defendant does not agree with Orckit's application of the claims, or that the claims satisfy the requirements of 35 U.S.C. § 112. Defendant's contentions herein are not, and should in no way be seen as, admissions or adoptions as to any particular claim scope or construction, or as any admission that any particular element is met by any accused product in any particular way. Defendant objects to any attempt to imply claim construction from this chart. Defendant's prior art invalidity contentions are made in a variety of alternatives and do not represent Defendant's agreement or view as to the meaning, definiteness, written description support for, or enablement of any claim contained therein.



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 550 of 1100

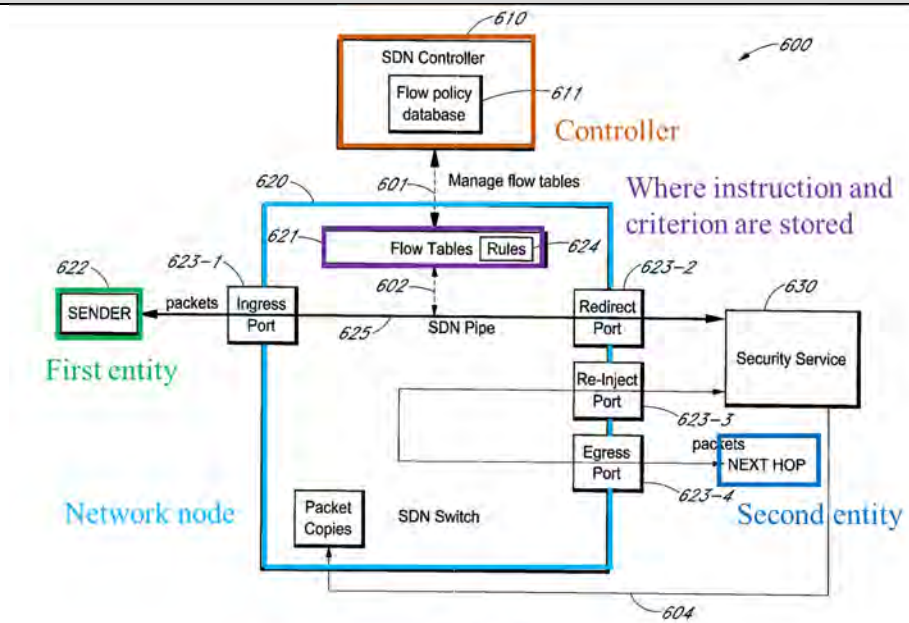
The following contentions are subject to revision and amendment pursuant to Federal Rule of Civil Procedure 26(e), the Local Rules, and the Orders of record in this matter subject to further investigation and discovery regarding the prior art and the Court's construction of the claims at issue.

| No. | '111 Patent Claim 1 | Lin '400 |
|---|---|---|
| 1[preamble] | A method for use with a packet network including a network node for transporting packets between first and second entities under control of a controller that is external to the network node, the method comprising: | Lin '400 discloses a method for use with a packet network including a network node for transporting packets between first and second entities under control of a controller that is external to the network node.<br><br>For example, Lin '400 discloses that it relates to a software defined networking (SDN) computer network under the control of an SDN controller over the SDN computer network, from a sender component through an ingress port, out an egress port, and to the "next hope" destination. Lin '400 further discloses that the SDN controller is external to the SDN switch. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met.<br><br>Lin '400 1:7-9 ("The present invention relates generally to computer security, and more particularly but not exclusively to software defined networking.").<br><br>Lin '400 1:58-2:4 ("In one embodiment, a software defined networking (SDN) computer network includes an SDN controller and an SDN switch. The SDN controller inserts flow rules in a flow table of the SDN switch to create an SDN pipe between a sender component and a security component. A broadcast function of the SDN switch to the ports that form the SDN pipe may be disabled. The SDN pipe allows outgoing packets sent by the sender component to be received by the security component. The security component inspects the outgoing packets for compliance with security policies and allows the outgoing packets to be forwarded to their destination when the outgoing packets pass inspection. The SDN controller may also insert a flow rule in the flow table of the SDN switch to bypass inspection of specified packets.").<br><br>Lin '400 2:47-65 ("FIG. 2 shows a schematic diagram of a computer system 100 that may be employed with embodiments of the present invention. The computer system 100 may be employed as a control plane and/or a data plane, for example. As another example, the computer system 100 may be employed to host a virtualization environment that supports a plurality of virtual machines. The computer system 100 may have fewer or more components |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 551 of 1100

| No. | '111 Patent Claim 1 | Lin '400 |
|---|---|---|
| | | to meet the needs of a particular application. The computer system 100 may include one or more processors 101. The computer system 100 may have one or more buses 103 coupling its various components. The computer system 100 may include one or more user input devices 102 (e.g., keyboard, mouse), one or more data storage devices 106 (e.g., hard drive, optical disk, Universal Serial Bus memory), a display monitor 104 (e.g., liquid crystal display, flat panel monitor), a computer network interface 105 (e.g., network adapter, modem), and a main memory 108 (e.g., random access memory). The computer network interface 105 may be coupled to a computer network 109.").<br><br>Lin '400 3:25-33 ("Another way of intercepting network traffic is to mirror the packets to be inspected on a switch that provides vendor specific mirroring application programming interface (API) as shown in FIG. 4. A user may make an API call such that particular packets that enter the ingress port of the switch are redirected or mirrored to the security service by way of a connection tunnel or a mirror port. The security service may forward the redirected or mirrored packets back to an egress port of the switch after inspection.").<br><br>Lin '400 3:40-64 ("Referring now to FIG. 6, there is shown a schematic diagram of an SDN computer network 600 in accordance with an embodiment of the present invention. In one embodiment, the SDN computer network 600 is compliant with the OpenFlow™ protocol. Accordingly, in one embodiment, the SDN controller 610 comprises an OpenFlow™ controller and the SDN switch 620 comprises an OpenFlow™ switch. The SDN controller 610 and the SDN switch 620 comprise the control plane and data plane, respectively, of the SDN computer network 600. The SDN computer network 600 may have a plurality of SDN switches 620 but only one is shown for clarity of illustration. The SDN controller 610 and the SDN switch 620 are logically separate components.<br>In one embodiment, the SDN computer network 600 is a virtual computer network that allows for transmission of packets from one virtual machine to another. Accordingly, the SDN controller 610 may comprise a virtual OpenFlow™ controller and the SDN switch 620 may comprise a virtual OpenFlow™ switch. The SDN computer network 600 may be implemented in a computer system comprising one or more computers that host a virtualization environment. For example, the SDN computer network 600 may be implemented in the Amazon Web Services™ virtualization environment. The sender component 622 may be a virtual machine in that embodiment.").<br><br>Orckit Exhibit 2018<br>Cisco Systems v. Orckit Corp.<br>IPR2023-00714, Page 552 of 1100 |

| No. | '111 Patent Claim 1 | Lin '400 |
|---|---|---|
| | |  <br>*FIG. 6*<br>Fig. 6 (annotation added)<br><br>Lin '400 4:33-67 ("The SDN switch 620 may comprise a plurality of ports 623 (i.e., 623-1, 623-2, 623-3, 623-4, etc.). The SDN switch 620 may forward packets from one port 623 to another port 623 in accordance with flow rules in the flow tables 621. In the example of FIG. 6, the port 6231-1 is coupled to a sender component 622. The port 623-1 is referred to as an "ingress port" in that it is a port for receiving outgoing packets sent by the sender component 622. Similarly, the port 623-4 is referred to as an "egress port" in that it is a port for transmitting outgoing packets sent by the sender component 622. It is to be noted that any port 623 may be employed as an "ingress port," "egress port," "redirect port," or "re-inject port." The aforementioned labels are used herein merely to illustrate processing of packets relative to the sender component 622. Packets going in the opposite direction, i.e., incoming packets that are going to the sender component 622, may be received in the egress port 623-4, forwarded to the re-inject port 623-3, received in the redirect port 623-2, and forwarded to the ingress port 623-1 toward the sender component 622. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 553 of 1100

| No. | '111 Patent Claim 1 | Lin '400 |
|---|---|---|
| | | The SDN switch 620 may comprise one or more flow tables 621. The flow tables 621 may comprise one or more flow rules (labeled as 624) that indicate how to manipulate or process packets that are passing through the SDN switch 620. As a particular example, a flow rule may indicate that a packet received in the ingress port 623-1 is to be forwarded to the redirect port 623-2. Another flow rule may indicate that a packet received in the redirect port 623-2 is to be forwarded to the ingress port 623-1. The just mentioned pair of flow rules are redirect flow rules that create an SDN pipe between the sender component 622 and the security service 630, allowing the security service 630 to inspect packets sent by or going to the sender component 622. Table 1 shows an example flow table with flow rules that create an SDN pipe between the security service 630 and the sender component 622.").<br><br>Lin '400 6:13-23 ("Once outgoing packets from the sender component 622 are inspected by the security service 630 and re-injected by the security service 630 back into the SDN switch 620 through the re-inject port 623-3 and then forwarded out to the egress port 623-4, the L2 switching logic of the SDN computer network 600 (which is controlled by the SDN controller 610) remembers that packets destined for the sender component 622 and entering the SDN switch 620 by way of the egress port 623-4 are to be forwarded to the re-inject port 623-3. This allows the security service 630 to also receive incoming packets going to the sender component 622 for inspection.").<br><br>Lin '400 6:57-63 ("Once back in the SDN switch 620 by way of the re-inject port 623-3, the flow rules that govern packets received in the ingress port 623-1 and the redirect port 623-2 no longer apply. Accordingly, the re-injected packets are forwarded to the egress port 623-4 (or some other port) toward the next hop in accordance with the L2 switching logic of the SDN computer network 600.").<br><br>Lin '400 7:10-23 ("Re-injecting packets that pass inspection consume bandwidth, as the packets will have to be transmitted by the security service 630 to the re-inject port 623-3. For optimization, the SDN switch 620 may be configured to copy packets that are redirected to the security service 630 for inspection. This way, the security service 630 simply has to inform the SDN switch 620 an action to take on packets based on the result of the inspection (see arrow 604). For example, the security service 630 may send an index identifying the packets and an action on how to manipulate the packets. The action may instruct the SDN |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 554 of 1100

| No. | '111 Patent Claim 1 | Lin '400 |
|---|---|---|
| | | switch 620 to drop the copied packets, forward the copied packets to their destinations, quarantine the copied packets, etc."). |

Lin '400 7:39-67 ("In the example of Table 2, the first two rows are bypass rules for bypassing packets coming from or going to a transport control protocol (TCP) port 80. More specifically, hypertext transfer protocol (HTTP) packets, i.e., port 80 packets, that are received in the ingress port with the Ingress_port_ID (i.e., ingress port 623-1) are forwarded directly to the egress port (i.e., egress port 623-4), instead of being redirected to the redirect port 623-2 for inspection by the security service 630. Similarly, HTTP packets received in the egress port with the Egress_port_ID (i.e., egress port 623-4) are forwarded directly to the ingress port 623-1 without being redirected to the security service 630.

In the example of Table 2, the bottom two rows are redirect flow rules for forming the SDN pipe between the sender component 622 and the security service 630. Because the bypass flow rules are inserted in the flow tables 621 with higher priority than the redirect flow rules, the bypass flow rules are followed by the SDN switch 620 before the redirect flow rules. Accordingly, HTTP packets are not redirected for inspection by the security service 630. Other packets, i.e., non-HTTP packets, are redirected to the security service 630 per the redirect flow rules. Bypass flow rules and redirect flow rules may be set at different priority levels to meet particular packet inspection needs.

The bypass and redirect flow rules also allow for inspection of particular packets, while allowing all other packets to bypass inspection. This is illustrated in the example flow table of Table 3.").

TABLE 3

| IN_PORT | . . . | IP src | TCP src port | | | TCP dst port | | . . . | Action | Count |
|---|---|---|---|---|---|---|---|---|---|---|
| Ingress_port_ID | | | * | * | * | 80 | * | | Redirect port | 10 |
| Redirect_port_ID | | | * | * | 80 | * | * | | Ingress port | 10 |
| Ingress_port_ID | | | * | * | * | * | * | | Egress port | 130 |
| Egress_port_ID | | | * | * | * | * | * | | Ingress port | 130 |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 555 of 1100

| No. | '111 Patent Claim 1 | Lin '400 |
|---|---|---|
| | | Lin '400 8:10-18 ("In the example of Table 3, the top two rows are redirect flow rules for redirecting HTTP packets to the security service 630 for inspection, while the bottom two rows are bypass flow rules for all packets. Because the redirect flow rules are at higher priority than the bypass flow rules, HTTP packets are sent through the SDN pipe formed in the SDN switch 620 between the sender component 622 and the security service 630. All other packets bypass the SDN pipe, and are accordingly not inspected by the security service 630."). |
| | | Lin '400 Claim 1 ("A software defined networking (SDN) computer network comprising:<br>an SDN switch comprising a plurality of ports that receives network traffic of an SDN computer network, the SDN switch having a first port coupled to a sender component and a second port coupled to a security component, the SDN switch comprising a flow table that comprises a first flow rule to forward a packet received in the first port to the second port and a second flow rule to forward a packet received in the second port to the first port, the SDN switch receiving outgoing packets from the first port and forwarding the outgoing packets to the second port in accordance with the first flow rule, the outgoing packets being sent by the sender component to a destination component; and<br>an SDN controller that controls 7orwardding behavior of the SDN switch and inserts the first and second flow rules into the flow table of the SDN switch,<br>wherein the security component receives the outgoing packets from the second port of the SDN switch, inspects the outgoing packets, and allows the outgoing packets to be forwarded to their destination when the outgoing packets pass inspection,<br>wherein the security component allows the outgoing packets to be forwarded to their destination by instructing the SDN switch to release copies of the outgoing packets."). |
| | | Lin '400 4:8-31 ("The SDN controller 610 provides a logically centralized framework for controlling the behavior of the SDN computer network 600. This is in marked contrast to traditional computer networks where the behavior of the computer network is controlled by low-level device configurations of switches and other network devices. The SDN controller 610 may include a flow policy database 611. The flow policy database 611 may comprise flow policies that are enforced by the controller 610 on network traffic transmitted over the SDN computer network 600. The flow policies may specify security policies that govern transmission of packets over the SDN computer network 600. The flow policies may be enforced in terms of flow rules (labeled as 624) that are stored in the flow tables 621 of the |

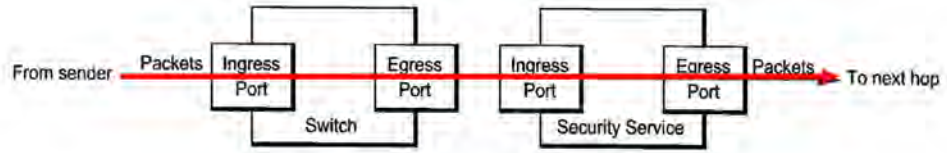


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 556 of 1100

| No. | '111 Patent Claim 1 | Lin '400 |
|---|---|---|
| | | SDN switch 620. As a particular example, a flow policy in the flow policy database 611 may indicate inspection of particular packets (e.g., those that meet one or more conditions) by a security service 630. That flow policy may be implemented as a flow rule that forwards the particular packets received in an ingress port 623-1 to the redirect port 623-2 for inspection, for example.").<br><br>Lin '400 6:1-12 ("The SDN controller 610 may insert flow rules in the flow tables 621 (see arrow 601) to create an SDN pipe (labeled as 625) between the sender component 622 and the security service 630. The SDN pipe allows outgoing packets sent by the sender component 622 or incoming packets going to the sender component 622 to be redirected to the security service 630 for inspection before the packets are sent out of the SDN switch 620. In one embodiment, the SDN pipe is created by creating a first flow rule that forwards packets received in the ingress port 623-1 to the redirect port 623-2, and a second flow rule that forwards packets received in the redirect port 623-2 to the ingress port 623-1.").<br><br>Lin '400 3:11-12 ("Network security vendors provide network security services, such as firewall or deep packet inspection (DPI). Generally speaking, to provide network security services, packets of network traffic are intercepted for inspection. One way of intercepting network traffic is to place the security service in the middle of the packet forwarding path. This is illustrated in FIG. 3, where packets from a sender component (e.g., a sender computer) are received in an ingress port of a switch, forwarded to an egress port of the switch, and forwarded to the ingress port of a security component, such as a security service. The security service may inspect the packets, and forward the packets to an egress port of the switch toward the next hop, which may be another switch or a destination component (e.g., destination computer), for example."). |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 557 of 1100

| No. | '111 Patent Claim 1 | Lin '400 |
|---|---|---|
| | |  <br>FIG. 3<br><br>Fig. 3 (annotation added) |
| 1[a] | sending, by the controller to the network node over the packet network, an instruction and a packet-applicable criterion; | Lin '400 discloses sending, by the controller to the network node over the packet network, an instruction and a packet-applicable criterion.<br><br>For example, Lin '400 identifies a command to determine whether or not a packet requires inspection. Lin '400 further discloses that its SDN controllers inserts flow rules in a flow table of the SDN switch (which corresponds to the claimed network node) to create an AND pipe between a sender component and a security components, where these flow tables are stored in switches.<br><br>Lin '400 1:58-2:4 ("In one embodiment, a software defined networking (SDN) computer network includes an SDN controller and an SDN switch. The SDN controller inserts flow rules in a flow table of the SDN switch to create an SDN pipe between a sender component and a security component. A broadcast function of the SDN switch to the ports that form the SDN pipe may be disabled. The SDN pipe allows outgoing packets sent by the sender component to be received by the security component. The security component inspects the outgoing packets for compliance with security policies and allows the outgoing packets to be forwarded to their destination when the outgoing packets pass inspection. The SDN controller may also insert a flow rule in the flow table of the SDN switch to bypass inspection of specified packets."). |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 558 of 1100

| No. | '111 Patent Claim 1 | Lin '400 |
|---|---|---|
| | | Lin '400 7:39-8:18 ("In the example of Table 2, the first two rows are bypass rules for bypassing packets coming from or going to a transport control protocol (TCP) port 80. More specifically, hypertext transfer protocol (HTTP) packets, i.e., port 80 packets, that are received in the ingress port with the Ingress_port_ID (i.e., ingress port 623-1) are forwarded directly to the egress port (i.e., egress port 623-4), instead of being redirected to the redirect port 623-2 for inspection by the security service 630. Similarly, HTTP packets received in the egress port with the Egress_port_ID (i.e., egress port 623-4) are forwarded directly to the ingress port 623-1 without being redirected to the security service 630.<br>In the example of Table 2, the bottom two rows are redirect flow rules for forming the SDN pipe between the sender component 622 and the security service 630. Because the bypass flow rules are inserted in the flow tables 621 with higher priority than the redirect flow rules, the bypass flow rules are followed by the SDN switch 620 before the redirect flow rules. Accordingly, HTTP packets are not redirected for inspection by the security service 630. Other packets, i.e., non-HTTP packets, are redirected to the security service 630 per the redirect flow rules. Bypass flow rules and redirect flow rules may be set at different priority levels to meet particular packet inspection needs.<br>The bypass and redirect flow rules also allow for inspection of particular packets, while allowing all other packets to bypass inspection. This is illustrated in the example flow table of Table 3.").<br><br>Lin '400 4:23-31 ("The flow policies may be enforced in terms of flow rules (labeled as 624) that are stored in the flow tables 621 of the SDN switch 620. As a particular example, a flow policy in the flow policy database 611 may indicate inspection of particular packets (e.g., those that meet one or more conditions) by a security service 630. That flow policy may be implemented as a flow rule that forwards the particular packets received in an ingress port 623-1 to the redirect port 623-2 for inspection, for example.").<br><br>Lin '400 5:22-24 ("When the conditions are met, i.e., the particular packet is identified, the action indicated in the corresponding "Action" column is performed on the packet.").<br><br>TABLE 1<br><br>IN_PORT / MAC src / MAC dst / IP src / IP dst / . . . / Action / Count |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 559 of 1100

| No. | '111 Patent Claim 1 | Lin '400 |
|---|---|---|
| | | Ingress_port_ID * * * * * Redirect port 10<br>Redirect_port_ID * * * * * Ingress port 10<br><br>Lin '400 5:26-36 ("In the example of Table 1, the first and second rows are redirect flow rules for forming an SDN pipe between the sender component 622 and the security service 630. More specifically, the first row of Table 1 is a flow rule instructing the SDN switch 620 to forward packets received in a port having the Ingress_port_ID (e.g., ingress port 623-1) to the redirect port (e.g., redirect port 623-2). Similarly, the second row of Table 1 is a flow rule instructing the SDN switch 620 to forward packets received in a port having a "Redirect_port_ID" to the ingress port.").<br><br>Lin '400 6:1-12 ("The SDN controller 610 may insert flow rules in the flow tables 621 (see arrow 601) to create an SDN pipe (labeled as 625) between the sender component 622 and the security service 630. The SDN pipe allows outgoing packets sent by the sender component 622 or incoming packets going to the sender component 622 to be redirected to the security service 630 for inspection before the packets are sent out of the SDN switch 620. In one embodiment, the SDN pipe is created by creating a first flow rule that forwards packets received in the ingress port 623-1 to the redirect port 623-2, and a second flow rule that forwards packets received in the redirect port 623-2 to the ingress port 623-1.").<br><br>Lin '400 6:40-54 ("After the redirect flow rules for creating the SDN pipe are inserted in the flow tables 621, any packet received by the SDN switch 620 in the ingress port 623-1 will be identified as to be forwarded to the redirect port 623-2, and any packet received by the SDN switch 620 in the redirect port 623-2 will be identified as to be forwarded to the ingress port 623-1 (see arrow 602). This allows the security service 630 to receive from the redirect port 623-2 all outgoing packets sent by the sender component 622 to the ingress port 623-1. The security service 630 may inspect the outgoing packets for compliance with security policies. The security service 630 may drop, or perform other security response, to packets that do not pass inspection (e.g., packets that do not meet firewall policies, packets containing prohibited payload, packets with malicious content, etc.).") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 560 of 1100

| No. | '111 Patent Claim 1 | Lin '400 |
|---|---|---|
| 1[b] | receiving, by the network node from the controller, the instruction and the criterion; | Lin '400 discloses receiving, by the network node from the controller, the instruction and the criterion.<br><br>For example, Lin '400 discloses flow rules in a flow table of the SDN switch to create an SDN pipe between a sender component and a security component that are sent to the SDN switches in order to provide instruction to the SDN switches on what packets should be sent to the security component for analysis. Lin '400 further discloses that its SDN controller inserts flow rules in a flow table of the SDN switch where these flow rules may indicate inspection of particular packets (*e.g.*, those that meet one or more conditions) by a security service.<br><br>Lin '400 1:57-2:4 ("In one embodiment, a software defined networking (SDN) computer network includes an SDN controller and an SDN switch. The SDN controller inserts flow rules in a flow table of the SDN switch to create an SDN pipe between a sender component and a security component. A broadcast function of the SDN switch to the ports that form the SDN pipe may be disabled. The SDN pipe allows outgoing packets sent by the sender component to be received by the security component. The security component inspects the outgoing packets for compliance with security policies and allows the outgoing packets to be forwarded to their destination when the outgoing packets pass inspection. The SDN controller may also insert a flow rule in the flow table of the SDN switch to bypass inspection of specified packets.").<br><br>Lin '400 4:23-31 ("The flow policies may be enforced in terms of flow rules (labeled as 624) that are stored in the flow tables 621 of the SDN switch 620. As a particular example, a flow policy in the flow policy database 611 may indicate inspection of particular packets (e.g., those that meet one or more conditions) by a security service 630. That flow policy may be implemented as a flow rule that forwards the particular packets received in an ingress port 623-1 to the redirect port 623-2 for inspection, for example.").<br><br>Lin '400 5:8-12 ("A flow table may include columns that indicate one or more conditions, a column that indicates an action to take when the conditions are met, and a column for statistics. A row on the flow table may comprise a flow rule."). |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 561 of 1100

| No. | '111 Patent Claim 1 | Lin '400 |
|---|---|---|
| | | Lin '400 6:1-4 ("The SDN controller 610 may insert flow rules in the flow tables 621 (see arrow 601) to create an SDN pipe (labeled as 625) between the sender component 622 and the security service 630. "). |

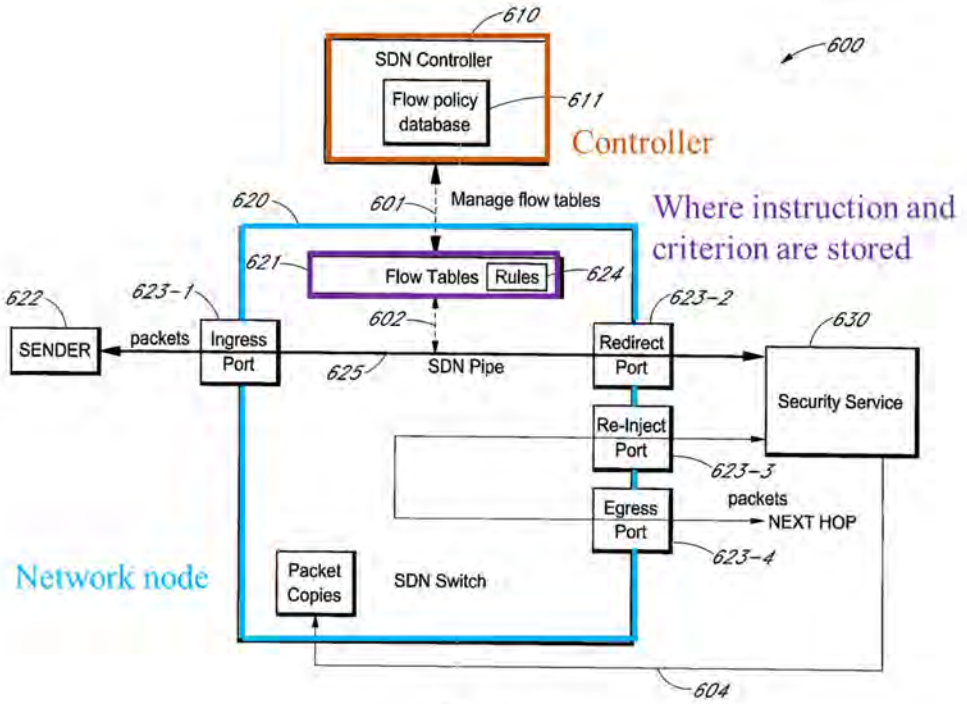



FIG. 6

Fig. 6 (annotation added)

Lin '400 6:40-41 ("…redirect flow rules for creating the SDN pipe are inserted in the flow tables 621…").

Lin '400 9:10-12 ("…the SDN controller inserts one or more bypass flow rules in the flow table of an SDN switch that is controlled by the SDN controller (step 701).").

*See supra at* 1[a].



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 562 of 1100

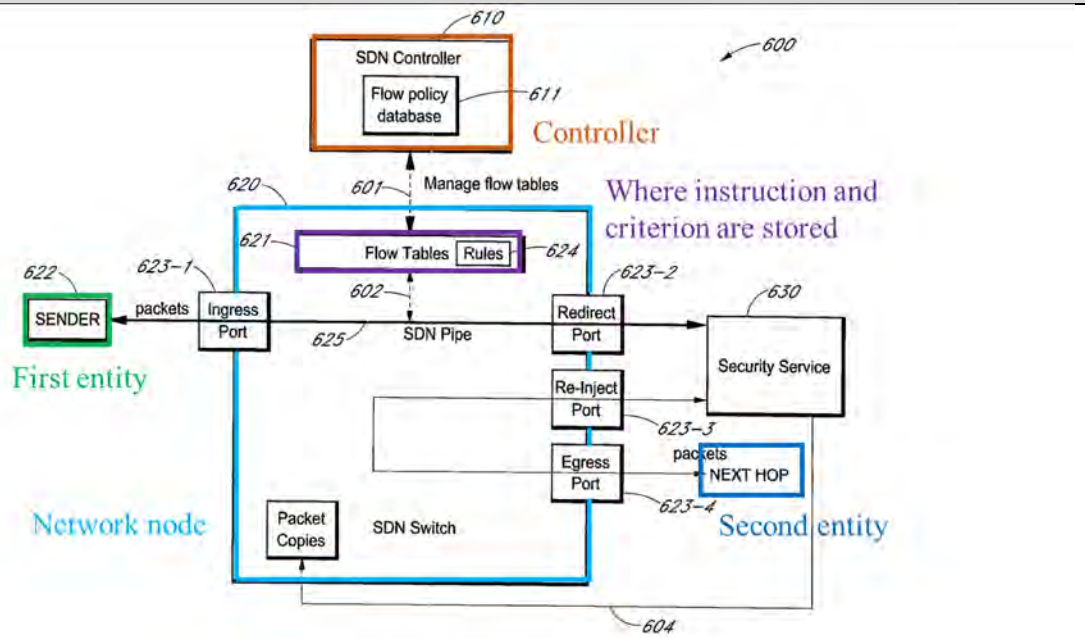
| No. | '111 Patent Claim 1 | Lin '400 |
|---|---|---|
| | | |
| 1[c] | receiving, by the network node from the first entity over the packet network, a packet addressed to the second entity; | Lin '400 discloses receiving, by the network node from the first entity over the packet network, a packet addressed to the second entity.<br><br>For example, Lin '400 explains that the SDN switch transports packets from a sender component through an ingress port, out an egress port, and follows the directed address to the "next hop" or destination.<br><br>Lin '400 1:58-2:4 ("In one embodiment, a software defined networking (SDN) computer network includes an SDN controller and an SDN switch. The SDN controller inserts flow rules in a flow table of the SDN switch to create an SDN pipe between a sender component and a security component. A broadcast function of the SDN switch to the ports that form the SDN pipe may be disabled. The SDN pipe allows outgoing packets sent by the sender component to be received by the security component. The security component inspects the outgoing packets for compliance with security policies and allows the outgoing packets to be forwarded to their destination when the outgoing packets pass inspection. The SDN controller may also insert a flow rule in the flow table of the SDN switch to bypass inspection of specified packets.").<br><br>Lin '400 3:11-12 ("Network security vendors provide network security services, such as firewall or deep packet inspection (DPI). Generally speaking, to provide network security services, packets of network traffic are intercepted for inspection. One way of intercepting network traffic is to place the security service in the middle of the packet forwarding path. This is illustrated in FIG. 3, where packets from a sender component (e.g., a sender computer) are received in an ingress port of a switch, forwarded to an egress port of the switch, and forwarded to the ingress port of a security component, such as a security service. The security service may inspect the packets, and forward the packets to an egress port of the switch toward the next hop, which may be another switch or a destination component (e.g., destination computer), for example.").<br><br>Lin '400 4:33-67 ("The SDN switch 620 may comprise a plurality of ports 623 (i.e., 623-1, 623-2, 623-3, 623-4, etc.). The SDN switch 620 may forward packets from one port 623 to another port 623 in accordance with flow rules in the flow tables 621. In the example of FIG. 6, the port 6231-1 is coupled to a sender component 622. The port 623-1 is referred to as an |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 563 of 1100

| No. | '111 Patent Claim 1 | Lin '400 |
|---|---|---|
| | | "ingress port" in that it is a port for receiving outgoing packets sent by the sender component 622. Similarly, the port 623-4 is referred to as an "egress port" in that it is a port for transmitting outgoing packets sent by the sender component 622. It is to be noted that any port 623 may be employed as an "ingress port," "egress port," "redirect port," or "re-inject port." The aforementioned labels are used herein merely to illustrate processing of packets relative to the sender component 622. Packets going in the opposite direction, i.e., incoming packets that are going to the sender component 622, may be received in the egress port 623-4, forwarded to the re-inject port 623-3, received in the redirect port 623-2, and forwarded to the ingress port 623-1 toward the sender component 622.<br>The SDN switch 620 may comprise one or more flow tables 621. The flow tables 621 may comprise one or more flow rules (labeled as 624) that indicate how to manipulate or process packets that are passing through the SDN switch 620. As a particular example, a flow rule may indicate that a packet received in the ingress port 623-1 is to be forwarded to the redirect port 623-2. Another flow rule may indicate that a packet received in the redirect port 623-2 is to be forwarded to the ingress port 623-1. The just mentioned pair of flow rules are redirect flow rules that create an SDN pipe between the sender component 622 and the security service 630, allowing the security service 630 to inspect packets sent by or going to the sender component 622. Table 1 shows an example flow table with flow rules that create an SDN pipe between the security service 630 and the sender component 622."). |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 564 of 1100

| No. | '111 Patent Claim 1 | Lin '400 |
|---|---|---|
| | |  <br>*FIG. 6*<br>Fig. 6 (annotation added)<br><br>Lin '400 Claim 1 (". A software defined networking (SDN) computer network comprising:<br>an SDN switch comprising a plurality of ports that receives network traffic of an SDN computer network, the SDN switch having a first port coupled to a sender component and a second port coupled to a security component, the SDN switch comprising a flow table that comprises a first flow rule to forward a packet received in the first port to the second port and a second flow rule to forward a packet received in the second port to the first port, the SDN switch receiving outgoing packets from the first port and forwarding the outgoing packets to the second port in accordance with the first flow rule, the outgoing packets being sent by the sender component to a destination component; and<br>an SDN controller that controls forwarding behavior of the SDN switch and inserts the first and second flow rules into the flow table of the SDN switch, |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 565 of 1100

| No. | '111 Patent Claim 1 | Lin '400 |
|---|---|---|
| | | wherein the security component receives the outgoing packets from the second port of the SDN switch, inspects the outgoing packets, and allows the outgoing packets to be forwarded to their destination when the outgoing packets pass inspection,<br>wherein the security component allows the outgoing packets to be forwarded to their destination by instructing the SDN switch to release copies of the outgoing packets."). |
| 1[d] | checking, by the network node, if the packet satisfies the criterion; | Lin '400 discloses checking, by the network node, if the packet satisfies the criterion.<br><br>For example, Lin '400 discloses flow rules that may indicate inspection of particular packets (*e.g.*, those that meet one or more conditions) by a security service, where under the inspection, if the conditions are met, the action indicated in the corresponding "Action" column in the table is performed on the packet. Lin '400 further discloses that the SDN switch implements bypass flow rules that check whether the packet meets certain criterion, such as identification of HTTP packets, that indicate that the packet should be routed to the destination node instead of the security device.<br><br>Lin '400 4:23-31 ("The flow policies may be enforced in terms of flow rules (labeled as 624) that are stored in the flow tables 621 of the SDN switch 620. As a particular example, a flow policy in the flow policy database 611 may indicate inspection of particular packets (e.g., those that meet one or more conditions) by a security service 630. That flow policy may be implemented as a flow rule that forwards the particular packets received in an ingress port 623-1 to the redirect port 623-2 for inspection, for example.").<br><br>Lin '400 5:8-12 ("A flow table may include columns that indicate one or more conditions, a column that indicates an action to take when the conditions are met, and a column for statistics. A row on the flow table may comprise a flow rule.").<br><br>TABLE 1<br><br>IN_PORT \| MAC src \| MAC dst \| IP src \| IP dst \| . . . \| Action \| Count<br><br>Ingress_port_ID \| * * * * * \| Redirect port \| 10 |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 566 of 1100

| No. | '111 Patent Claim 1 | Lin '400 |
|---|---|---|
| | | Redirect_port_ID * * * * * Ingress port 10<br><br>Lin '400 5:16-21 (" For example, "IN_PORT", "MAC src" (media access control (MAC) address of the source of the packet), "MAC dst" (MAC address of the destination of the packet), "IP src" (Internet Protocol (IP) address of the source of the packet), "IP dst" (IP address of the destination of the packet), etc. are conditions that identify a particular packet. When the conditions are met, i.e., the particular packet is identified, the action indicated in the corresponding "Action" column is performed on the packet. The asterisks in Table 1 indicate an irrelevant condition.").<br><br>Lin '400 5:22-24 ("When the conditions are met, i.e., the particular packet is identified, the action indicated in the corresponding "Action" column is performed on the packet.").<br><br>Lin '400 5:26-36 ("In the example of Table 1, the first and second rows are redirect flow rules for forming an SDN pipe between the sender component 622 and the security service 630. More specifically, the first row of Table 1 is a flow rule instructing the SDN switch 620 to forward packets received in a port having the Ingress_port_ID (e.g., ingress port 623-1) to the redirect port (e.g., redirect port 623-2). Similarly, the second row of Table 1 is a flow rule instructing the SDN switch 620 to forward packets received in a port having a "Redirect_port_ID" to the ingress port.").<br><br>Lin '400 6:1-12 ("The SDN controller 610 may insert flow rules in the flow tables 621 (see arrow 601) to create an SDN pipe (labeled as 625) between the sender component 622 and the security service 630. The SDN pipe allows outgoing packets sent by the sender component 622 or incoming packets going to the sender component 622 to be redirected to the security service 630 for inspection before the packets are sent out of the SDN switch 620. In one embodiment, the SDN pipe is created by creating a first flow rule that forwards packets received in the ingress port 623-1 to the redirect port 623-2, and a second flow rule that forwards packets received in the redirect port 623-2 to the ingress port 623-1.").<br><br>Lin '400 6:40-54 ("After the redirect flow rules for creating the SDN pipe are inserted in the flow tables 621, any packet received by the SDN switch 620 in the ingress port 623-1 will be identified as to be forwarded to the redirect port 623-2, and any packet received by the SDN switch 620 in the redirect port 623-2 will be identified as to be forwarded to the ingress |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 567 of 1100

| No. | '111 Patent Claim 1 | Lin '400 |
|---|---|---|
| | | port 623-1 (see arrow 602). This allows the security service 630 to receive from the redirect port 623-2 all outgoing packets sent by the sender component 622 to the ingress port 623-1. The security service 630 may inspect the outgoing packets for compliance with security policies. The security service 630 may drop, or perform other security response, to packets that do not pass inspection (e.g., packets that do not meet firewall policies, packets containing prohibited payload, packets with malicious content, etc.).").<br><br>Lin '400 7:24-8:18 ("In one embodiment, bypass flow rules are inserted in the flow tables 621 such that particular packets that do not need to be inspected are not redirected to the security service 630. This embodiment is explained with reference to example flow tables of Tables 2 and 3.<br><br>TABLE 2<br><br>IN_PORT . . . IP src / TCP src port / TCP dst port . . . Action Count<br><br>Ingress_port_ID * * * 80 * Egress port 120<br>Egress_port_ID * * 80 * * Ingress port 120<br>Ingress_port_ID * * * * * Redirect port 10<br>Redirect_port_ID * * * * * Ingress port 10<br><br>TABLE 3<br><br>IN_PORT . . . IP src / TCP src port / TCP dst port . . . Action Count<br><br>Ingress_port_ID * * * 80 * Redirect port 10<br>Redirect_port_ID * * 80 * * Ingress port 10<br>Ingress_port_ID * * * * * Egress port 130 |

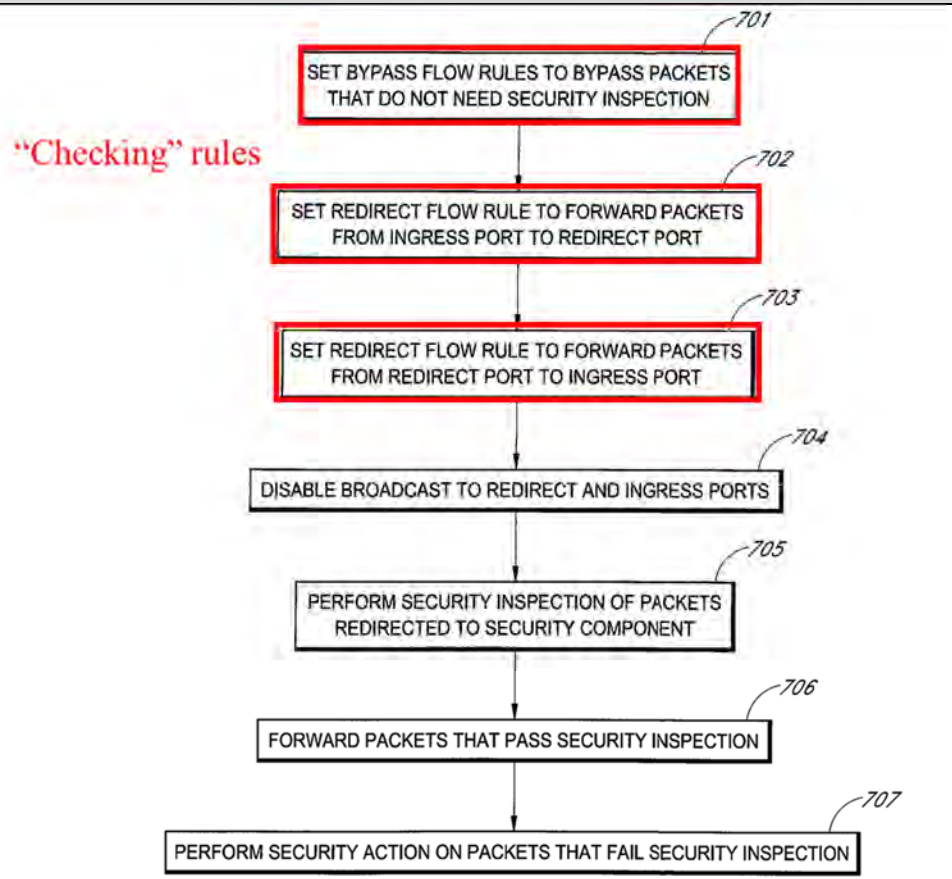


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 568 of 1100

| No. | '111 Patent Claim 1 | Lin '400 |
|---|---|---|
| | | Egress_port_ID * * * * * Ingress port 130<br><br>Lin '400 9:10-40 ("In the method of FIG. 9, the SDN controller inserts one or more bypass flow rules in the flow table of an SDN switch that is controlled by the SDN controller (step 701). The bypass flow rules may instruct the SDN switch to forward particular packets directly from one port to another port of the SDN switch without being redirected to a security component, such as a security service, for inspection. The bypass flow rules may be inserted at higher or lower priority than redirect flow rules for creating an SDN pipe that redirects packets to be inspected to the security service. The security service may comprise a physical or virtual machine that provides a security service by inspecting network traffic.<br>In the following two steps (steps 702 and 703), an SDN pipe is created in the SDN switch by inserting redirect flow rules in the flow table of the SDN switch. In one embodiment, the SDN pipe formed by the redirect flow rules allows for interception of packets sent by or transmitted to a virtual machine for inspection by the security service. The SDN controller inserts a first flow rule that instructs the SDN switch to forward packets received in an ingress port to a redirect port (step 702). The ingress port may be a port of the SDN switch that is connected to the virtual machine, and the redirect port may be a port of the SDN switch that is connected to the security service. The SDN controller also inserts a second flow rule that instructs the SDN switch to forward packets received in the redirect port to the egress port (step 703). The SDN controller may also disable the broadcast function of the SDN switch to the ingress port and the redirect port (step 704) to prevent the same broadcast packets to be received multiple times by the virtual machine and the security service"). |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 569 of 1100

| No. | '111 Patent Claim 1 | Lin '400 |
|---|---|---|
| | |  <br>*FIG. 9*<br>Fig. 9 (annotation added) |
| 1[e] | responsive to the packet not satisfying the criterion, sending, by the network node over the packet | Lin '400 discloses responsive to the packet not satisfying the criterion, sending, by the network node over the packet network, the packet to the second entity. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 570 of 1100

<table>
<tr><th>No.</th><th>’111 Patent Claim 1</th><th>Lin ’400</th></tr>
<tr><td></td><td>network, the packet to the second entity; and</td><td>For example, Lin ’400 discloses that during the checking phase, if a packet does not indicate that port 80 is the source or destination port, then the SDN switch sends the packet over the packet network to its destination node.<br><br>Lin ’400 7:24-8:18 (“In one embodiment, bypass flow rules are inserted in the flow tables 621 such that particular packets that do not need to be inspected are not redirected to the security service 630. This embodiment is explained with reference to example flow tables of Tables 2 and 3.<br><br>Lin ’400 8:10-18 (“In the example of Table 3, the top two rows are redirect flow rules for redirecting HTTP packets to the security service 630 for inspection, while the bottom two rows are bypass flow rules for all packets. Because the redirect flow rules are at higher priority than the bypass flow rules, HTTP packets are sent through the SDN pipe formed in the SDN switch 620 between the sender component 622 and the security service 630. All other packets bypass the SDN pipe, and are accordingly not inspected by the security service 630.”).<br><br>TABLE 3<br><br>
<table>
<tr><th>IN_PORT</th><th>. . .</th><th>IP src</th><th>TCP src port</th><th></th><th>TCP dst port</th><th></th><th>. . .</th><th>Action</th><th>Count</th></tr>
<tr><td>Ingress_port_ID</td><td></td><td></td><td>* * *</td><td></td><td>80</td><td>*</td><td></td><td>Redirect port</td><td>10</td></tr>
<tr><td>Redirect_port_ID</td><td></td><td></td><td>* *</td><td>80</td><td>*</td><td>*</td><td></td><td>Ingress port</td><td>10</td></tr>
<tr><td>Ingress_port_ID</td><td></td><td></td><td>* * *</td><td></td><td>*</td><td>*</td><td></td><td>Egress port</td><td>130</td></tr>
<tr><td>Egress_port_ID</td><td></td><td></td><td>* * *</td><td></td><td>*</td><td>*</td><td></td><td>Ingress port</td><td>130</td></tr>
</table>
<br>Lin ’400 9:10-40 (“In the method of FIG. 9, the SDN controller inserts one or more bypass flow rules in the flow table of an SDN switch that is controlled by the SDN controller (step 701). The bypass flow rules may instruct the SDN switch to forward particular packets directly from one port to another port of the SDN switch without being redirected to a security component, such as a security service, for inspection. The bypass flow rules may be inserted at higher or lower priority than redirect flow rules for creating an SDN pipe that redirects</td></tr>
</table>



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 571 of 1100

| No. | '111 Patent Claim 1 | Lin '400 |
|---|---|---|
| | | packets to be inspected to the security service. The security service may comprise a physical or virtual machine that provides a security service by inspecting network traffic.<br>In the following two steps (steps 702 and 703), an SDN pipe is created in the SDN switch by inserting redirect flow rules in the flow table of the SDN switch. In one embodiment, the SDN pipe formed by the redirect flow rules allows for interception of packets sent by or transmitted to a virtual machine for inspection by the security service. The SDN controller inserts a first flow rule that instructs the SDN switch to forward packets received in an ingress port to a redirect port (step 702). The ingress port may be a port of the SDN switch that is connected to the virtual machine, and the redirect port may be a port of the SDN switch that is connected to the security service. The SDN controller also inserts a second flow rule that instructs the SDN switch to forward packets received in the redirect port to the egress port (step 703). The SDN controller may also disable the broadcast function of the SDN switch to the ingress port and the redirect port (step 704) to prevent the same broadcast packets to be received multiple times by the virtual machine and the security service"). |
| 1[f] | responsive to the packet satisfying the criterion, sending the packet, by the network node over the packet network, to an entity that is included in the instruction and is other than the second entity. | Lin '400 discloses responsive to the packet satisfying the criterion, sending the packet, by the network node over the packet network, to an entity that is included in the instruction and is other than the second entity.<br><br>For example, Lin '400 teaches that the devices check for a specific packet-applicable criterion, where if a packet satisfies this criterion by indication that port 80 is the s destination port, then the SDN switch sends the packet over the packet network to the security service.<br><br>Lin '400 8:10-18 ("In the example of Table 3, the top two rows are redirect flow rules for redirecting HTTP packets to the security service 630 for inspection, while the bottom two rows are bypass flow rules for all packets. Because the redirect flow rules are at higher priority than the bypass flow rules, HTTP packets are sent through the SDN pipe formed in the SDN switch 620 between the sender component 622 and the security service 630. All other packets bypass the SDN pipe, and are accordingly not inspected by the security service 630.").<br><br>Lin '400 1:60-62 ("The SDN controller inserts flow rules in a flow table of the SDN switch to create an SDN pipe between a sender component and a security component."). |

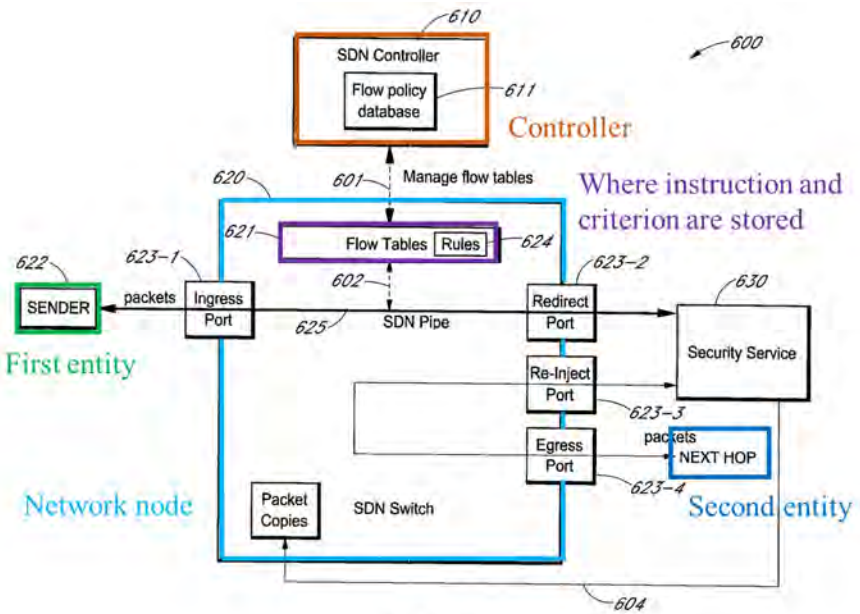


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 572 of 1100

| No. | '111 Patent Claim 1 | Lin '400 |
|---|---|---|
| | | Lin '400 4:8-31 ("The SDN controller 610 provides a logically centralized framework for controlling the behavior of the SDN computer network 600. This is in marked contrast to traditional computer networks where the behavior of the computer network is controlled by low-level device configurations of switches and other network devices. The SDN controller 610 may include a flow policy database 611. The flow policy database 611 may comprise flow policies that are enforced by the controller 610 on network traffic transmitted over the SDN computer network 600. The flow policies may specify security policies that govern transmission of packets over the SDN computer network 600. The flow policies may be enforced in terms of flow rules (labeled as 624) that are stored in the flow tables 621 of the SDN switch 620. As a particular example, a flow policy in the flow policy database 611 may indicate inspection of particular packets (e.g., those that meet one or more conditions) by a security service 630. That flow policy may be implemented as a flow rule that forwards the particular packets received in an ingress port 623-1 to the redirect port 623-2 for inspection, for example.").<br><br>Lin '400 4:53-67 ("The SDN switch 620 may comprise one or more flow tables 621. The flow tables 621 may comprise one or more flow rules (labeled as 624) that indicate how to manipulate or process packets that are passing through the SDN switch 620. As a particular example, a flow rule may indicate that a packet received in the ingress port 623-1 is to be forwarded to the redirect port 623-2. Another flow rule may indicate that a packet received in the redirect port 623-2 is to be forwarded to the ingress port 623-1. The just mentioned pair of flow rules are redirect flow rules that create an SDN pipe between the sender component 622 and the security service 630, allowing the security service 630 to inspect packets sent by or going to the sender component 622. Table 1 shows an example flow table with flow rules that create an SDN pipe between the security service 630 and the sender component 622.").<br><br>Lin '400 6:1-12 ("The SDN controller 610 may insert flow rules in the flow tables 621 (see arrow 601) to create an SDN pipe (labeled as 625) between the sender component 622 and the security service 630. The SDN pipe allows outgoing packets sent by the sender component 622 or incoming packets going to the sender component 622 to be redirected to the security service 630 for inspection before the packets are sent out of the SDN switch 620. In one embodiment, the SDN pipe is created by creating a first flow rule that forwards packets |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 573 of 1100

| No. | '111 Patent Claim 1 | Lin '400 |
|---|---|---|
| | | received in the ingress port 623-1 to the redirect port 623-2, and a second flow rule that forwards packets received in the redirect port 623-2 to the ingress port 623-1.").<br>  <br><br>*FIG. 6*<br>Fig. 6 (annotation added)<br><br>Lin '400 1:66-2:4 ("The security component inspects the outgoing packets for compliance with security policies and allows the outgoing packets to be forwarded to their destination when the outgoing packets pass inspection. The SDN controller may also insert a flow rule in the flow table of the SDN switch to bypass inspection of specified packets.").<br><br>Lin '400 3:21-24 ("The security service may inspect the packets, and forward the packets to an egress port of the switch toward the next hop, which may be another switch or a destination component (e.g., destination computer), for example.").<br><br>Lin '400 6:54-63 ("The security service 630 may forward those packets that pass inspection toward their destination by re-injecting the packets back into the SDN switch 620 by way of |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 574 of 1100

| No. | '111 Patent Claim 1 | Lin '400 |
|---|---|---|
| | | the re-inject port 623-3. Once back in the SDN switch 620 by way of the re-inject port 623-3, the flow rules that govern packets received in the ingress port 623-1 and the redirect port 623-2 no longer apply. Accordingly, the re-injected packets are forwarded to the egress port 623-4 (or some other port) toward the next hop in accordance with the L2 switching logic of the SDN computer network 600.").<br><br>Lin '400 7:23-27 ("In one embodiment, bypass flow rules are inserted in the flow tables 621 such that particular packets that do not need to be inspected are not redirected to the security service 630. This embodiment is explained with reference to example flow tables of Tables 2 and 3."). |

| No. | '111 Patent Claim 2 | Lin '400 |
|---|---|---|
| 2[a] | The method according to claim 1, wherein the instruction is 'probe', 'mirror', or 'terminate' instruction, and | Lin '400 discloses the method according to claim 1, wherein the instruction is a 'probe', a 'mirror', or a 'terminate' instruction.<br><br>For example, Lin '400 discloses flow rules to (1) redirecting a packet from its intended destination to a new destination (such as a security service) or (2) making a copy of a packet and sending the copy of the packet to a new destination (such as a security service, as well as a rule to drop packets from the network such that they are no longer forwarded to a destination within the network. Lin further discloses an instruction that, if the packet satisfies the criterion, to send the packet to the security service for inspection.<br><br>Lin '400 3:25-33 ("Another way of intercepting network traffic is to mirror the packets to be inspected on a switch that provides vendor specific mirroring application programming interface (API) as shown in FIG. 4. A user may make an API call such that particular packets that enter the ingress port of the switch are redirected or mirrored to the security service by way of a connection tunnel or a mirror port. The security service may forward the redirected or mirrored packets back to an egress port of the switch after inspection.").<br><br>Lin '400 7:10-22 ("Re-injecting packets that pass inspection consume[s] bandwidth, as the packets will have to be transmitted by the security service 630 to the re-inject port 623-3. For optimization, the SDN switch 620 may be configured to copy packets that are redirected to the security service 630 for inspection. This way, the security service 630 simply has to inform |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 575 of 1100

| No. | '111 Patent Claim 2 | Lin '400 |
|---|---|---|
| | | the SDN switch 620 an action to take on the packets based on the result of the inspection (see arrow 604). For example, the security service 630 may send an index identifying the packets and an action how to manipulate the packets. The action may instruct the SDN switch 620 to drop the copied packets, forward the copied packets to their destinations, quarantine the copied packets, etc."). |
| | | Lin '400 1:28-32 ("Example packet manipulation actions include forwarding a packet to a specific port, modifying one or more fields of the packet, asking the controller for action to perform on the packet, or dropping the packet."). |
| | | Lin '400 7:19-22 ("The action may instruct the SDN switch 620 to drop the copied packets, forward the copied packets to their destinations, quarantine the copied packets, etc."). |
| | | Lin '400 4:8-31 ("The SDN controller 610 provides a logically centralized framework for controlling the behavior of the SDN computer network 600. This is in marked contrast to traditional computer networks where the behavior of the computer network is controlled by low-level device configurations of switches and other network devices. The SDN controller 610 may include a flow policy database 611. The flow policy database 611 may comprise flow policies that are enforced by the controller 610 on network traffic transmitted over the SDN computer network 600. The flow policies may specify security policies that govern transmission of packets over the SDN computer network 600. The flow policies may be enforced in terms of flow rules (labeled as 624) that are stored in the flow tables 621 of the SDN switch 620. As a particular example, a flow policy in the flow policy database 611 may indicate inspection of particular packets (e.g., those that meet one or more conditions) by a security service 630. That flow policy may be implemented as a flow rule that forwards the particular packets received in an ingress port 623-1 to the redirect port 623-2 for inspection, for example."). |
| | | Lin '400 4:53-67 ("The SDN switch 620 may comprise one or more flow tables 621. The flow tables 621 may comprise one or more flow rules (labeled as 624) that indicate how to manipulate or process packets that are passing through the SDN switch 620. As a particular example, a flow rule may indicate that a packet received in the ingress port 623-1 is to be forwarded to the redirect port 623-2. Another flow rule may indicate that a packet received in the redirect port 623-2 is to be forwarded to the ingress port 623-1. The just mentioned pair |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 576 of 1100

| No. | '111 Patent Claim 2 | Lin '400 |
|---|---|---|
| | | of flow rules are redirect flow rules that create an SDN pipe between the sender component 622 and the security service 630, allowing the security service 630 to inspect packets sent by or going to the sender component 622. Table 1 shows an example flow table with flow rules that create an SDN pipe between the security service 630 and the sender component 622.").<br><br>Lin '400 6:1-12 ("The SDN controller 610 may insert flow rules in the flow tables 621 (see arrow 601) to create an SDN pipe (labeled as 625) between the sender component 622 and the security service 630. The SDN pipe allows outgoing packets sent by the sender component 622 or incoming packets going to the sender component 622 to be redirected to the security service 630 for inspection before the packets are sent out of the SDN switch 620. In one embodiment, the SDN pipe is created by creating a first flow rule that forwards packets received in the ingress port 623-1 to the redirect port 623-2, and a second flow rule that forwards packets received in the redirect port 623-2 to the ingress port 623-1."). |
| 2[b] | upon receiving by the network node the 'terminate' instruction, the method further comprising blocking, by the network node, the packet from being sent to the second entity and to the controller. | Lin '400 discloses upon receiving by the network node the 'terminate' instruction, the method further comprising blocking, by the network node, the packet from being sent to the second entity and to the controller.<br><br>For example, Lin '400 discloses blocking a packet from being sent to the destination node. If the controller is the destination, the packet would be also be blocked from being sent to the controller.<br><br>Lin '400 7:10-22 ("Re-injecting packets that pass inspection consume[s] bandwidth, as the packets will have to be transmitted by the security service 630 to the re-inject port 623-3. For optimization, the SDN switch 620 may be configured to copy packets that are redirected to the security service 630 for inspection. This way, the security service 630 simply has to inform the SDN switch 620 an action to take on the packets based on the result of the inspection (see arrow 604). For example, the security service 630 may send an index identifying the packets and an action how to manipulate the packets. The action may instruct the SDN switch 620 to drop the copied packets, forward the copied packets to their destinations, quarantine the copied packets, etc."). |



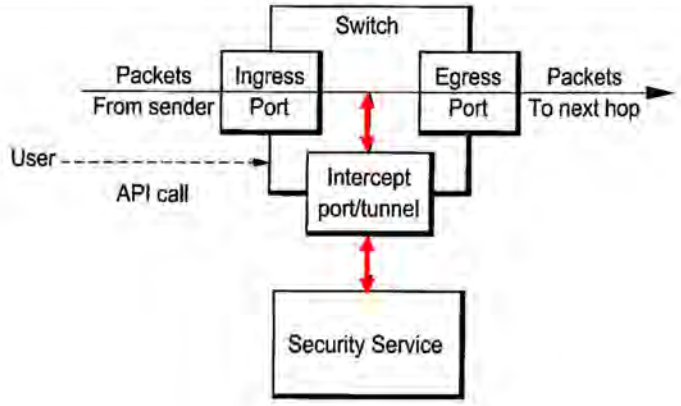
Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 577 of 1100

| No. | '111 Patent Claim 2 | Lin '400 |
|---|---|---|
| | | Lin '400 1:28-32 ("Example packet manipulation actions include forwarding a packet to a specific port, modifying one or more fields of the packet, asking the controller for action to perform on the packet, or dropping the packet."). <br><br> Lin '400 7:19-22 ("The action may instruct the SDN switch 620 to drop the copied packets, forward the copied packets to their destinations, quarantine the copied packets, etc."). |

| No. | '111 Patent Claim 3 | Lin '400 |
|---|---|---|
| 3[a] | The method according to claim 1, wherein the instruction is a 'probe', a 'mirror', or a 'terminate' instruction, and | Lin '400 discloses the method according to claim 1, wherein the instruction is a 'probe', a 'mirror', or a 'terminate' instruction. <br><br> *See supra* Claim 2[a]. |
| 3[b] | upon receiving by the network node the 'mirror' instruction and responsive to the packet satisfying the criterion, the method further comprising sending the packet, by the network node, to the second entity and to the controller. | Lin '400 discloses upon receiving by the network node the 'mirror' instruction and responsive to the packet satisfying the criterion, the method further comprising sending the packet, by the network node, to the second entity and to the controller. <br><br> For example, Lin '400 discloses mirroring the packets that enter the ingress port of the switch to continue on to the next hop as well as be sent to the security service. <br><br> Lin '400 3:25-33 ("Another way of intercepting network traffic is to mirror the packets to be inspected on a switch that provides vendor specific mirroring application programming interface (API) as shown in FIG. 4. A user may make an API call such that particular packets that enter the ingress port of the switch are redirected or mirrored to the security service by way of a connection tunnel or a mirror port. The security service may forward the redirected or mirrored packets back to an egress port of the switch after inspection."). |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 578 of 1100

| No. | '111 Patent Claim 3 | Lin '400 |
|---|---|---|
| | |  <br>*FIG. 4*<br>Fig. 4 (annotation added) |

| No. | '111 Patent Claim 4 | Lin '400 |
|---|---|---|
| 4[a] | The method according to claim 1, wherein the instruction is 'probe', 'mirror', or 'terminate' instruction, and | Lin '400 discloses the method according to claim 1, wherein the instruction is 'probe', 'mirror', or 'terminate' instruction.<br><br>*See supra* Claim 2[a]. |
| 4[b] | upon receiving by the network node the 'probe' instruction and responsive to the packet satisfying the criterion, the method further comprising: sending the packet, by | Lin '400 discloses upon receiving by the network node the 'probe' instruction and responsive to the packet satisfying the criterion, the method further comprising: sending the packet, by the network node, to the controller.<br><br>For example, Lin '400 discloses a command to send a packet to a security service for further inspection upon positive identification. A person of ordinary skill in the art would understand that the security service could be located in the SDN controller. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Lin '400 is found to not meet this limitation, upon receiving by the network |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 579 of 1100

| No. | '111 Patent Claim 4 | Lin '400 |
|---|---|---|
| | the network node, to the controller; | node the 'probe' instruction and responsive to the packet satisfying the criterion, the method further comprising: sending the packet, by the network node, to the controller would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>Lin '400 6:40-63 ("After the redirect flow rules for creating the SDN pipe are inserted in the flow tables 621, any packet received by the SDN switch 620 in the ingress port 623-1 will be identified as to be forwarded to the redirect port 623-2, and any packet received by the SDN switch 620 in the redirect port 623-2 will be identified as to be forwarded to the ingress port 623-1 (see arrow 602). This allows the security service 630 to receive from the redirect port 623-2 all outgoing packets sent by the sender component 622 to the ingress port 623-1. The security service 630 may inspect the outgoing packets for compliance with security policies. The security service 630 may drop, or perform other security response, to packets that do not pass inspection (e.g., packets that do not meet firewall policies, packets containing prohibited payload, packets with malicious content, etc.). The security service 630 may forward those packets that pass inspection toward their destination by re-injecting the packets back into the SDN switch 620 by way of the re-inject port 623-3. Once back in the SDN switch 620 by way of the re-inject port 623-3, the flow rules that govern packets received in the ingress port 623-1 and the redirect port 623-2 no longer apply. Accordingly, the re-injected packets are forwarded to the egress port 623-4 (or some other port) toward the next hop in accordance with the L2 switching logic of the SDN computer network 600.").<br><br>Lin '400 5:45-55 ("The security service 630 may inspect packets for compliance/non-compliance with security policies, such as for presence of malicious code, compliance with firewall rules and access control lists, network intrusion detection, and other computer network security services. The security service 630 may employ conventional packet inspection algorithms. The security service 630 may comprise the Trend Micro Deep Security™ service, for example. The security service 630 may also comprise a physical machine, e.g., a server computer, an appliance, a gateway computer, etc.").<br><br>Lin '400 7:10-22 ("Re-injecting packets that pass inspection consume bandwidth, as the packets will have to be transmitted by the security service 630 to the re-inject port 623-3. For optimization, the SDN switch 620 may be configured to copy packets that are redirected to the security service 630 for inspection. This way, the security service 630 simply has to inform the SDN switch 620 an action to take on packets based on the result of the inspection |

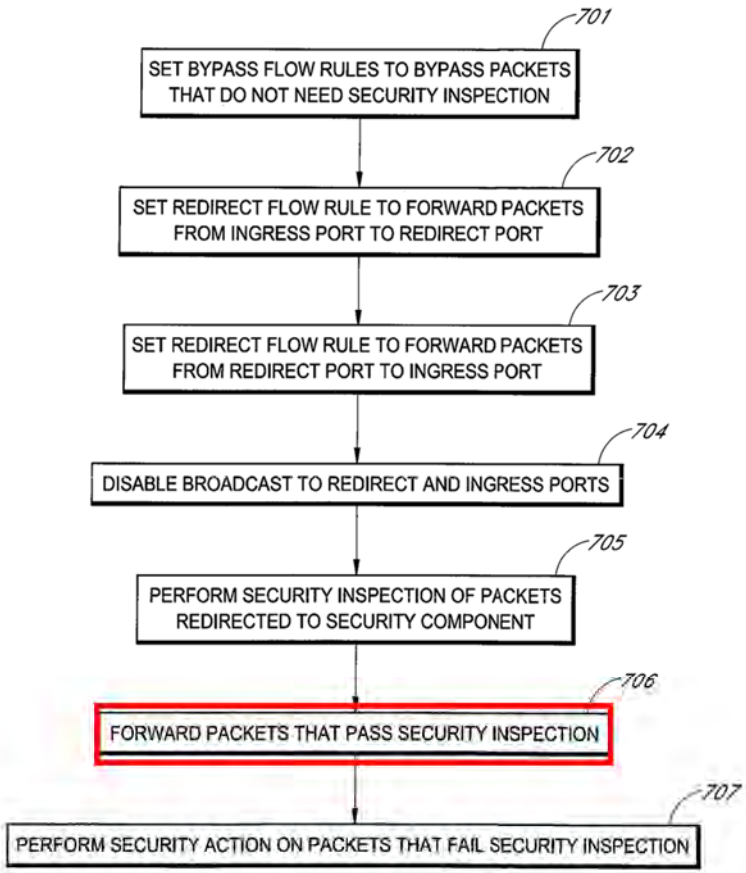


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 580 of 1100

| No. | '111 Patent Claim 4 | Lin '400 |
|---|---|---|
| | | (see arrow 604). For example, the security service 630 may send an index identifying the packets and an action on how to manipulate the packets. The action may instruct the SDN switch 620 to drop the copied packets, forward the copied packets to their destinations, quarantine the copied packets, etc."). <br><br> Lin '400 8:33-45 ("The security service 630 receives the outgoing packets from the redirect port 623-2 (see arrow 654) and inspects the outgoing packets. After inspection, the security service 630 re-injects the outgoing packets (e.g., outgoing packets that passed inspection) back into the SDN switch 620 by way of the re-inject port 623-3 (see arrow 655). The SDN switch 620 receives the outgoing packets on the re-inject port 623-3. The SDN switch 620 forwards the outgoing packets from the re-inject port 623-3 to the egress port 623-4 in accordance with the L2 switching logic of the SDN computer network 600 (see arrow 657). The outgoing packets exit the SDN switch 620 through the egress port 623-4 (see arrow 658) and move towards their destination."). |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 581 of 1100

| No. | '111 Patent Claim 4 | Lin '400 |
|---|---|---|
| | |  <br>*FIG. 9*<br><br>Figure 9 (annotation added)<br><br>Lin '400 4:53-67 ("The SDN switch 620 may comprise one or more flow tables 621. The flow tables 621 may comprise one or more flow rules (labeled as 624) that indicate how to manipulate or process packets that are passing through the SDN switch 620. As a particular |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 582 of 1100

| No. | '111 Patent Claim 4 | Lin '400 |
|---|---|---|
| | | example, a flow rule may indicate that a packet received in the ingress port 623-1 is to be forwarded to the redirect port 623-2. Another flow rule may indicate that a packet received in the redirect port 623-2 is to be forwarded to the ingress port 623-1. The just mentioned pair of flow rules are redirect flow rules that create an SDN pipe between the sender component 622 and the security service 630, allowing the security service 630 to inspect packets sent by or going to the sender component 622. Table 1 shows an example flow table with flow rules that create an SDN pipe between the security service 630 and the sender component 622.").<br><br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Lin '400 in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 4[b] of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references.<br><br>For example, Kempf discloses sending the packet from the network element to the controller or another table, in response to the packet matching the action in the flow table.<br><br>Kempf at [0044] ("FIG. 1 is a diagram of one embodiment of an example network with an OpenFlow switch, conforming to the OpenFlow 1.0 specification. The OpenFlow 1.0 protocol enables a controller 101 to connect to an OpenFlow 1.0 enabled switch 109 using a secure channel 103 and control a single forwarding table 107 in the switch 109. The controller 101 is an external software component executed by a remote computing device that enables a user to configure the Open-Flow 1.0 switch 109. The secure channel 103 can be provided by any type of network including a local area network (LAN) or a wide area network (WAN), such as the Internet.")<br><br>Kempf at [0045] ("FIG. 2 is a diagram illustrating one embodiment of the contents of a flow table entry. The forwarding table 107 is populated with entries consisting of a rule 201 defining matches for fields in packet headers; an action 203 associated to the flow match; and a collection of statistics 205 on the flow. When an incoming packet is received a lookup for a matching rule is made in the flow table 107. If the incoming packet matches a particular rule, the associated action defined in that flow table entry is performed on the packet.") |

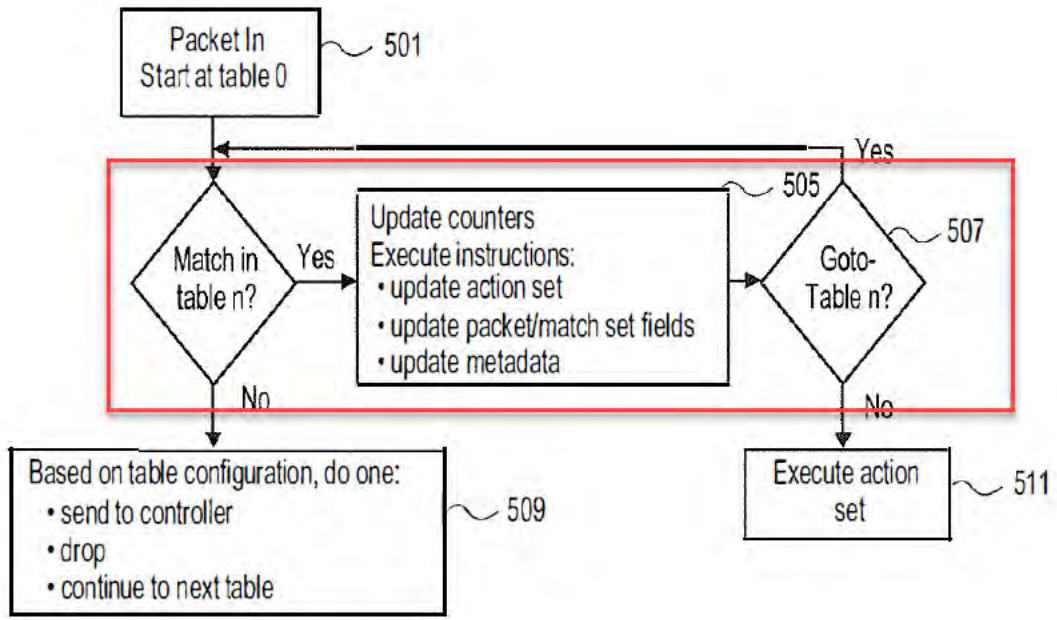


~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 583 of 1100

| No. | '111 Patent Claim 4 | Lin '400 |
|---|---|---|
| | | Kempf at [0046] ("A rule 201 contains key fields from several headers in the protocol stack, for example source and destination Ethernet MAC addresses, source and destination IP addresses, IP protocol type number, incoming and outgoing TCP or UDP port numbers. To define a flow, all the available matching fields may be used. But it is also possible to restrict the matching rule to a subset of the available fields by using wildcards for the unwanted fields.")<br><br>Kempf at [0047] ("The actions that are defined by the specification of OpenFlow 1.0 are Drop, which drops the matching packets; Forward, which forwards the packet to one or all outgoing ports, the incoming physical port itself, the controller via the secure channel, or the local networking stack (if it exists). OpenFlow 1.0 protocol data units (PDU s) are defined with a set of structures specified using the C programming language. Some of the more commonly used messages are: report switch configuration message; modify state messages (in-cluding a modify flow entry message and port modification message); read state messages, where while the system is running, the datapath may be queried about its current state using this message; and send packet message, which is used when the controller wishes to send a packet out through the datapath.")<br><br>Kempf at [0050] ("FIG. 4 illustrates one embodiment of the processing of packets through an OpenFlow 1.1 switched packet pro-cessing pipeline. A received packet is compared against each of the flow tables 401. After each flow table match, the actions are accumulated into an action set. If processing requires matching against another flow table, the actions in the matched rule include an action directing processing to the next table in the pipeline. Absent the inclusion of an action in the set to execute all accumulated actions immediately, the actions are executed at the end 403 of the packet processing pipeline. An action allows the writing of data to a metadata register, which is carried along in the packet processing pipe-line like the packet header.")<br><br>Kempf at [0091] ("When a packet header matches a rule associated with the virtual port, the GTP TEID is written into the lower 32 bits of the metadata and the packet is directed to the virtual port. The virtual port calculates the hash of the TEID and looks up the tunnel header information in the tunnel header table. If no such tunnel information is present, the packet is forwarded to the controller with an error indication. Other-wise, the virtual port constructs a GTP tunnel header and encapsulates the packet. Any DSCP bits or VLAN priority bits are |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 584 of 1100

| No. | '111 Patent Claim 4 | Lin '400 |
|---|---|---|
| | | additionally set in the IP or MAC tunnel headers, and any VLAN tags or MPLS labels are pushed onto the packet. The encapsulated packet is forwarded out the physical port to which the virtual port is bound.")<br><br>Kempf at [0106] ("This encapsulates the packet and sends it to the OpenFlow controller.")<br><br>Kempf at Figure 5 (annotation added)<br><br>  <br><br>FIG. 5<br><br>Kempf at Figure 2 (annotation added) |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 585 of 1100

| No. | '111 Patent Claim 4 | Lin '400 |
|---|---|---|

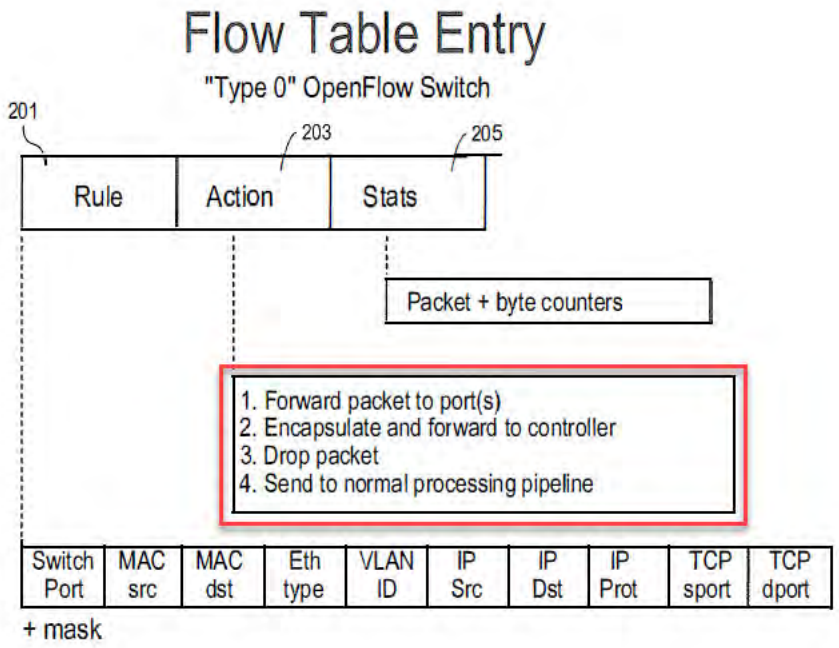



FIG. 2

For example, Swenson discloses determining by the steering device monitors flows that match one or more signatures or criteria of the packet. Swenson further discloses that when a matching flow is detected the steering device forwards the packet to the network controller.

Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 586 of 1100

| No. | '111 Patent Claim 4 | Lin '400 |
|---|---|---|
| | | steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0028] ("In contrast to conventional inline TCP throughput monitoring devices that monitor every single data packets transmitted and received, the network controller 140 is an "out-of-band" computer server that interfaces with the steer-ing device 130 to selectively inspect user flows of interest. The network controller 140 may further identify user flows (e.g., among the flows of interest) for optimization. In one embodiment, the network controller 140 may be imple-mented at the steering device 130 to monitor traffic. In other embodiments, the network controller 140 is coupled to and communicates with the steering device 130 for traffic moni-toring and optimization. When queried by the steering device 130, the network controller 140 determines if a given network flow should be ignored, monitored further or optimized. Opti-mization of a flow is often decided at the beginning of the flow because it is rarely possible to switch to optimized content mid-stream once non-optimized content delivery has begun. However, the network controller 140 may determine that existing flows associated with a particular subscriber or other entity should be optimized. In turn, new flows ( e.g., resulting from seek requests in media, new media requests, resume after pause, etc.) determined to be associated with the entity may be optimized. The network controller 140 uses the net-work state as well as historical traffic data in its decision for monitoring and optimization. Knowledge on the current net-work state, such as congestion, deems critical when it comes to data optimization.")<br><br>Swenson at [0029] ("As a flow is sent to the network controller 140 for inspection, historical network traffic data stored at the net-work controller 140 may be searched. The historical network traffic data includes information such as subscriber informa-tion, the cell towers to which the user devices attached, rout-ers through which the traffic is passing, geography regions, the backhaul segments, and time-of-day of the flows. For example, in a mobile network, the cell tower to which a user device is attached can be most useful, since it is the location where most congestion occurs due to limited bandwidth and high cost of the radio access network infrastructure. The network controller 140 looks into the historical traffic data for the average of the bandwidth per user at the particular cell tower. The network controller |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 587 of 1100

| No. | '111 Patent Claim 4 | Lin '400 |
|---|---|---|
| | | 140 can then estimate the amount ofbandwidth or degree of congestion for the new flow based on the historical record.")<br><br>Swenson at [0038] ("Turning back to FIG. 1, the network controller 140 allows network operators to apply fine granular optimization policies to ensure high quality of experience (QoE) based on cell tower congestion, device types, subscriber profiles and service plans with lower hardware and software costs. The architecture of the network controller 140 provides an excel-lent fit for the net neutrality guideline of"reasonable network management", and better compliance to the copyright law (DMCA) than solutions that rely on long-term caching. Hav-ing the ability of monitoring network traffic on a per sub-scriber, per flow, or per video file basis, the network controller 140 also selectively monitors and optimizes only a subset of traffic that benefits from optimization the most, thus achiev-ing both scalability and efficiency for optimization at a com-petitive price-point. The core element of the network control-ler 140 lies in its mechanisms for congestion detection and mitigation, which allows optimization resources to be utilized in the most efficient and surgical manner.")<br><br>Swenson at [0039] ("Referring now to FIG. 3, it illustrates one embodi-ment of an example architecture of the network controller 140 for providing selective real-time network monitoring and subscriber identification. The network controller 140 com-prises a flow analyzer 312, a policy engine 314, a steering device interface 316, a video optimizer redirector 318, a flow cache 322, and a subscriber log 324. In other embodiments, the network controller 140 may include additional, fewer, or different components for various applications. Conventional components such as network interfaces, security functions, failover servers, management and network operations con-soles, and the like are not shown so as to not obscure the details of the system architecture.")<br><br>Swenson at [0045] ("The steering device interface 316 interacts with an external routing appliance, such as the steering device 130 to divert portions of the network traffic ( e.g., large object net-work flows). Existing routing appliances in most carrier net-works are designed to handle large amounts of network traf-fic. They are not, however, ideal devices to operate for monitoring and analysis individual flows. Through the steer-ing device interface 316, the network controller 140 may communicate with the external routing appliances, such as the steering device 130, to steer a portion of network traffic to the network controller 140 when certain conditions are met. Generally, network flows of interest to the network controller 140 |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 588 of 1100

| No. | '111 Patent Claim 4 | Lin '400 |
|---|---|---|
| | | contain larger media objects, such as videos and images. In one embodiment, the smaller flows, such as web page and text information, are not exchanged over the steering device interface 316.")<br><br>Swenson at [0059] ("In one embodiment, as the steering device 130 monitors network responses, it is looking for flows that match one or more signatures for video and images. When a match-ing flow is detected, the steering device 130 forwards the HTTP request and a portion of the HTTP response to the network controller 140 over the ICAP client interface 404. After receiving the request and the portion of response at the ICAP server interface 406, the flow analyzer 312 of the net-work controller 140 performs a deep flow inspection to deter-mine if the flow is worth bandwidth monitoring and/or user detection. For example, the flow inspection performed by the flow analyzer 312 may determine if the flow indeed contains large or medium object ( e.g., larger than 50 kB), and/or if the source IP address of the flow is from a user or a group of users that are required to be monitored by policies. The flow ana-lyzer 312 may also determine if the flow needs to be opti-mized based on historical flow statistical data.")<br><br>Swenson at [0060] ("If the flow is deemed of interest, the steering device 130 is notified to steer the flow through the network controller 140. This is known as the "continue" working mode for bandwidth monitoring. In the "continue" mode, the network controller 140 interfaces with the steering device 130 to func-tion, on-demand, as a traditional inline network element for flows deemed of interest. Thus, the network controller 140 ingests the network flow for inspection and subsequently forwards the network flow on the network response path. For example, for this particular flow, the origin server 160 responds to the user request by sending video or images over the network link 413 to the steering device 130, which for-wards the video or images to the network controller 140 over a network link 414. After the network controller 140 updates the flow statistics, the video or images are returned to the steering device 130 over a network link 415, which transmits the video or images to the user device 110 over the network link 416.")<br><br>Swenson at [0065] ("FIG. 5 is a block diagram illustrating an example event trace of "continue" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 512 to retrieve content from the origin server 160. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 589 of 1100

| No. | '111 Patent Claim 4 | Lin '400 |
|---|---|---|
| | | The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 512 to the intended origin server 160 and receives a response 514 back from the origin server 160. The steering device 130 then sends an ICAP request message 516 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 518. Upon receiving the instruc-tion, the steering device 130 re-writes the response 514 to an HTTP redirect response 520, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 520 directly to the user device 110. In case the flow dose not contain video or image objects, or the network controller 140 determines not to monitor the flow, the steering device 13 0 would forward the response to the user device 110.")<br><br>Swenson at [0069] ("FIG. 6 is a block diagram illustrating an example event trace of "counting" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 612 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 612 to the intended origin server 160 and receives a response 614 back from the origin server 160. The steering device 130 then sends an ICAP request message 616 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 618. Upon receiving the instruc-tion, the steering device 130 re-writes the response 614 to an HTTP redirect response 620, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 620 directly to the user device 110. In case the flow dose not contain video or image objects that need to be redirected, the steering device 130 would forward the response to the user device 110.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 590 of 1100

| No. | '111 Patent Claim 4 | Lin '400 |
|---|---|---|
| 4[c] | responsive to receiving the packet, analyzing the packet, by the controller; | Lin '400 discloses responsive to receiving the packet, analyzing the packet, by the controller.<br><br>For example, Lin '400 discloses analyzing packets by a security service that inspects the packets. A person of ordinary skill in the art would understand that the security service could be located in the SDN controller. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Lin '400 is found to not meet this limitation, responsive to receiving the packet, analyzing the packet, by the controller would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>Lin '400 3:11-24 ("Network security vendors provide network security services, such as firewall or deep packet inspection (DPI). Generally speaking, to provide network security services, packets of network traffic are intercepted for inspection. One way of intercepting network traffic is to place the security service in the middle of the packet forwarding path. This is illustrated in FIG. 3, where packets from a sender component (e.g., a sender computer) are received in an ingress port of a switch, forwarded to an egress port of the switch, and forwarded to the ingress port of a security component, such as a security service. The security service may inspect the packets, and forward the packets to an egress port of the switch toward the next hop, which may be another switch or a destination component (e.g., destination computer), for example.").<br><br>Lin '400 5:37-55 ("The SDN computer network 600 may include a security component in the form of the security service 630. The security service 630 may comprise a virtual machine that provides computer network security services, such as packet inspection, for the sender component 622 and other virtual machines. For example, the security service 630 may comprise a virtual machine with a virtual network interface card that is coupled to the redirect port 623-2 and re-inject port 623-3 of the SDN switch 620. The security service 630 may inspect packets for compliance/non-compliance with security policies, such as for presence of malicious code, compliance with firewall rules and access control lists, network intrusion detection, and other computer network security services. The security service 630 may employ conventional packet inspection algorithms. The security service 630 may comprise the Trend Micro Deep Security™ service, for example. The security service 630 may also comprise a physical machine, e.g., a server computer, an appliance, a gateway computer, etc."). |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 591 of 1100

| No. | '111 Patent Claim 4 | Lin '400 |
|---|---|---|
| | | Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Lin '400 in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 4(c) of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references.<br><br>For example, Swenson discloses the network controller comprising a flow analyzer for analyzing and inspecting the packet.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0028] ("In contrast to conventional inline TCP throughput monitoring devices that monitor every single data packets transmitted and received, the network controller 140 is an "out-of-band" computer server that interfaces with the steer-ing device 130 to selectively inspect user flows of interest. The network controller 140 may further identify user flows (e.g., among the flows of interest) for optimization. In one embodiment, the network controller 140 may be imple-mented at the steering device 130 to monitor traffic. In other embodiments, the network controller 140 is coupled to and communicates with the steering device 130 for traffic moni-toring and optimization. When queried by the steering device 130, the network controller 140 determines if a given network flow should be ignored, monitored further or optimized. Opti-mization of a flow is often decided at the beginning of the flow because it is rarely possible to switch to optimized content mid-stream once non-optimized content delivery has begun. However, the network controller 140 may determine that existing flows associated with a particular subscriber or other entity should be optimized. In turn, new |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 592 of 1100

| No. | '111 Patent Claim 4 | Lin '400 |
|---|---|---|
| | | flows ( e.g., resulting from seek requests in media, new media requests, resume after pause, etc.) determined to be associated with the entity may be optimized. The network controller 140 uses the net-work state as well as historical traffic data in its decision for monitoring and optimization. Knowledge on the current net-work state, such as congestion, deems critical when it comes to data optimization.")<br><br>Swenson at [0029] ("As a flow is sent to the network controller 140 for inspection, historical network traffic data stored at the net-work controller 140 may be searched. The historical network traffic data includes information such as subscriber informa-tion, the cell towers to which the user devices attached, rout-ers through which the traffic is passing, geography regions, the backhaul segments, and time-of-day of the flows. For example, in a mobile network, the cell tower to which a user device is attached can be most useful, since it is the location where most congestion occurs due to limited bandwidth and high cost of the radio access network infrastructure. The network controller 140 looks into the historical traffic data for the average of the bandwidth per user at the particular cell tower. The network controller 140 can then estimate the amount of bandwidth or degree of congestion for the new flow based on the historical record.")<br><br>Swenson at [0038] ("Turning back to FIG. 1, the network controller 140 allows network operators to apply fine granular optimization policies to ensure high quality of experience (QoE) based on cell tower congestion, device types, subscriber profiles and service plans with lower hardware and software costs. The architecture of the network controller 140 provides an excel-lent fit for the net neutrality guideline of "reasonable network management", and better compliance to the copyright law (DMCA) than solutions that rely on long-term caching. Hav-ing the ability of monitoring network traffic on a per sub-scriber, per flow, or per video file basis, the network controller 140 also selectively monitors and optimizes only a subset of traffic that benefits from optimization the most, thus achiev-ing both scalability and efficiency for optimization at a com-petitive price-point. The core element of the network control-ler 140 lies in its mechanisms for congestion detection and mitigation, which allows optimization resources to be utilized in the most efficient and surgical manner.")<br><br>Swenson at [0039] ("Referring now to FIG. 3, it illustrates one embodi-ment of an example architecture of the network controller 140 for providing selective real-time network monitoring and subscriber identification. The network controller 140 com-prises a flow |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 593 of 1100

| No. | '111 Patent Claim 4 | Lin '400 |
|---|---|---|
| | | analyzer 312, a policy engine 314, a steering device interface 316, a video optimizer redirector 318, a flow cache 322, and a subscriber log 324. In other embodiments, the network controller 140 may include additional, fewer, or different components for various applications. Conventional components such as network interfaces, security functions, failover servers, management and network operations con-soles, and the like are not shown so as to not obscure the details of the system architecture.")<br><br>Swenson at [0059] ("In one embodiment, as the steering device 130 monitors network responses, it is looking for flows that match one or more signatures for video and images. When a match-ing flow is detected, the steering device 130 forwards the HTTP request and a portion of the HTTP response to the network controller 140 over the ICAP client interface 404. After receiving the request and the portion of response at the ICAP server interface 406, the flow analyzer 312 of the net-work controller 140 performs a deep flow inspection to deter-mine if the flow is worth bandwidth monitoring and/or user detection. For example, the flow inspection performed by the flow analyzer 312 may determine if the flow indeed contains large or medium object ( e.g., larger than 50 kB), and/or if the source IP address of the flow is from a user or a group of users that are required to be monitored by policies. The flow ana-lyzer 312 may also determine if the flow needs to be opti-mized based on historical flow statistical data.")<br><br>Swenson at [0060] ("If the flow is deemed of interest, the steering device 130 is notified to steer the flow through the network controller 140. This is known as the "continue" working mode for bandwidth monitoring. In the "continue" mode, the network controller 140 interfaces with the steering device 130 to func-tion, on-demand, as a traditional inline network element for flows deemed of interest. Thus, the network controller 140 ingests the network flow for inspection and subsequently forwards the network flow on the network response path. For example, for this particular flow, the origin server 160 responds to the user request by sending video or images over the network link 413 to the steering device 130, which for-wards the video or images to the network controller 140 over a network link 414. After the network controller 140 updates the flow statistics, the video or images are returned to the steering device 130 over a network link 415, which transmits the video or images to the user device 110 over the network link 416.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 594 of 1100

| No. | '111 Patent Claim 4 | Lin '400 |
|---|---|---|
| | | Swenson at [0061] ("Once a flow is reported to the network controller 140, a flow cache entry is created for the flow in the flow cache 322. The flow cache entry keeps track of the flow and its associated bandwidth. For a flow that is marked in "continue" mode, each time the steering device 130 forwards a next portion of the flow payload to the network controller 140, the flow cache 3 22 updates the number of bytes for transmitted in the flow. By monitoring the number of bytes per flow over time, the flow analyzer 312 is capable of determining an estimate value of bandwidth associated with flow. Further-more, since the steering device 130 does not have infinite packet buffers, if congestion happens on the network link 416 from the steering device 130 to the user device 110, the TCP congestion control mechanism kicks in at the steering device 130, which may slows down and/or eventually stop receiving data over the network link 413 from origin server 160. During the congestion, the steering device 130 would not forward any data to the network controller 140, since the link 416 is congested and the network controller 140 would not be able to transmit data to the user device 110. Therefore, as an inline element, the network controller 140 can detect network con-gestions and estimate bandwidth associated with any flows of interest selected by the network controller 140. However, in the "continue" mode, the network controller 140 does not modify and transform the HTTP messaged it receives over the ICAP interface. The network controller 140 simply updates the flow statistics and returns the video or images to the steering device 130 for transmission to the user device 110.")<br><br>Swenson at [0065] ("FIG. 5 is a block diagram illustrating an example event trace of "continue" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 512 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 512 to the intended origin server 160 and receives a response 514 back from the origin server 160. The steering device 130 then sends an ICAP request message 516 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 518. Upon receiving the instruc-tion, the steering device 130 re-writes the response 514 to an HTTP redirect response 520, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP |

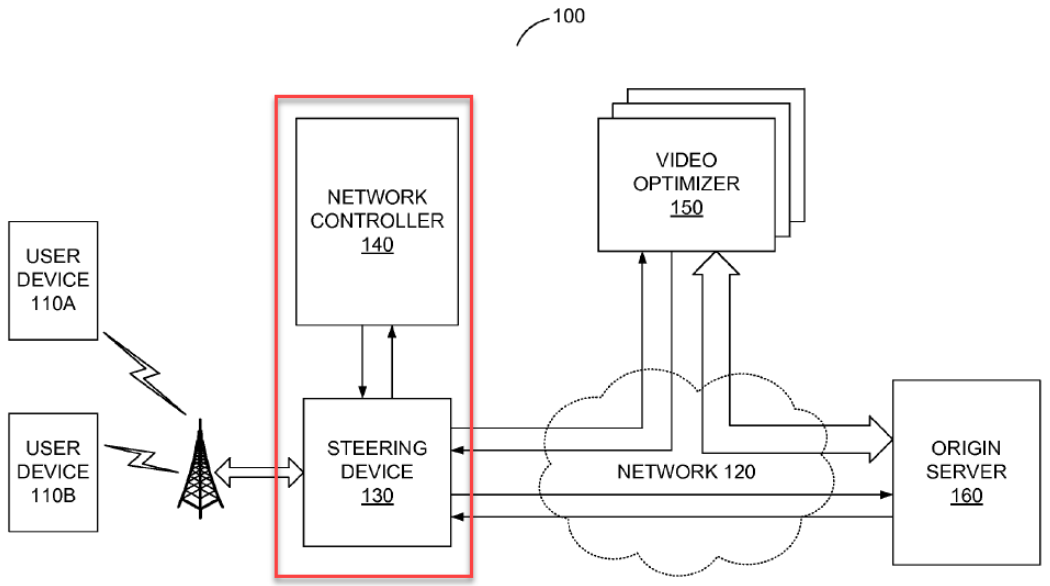


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 595 of 1100

| No. | '111 Patent Claim 4 | Lin '400 |
|---|---|---|
| | | redirect request 520 directly to the user device 110. In case the flow dose not contain video or image objects, or the network controller 140 determines not to monitor the flow, the steering device 13 0 would forward the response to the user device 110.")<br><br>Swenson at [0069] ("FIG. 6 is a block diagram illustrating an example event trace of "counting" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 612 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 612 to the intended origin server 160 and receives a response 614 back from the origin server 160. The steering device 130 then sends an ICAP request message 616 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 618. Upon receiving the instruc-tion, the steering device 130 re-writes the response 614 to an HTTP redirect response 620, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 620 directly to the user device 110. In case the flow dose not contain video or image objects that need to be redirected, the steering device 130 would forward the response to the user device 110.")<br><br>Swenson at [0071] ("After receiving the request, the video optimizer 150 forwards the video HTTP GET requests 622 to the origin server 160 and in return, receives a video file 624 from the origin server 160. The video optimizer 150 transcodes the video file to a format usable by the client device 110 based on network bandwidth available to the user device 110. The optimized video 626 is then transmitted from the video opti-mizer 150 to the steering device 130. In one embodiment, the steering device 130 intercepts the optimized video 626. The steering device 130 will then send an ICAP request to the network controller 140 for inspection. The network controller 140 deems this flow to be monitored and sends ICAP response 630. The steering device 130 then allows the flow to go through to the user device 110. The steering device 130 next sends periodic ICAP "counting" updates 632 to the network controller 140 until the flow completes. As such, the client receives the optimized video 626 for substantially real-time playback on an application executing on the user device 110.") |

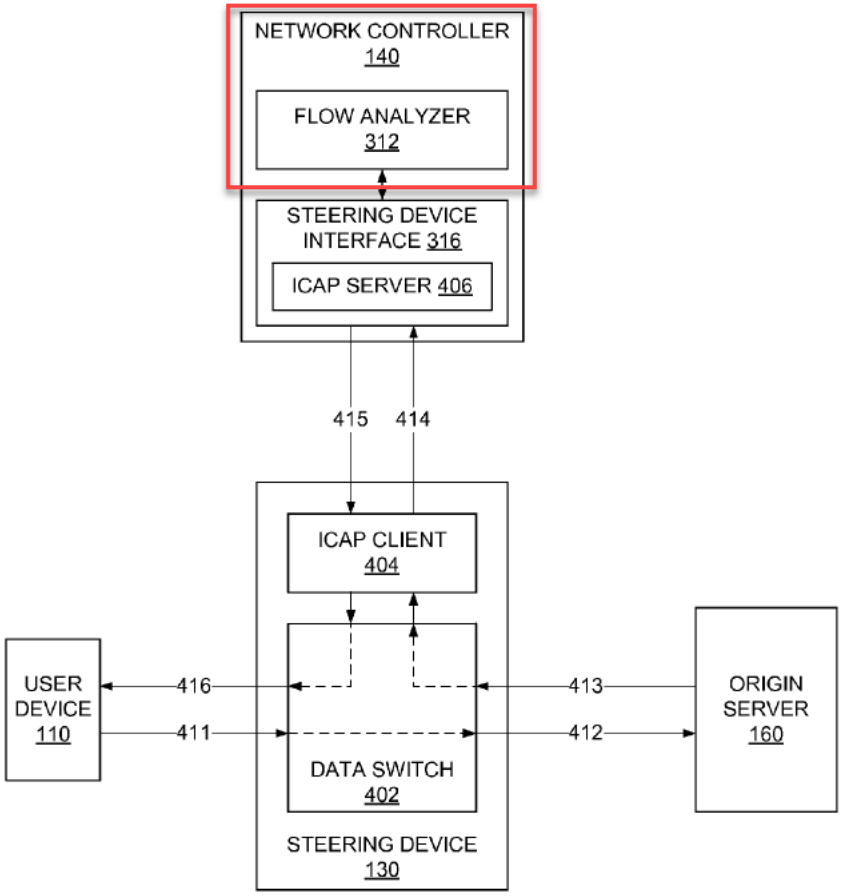


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 596 of 1100

| No. | '111 Patent Claim 4 | Lin '400 |
|---|---|---|
| | | Swenson at Figure 1 (annotation added)<br>  <br><br>FIG. 1<br>Swenson at Figure 4A (annotation added) |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 597 of 1100

| No. | '111 Patent Claim 4 | Lin '400 |
|---|---|---|
| | |  <br>FIG. 4A<br><br>For example, Copeland discloses analyzing packets received by the intrusion detection engine on the monitoring appliance. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 598 of 1100

| No. | '111 Patent Claim 4 | Lin '400 |
|---|---|---|
| | | Copeland at [0021] ("The present invention provides an accurate and reliable method for detecting network attacks through the use of sampled packet headers that are provided by a source such as that as defined in sFlow and further based in large part on "flows" as opposed to signatures or anomalies. By utilizing the host and flow information structures that are inherent with flow-based analysis and applying rules of statistical significance and analysis, the intrusion detection system can operate with sampled data such as provided by sFlow in order to provide a hybrid solution that combines many of the benefits of a packet capture implementation with the distributed nature of an IDS that operates on Netflow, thus providing an enhanced wide-area IDS solu-tion.")<br><br>Copeland at [0023] ("According to one aspect of the invention, the detection system works by assigning sampled data packets to various client/server ( C/S) flows. Statistics are collected for each determined flow. Then, the flow statistics are analyzed to determine if the flow appears to be legitimate traffic or possible suspicious activity. A value, referred to as a "concern index," is assigned to each flow that appears suspicious. By assigning a value to each flow that appears suspicious and adding that value to an accumulated concern index associated with the responsible host, it is possible to identify hosts that are engaged in intruder activity without generation of significant unwarranted false alarms. When the concern index value of a host exceeds a preset alarm value, an alert is issued and appropriate action can be taken.")<br><br>Copeland at [0024] ("Generally speaking, the intrusion detection system analyzes network communication traffic for potential detrimental activity. The system collects flow data from sampled packet headers between two hosts or Internet Protocol (IP) addresses. Collecting flow data from packet headers asso-ciated with a single service where at least one port remains constant allows for more efficient analysis of the flow data. The collected flow data is analyzed to assign a concern index value to the flow based upon a probability that the flow was not normal for data communications. A host list is main-tained containing an accumulated concern index derived from the flows associated with the host. Once the accumu-lated concern index has exceeded an alarm threshold value, an alarm signal is generated.")<br><br>Copeland at [0027] ("According to one aspect of the invention, the detection system works by assigning sampled data packets to various client/server ( C/S) flows. Statistics are collected for each determined flow. Then, the flow statistics are analyzed to determine if the flow |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 599 of 1100

| No. | '111 Patent Claim 4 | Lin '400 |
|---|---|---|
| | | appears to be legitimate traffic or possible suspicious activity. A value, referred to as a "concern index," is assigned to each flow that appears suspicious. By assigning a value to each flow that appears suspicious and adding that value to an accumulated concern index associated with the responsible host, it is possible to identify hosts that are engaged in intruder activity without generation of significant unwarranted false alarms. When the concern index value of a host exceeds a preset alarm value, an alert is issued and appropriate action can be taken.")<br><br>Copeland at [0028] ("Generally speaking, the intrusion detection system analyzes network communication traffic for potential detri-mental activity. The system collects flow data from sampled packet headers between two hosts or Internet Protocol (IP) addresses. Collecting flow data from packet headers asso-ciated with a single service where at least one port remains constant allows for more efficient analysis of the flow data. The collected flow data is analyzed to assign a concern index value to the flow based upon a probability that the flow was not normal for data communications. A host list is main-tained containing an accumulated concern index derived from the flows associated with the host. Once the accumu-lated concern index has exceeded an alarm threshold value, an alarm signal is generated.")<br><br>Copeland at [0063] ("Consequently, abnormal flows and/or events iden-tified by the intrusion detection engine 155 will raise the concern index (CI) for the associated host. The intrusion detection engine 155 analyzes the data flow between IP devices. However, different types of services have different flow characteristics associated with that service. Therefore, a C/S flow can be determined by the packets exchanged between the two hosts dealing with the same service.")<br><br>Copeland at [0065] ("The intrusion detection engine 155 analyzes the flow data 160 to determine if the flow appears to be legitimate traffic or possible suspicious activity. Flows with suspicious activity are assigned a predetermined concern index (CI) value based upon a heuristically predetermined assessment of the significance of the threat of the particular traffic or flow or suspicious activity. The flow concern index values have been derived heuristically from extensive net-work traffic analysis. Concern index values are associated with particular hosts and stored in the host data structure 166 (FIG. 1). Exemplary concern index values for various exemplary flow-based events and other types of events are illustrated in connection with FIGS. 6 and 7.) |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 600 of 1100

| No. | '111 Patent Claim 4 | Lin '400 |
|---|---|---|
| | | Copeland at [0069] ("It will now be appreciated that the disclosed meth-odology of intrusion detection is accomplished at least in part by analyzing communication flows to determine if such communications have the flow characteristics of probes or attacks. By analyzing communications for abnormal flow characteristics, attacks can be determined without the need for resource-intensive packet data analysis. A flow can be determined from the packets 101 that are transmitted between two hosts utilizing a single service. The addresses and port numbers of communications are easily discerned by analysis of the header information in a datagram.")<br><br>Copeland at [0087] ("As previously stated, the flow-based engine 155 does not analyze the data segments of packets for signature identification. Instead, the engine 155 associates all packets with a flow. It analyzes certain statistical data and assigns a concern index value to abnormal activity. The engine 155 builds a concern index for suspicious hosts by detecting suspicious activities on the network. An alarm is generated when those hosts build enough concern (in the form of a cumulated CI value) to cross the network administrator's predetermined threshold.")<br><br>Copeland at [0097] ("The described TCP session 300 of FIG. 3 is a generic TCP session in which a network might engage. In accordance with the invention, flow data is collected about the session to help determine if the communication is abnormal. In the preferred embodiment, information such as the total number of packets sent, the total amount of data sent, the session start time and duration, and the TCP flags set in all of the packets, are collected, stored in the database 160, and analyzed to determine if the communication was suspicious. If a communication is deemed suspicious, i.e. it meets predetermined criteria, a predetermined concern index value associated with a determined category of suspicious activity is added to the cumulated CI value associated with the host that made the communication.")<br><br>Copeland at [0111] ("As shown, the packets exchanged between two hosts associated with a single service can determine a flow. A port number designates a service application that is associated with the particular port. Communications utiliz-ing differing protocols or services provide differing flow characteristics. Consequently, the flow engine 155 analyzes each of the services separately.") |

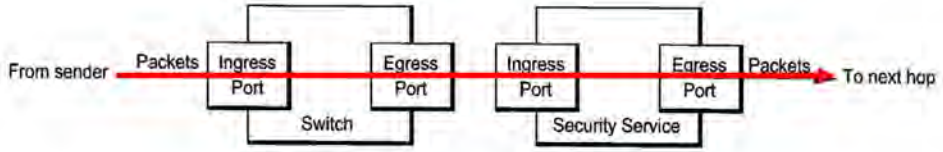
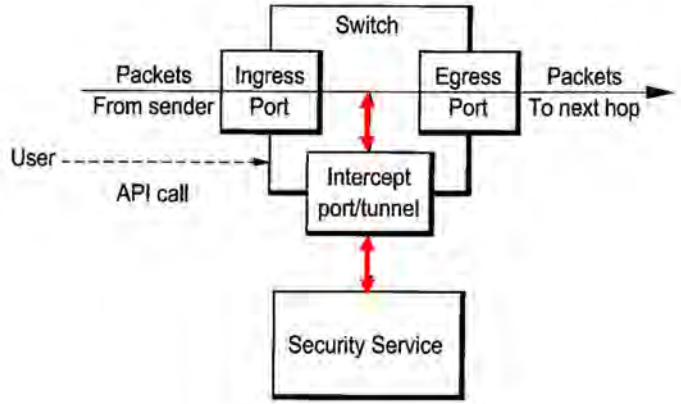


~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 601 of 1100

| No. | '111 Patent Claim 4 | Lin '400 |
|---|---|---|
| | | Copeland at [0150] ("A preferred hardware configuration 800 of an embodiment that executes the functions of the above-described flow-based engine is described in reference to FIG. 8. FIG. 8 illustrates a typically hardware configuration 800 for a network intrusion detection system. A monitoring appliance 150 serves as a pass-by filter of network traffic. A network device 135, such as a router or switch supporting sFlow provides the location for connecting the monitoring appliance 150 to the network 899 for monitoring the network traffic.") |
| 4[d] | sending the packet, by the controller, to the network node; and | Lin '400 discloses sending the packet, by the controller, to the network node.<br><br>For example, Lin '400 discloses returning the packet to the switch after analysis. A person of ordinary skill in the art would understand that the security service could be located in the SDN controller. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Lin '400 is found to not meet this limitation, sending the packet, by the controller, to the network node would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>Lin '400 3:11-24 ("Network security vendors provide network security services, such as firewall or deep packet inspection (DPI). Generally speaking, to provide network security services, packets of network traffic are intercepted for inspection. One way of intercepting network traffic is to place the security service in the middle of the packet forwarding path. This is illustrated in FIG. 3, where packets from a sender component (e.g., a sender computer) are received in an ingress port of a switch, forwarded to an egress port of the switch, and forwarded to the ingress port of a security component, such as a security service. The security service may inspect the packets, and forward the packets to an egress port of the switch toward the next hop, which may be another switch or a destination component (e.g., destination computer), for example.").<br><br>Lin '400 3:25-33 ("Another way of intercepting network traffic is to mirror the packets to be inspected on a switch that provides vendor specific mirroring application programming interface (API) as shown in FIG. 4. A user may make an API call such that particular packets that enter the ingress port of the switch are redirected or mirrored to the security service by way of a connection tunnel or a mirror port. The security service may forward the redirected or mirrored packets back to an egress port of the switch after inspection."). |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 602 of 1100

| No. | '111 Patent Claim 4 | Lin '400 |
|---|---|---|
| | |  <br>FIG. 3<br>Fig. 3 (annotation added)<br>  <br>FIG. 4<br>Fig. 4 (annotation added)<br><br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Lin '400 in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 603 of 1100

| No. | '111 Patent Claim 4 | Lin '400 |
|---|---|---|
| | | references identified in element 4(d) of Exhibit E-4 renders the claim, including the present limitation, obvious. Below is an example.<br><br>For example, Swenson discloses sending the packet, for example a video or image, back to the steering device after the network controller analyzes the packet and updates flow statistics.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0028] ("In contrast to conventional inline TCP throughput monitoring devices that monitor every single data packets transmitted and received, the network controller 140 is an "out-of-band" computer server that interfaces with the steer-ing device 130 to selectively inspect user flows of interest. The network controller 140 may further identify user flows (e.g., among the flows of interest) for optimization. In one embodiment, the network controller 140 may be imple-mented at the steering device 130 to monitor traffic. In other embodiments, the network controller 140 is coupled to and communicates with the steering device 130 for traffic moni-toring and optimization. When queried by the steering device 130, the network controller 140 determines if a given network flow should be ignored, monitored further or optimized. Opti-mization of a flow is often decided at the beginning of the flow because it is rarely possible to switch to optimized content mid-stream once non-optimized content delivery has begun. However, the network controller 140 may determine that existing flows associated with a particular subscriber or other entity should be optimized. In turn, new flows ( e.g., resulting from seek requests in media, new media requests, resume after pause, etc.) determined to be associated with the entity may be optimized. The network controller 140 uses the net-work state as well as historical traffic data in its decision for monitoring and |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 604 of 1100

| No. | '111 Patent Claim 4 | Lin '400 |
|---|---|---|
| | | optimization. Knowledge on the current net-work state, such as congestion, deems critical when it comes to data optimization.")<br><br>Swenson at [0029] ("As a flow is sent to the network controller 140 for inspection, historical network traffic data stored at the net-work controller 140 may be searched. The historical network traffic data includes information such as subscriber informa-tion, the cell towers to which the user devices attached, rout-ers through which the traffic is passing, geography regions, the backhaul segments, and time-of-day of the flows. For example, in a mobile network, the cell tower to which a user device is attached can be most useful, since it is the location where most congestion occurs due to limited bandwidth and high cost of the radio access network infrastructure. The network controller 140 looks into the historical traffic data for the average of the bandwidth per user at the particular cell tower. The network controller 140 can then estimate the amount ofbandwidth or degree of congestion for the new flow based on the historical record.")<br><br>Swenson at [0057] ("The Internet content adaption protocol is a light-weight protocol aimed at executing a simple remote proce-dure call on HTTP messages. ICAP leverages edge-based devices to help deliver value-added services using transparent HTTP proxy caches. Content adaptation refers to performing the particular value added service, such as content manipula-tion or other processing, for the associated HTTP client request/response. ICAP clients pass HTTP messages to ICAP servers for transformation or other processing. In tum, the ICAP server executes its transformation service on the HTTP messages and sends back responses to the ICAP client. At the core of this process is a cache that can proxy all client trans-actions and process them through ICAP servers, which may focus on specific functions, such as ad insertion, virus scan-ning, content translation, language translation, or content fil-tering. ICAP servers, such as those utilized by the network controller 140, handle these tasks to off-load value-added services from network devices including an ICAP client, such as the steering device 130. By offloading value added services from the steering device 130, processing infrastructure (e.g., optimization services and network controllers) may be scaled independent from the steering devices handling raw HTTP throughput.")<br><br>Swenson at [0059] ("In one embodiment, as the steering device 130 monitors network responses, it is looking for flows that match one or more signatures for video and images. When a match-ing flow is detected, the steering device 130 forwards the HTTP request and a |

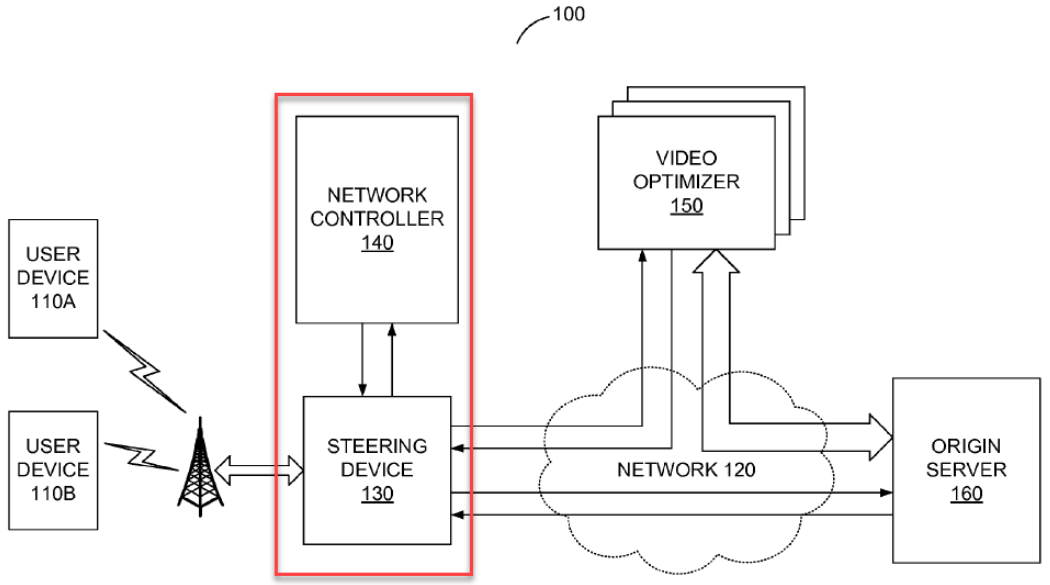


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 605 of 1100

| No. | '111 Patent Claim 4 | Lin '400 |
|---|---|---|
| | | portion of the HTTP response to the network controller 140 over the ICAP client interface 404. After receiving the request and the portion of response at the ICAP server interface 406, the flow analyzer 312 of the net-work controller 140 performs a deep flow inspection to deter-mine if the flow is worth bandwidth monitoring and/or user detection. For example, the flow inspection performed by the flow analyzer 312 may determine if the flow indeed contains large or medium object ( e.g., larger than 50 kB), and/or if the source IP address of the flow is from a user or a group of users that are required to be monitored by policies. The flow ana-lyzer 312 may also determine if the flow needs to be opti-mized based on historical flow statistical data.")<br><br>Swenson at [0060] ("If the flow is deemed of interest, the steering device 130 is notified to steer the flow through the network controller 140. This is known as the "continue" working mode for bandwidth monitoring. In the "continue" mode, the network controller 140 interfaces with the steering device 130 to func-tion, on-demand, as a traditional inline network element for flows deemed of interest. Thus, the network controller 140 ingests the network flow for inspection and subsequently forwards the network flow on the network response path. For example, for this particular flow, the origin server 160 responds to the user request by sending video or images over the network link 413 to the steering device 130, which for-wards the video or images to the network controller 140 over a network link 414. After the network controller 140 updates the flow statistics, the video or images are returned to the steering device 130 over a network link 415, which transmits the video or images to the user device 110 over the network link 416.")<br><br>Swenson at [0071] ("After receiving the request, the video optimizer 150 forwards the video HTTP GET requests 622 to the origin server 160 and in return, receives a video file 624 from the origin server 160. The video optimizer 150 transcodes the video file to a format usable by the client device 110 based on network bandwidth available to the user device 110. The optimized video 626 is then transmitted from the video opti-mizer 150 to the steering device 130. In one embodiment, the steering device 130 intercepts the optimized video 626. The steering device 130 will then send an ICAP request to the network controller 140 for inspection. The network controller 140 deems this flow to be monitored and sends ICAP response 630. The steering device 130 then allows the flow to go through to the user device 110. The steering device 130 next sends periodic ICAP "counting" updates 632 to the network |

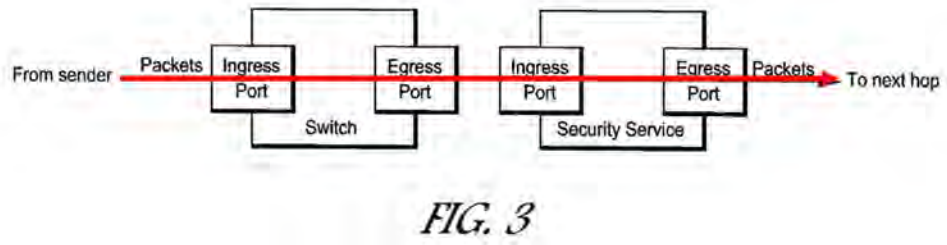


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 606 of 1100

| No. | '111 Patent Claim 4 | Lin '400 |
|---|---|---|
| | | controller 140 until the flow completes. As such, the client receives the optimized video 626 for substantially real-time playback on an application executing on the user device 110.")<br><br>Swenson at Figure 1 (annotation added)<br><br>  <br><br>FIG. 1 |
| 4[e] | responsive to receiving the packet, sending the packet, by the network node, to the second entity. | Lin '400 discloses responsive to receiving the packet, sending the packet, by the network node, to the second entity.<br><br>For example, Lin '400 discloses that the switch sends the packet to its destination upon receiving the returned packet after inspection by the security service. |

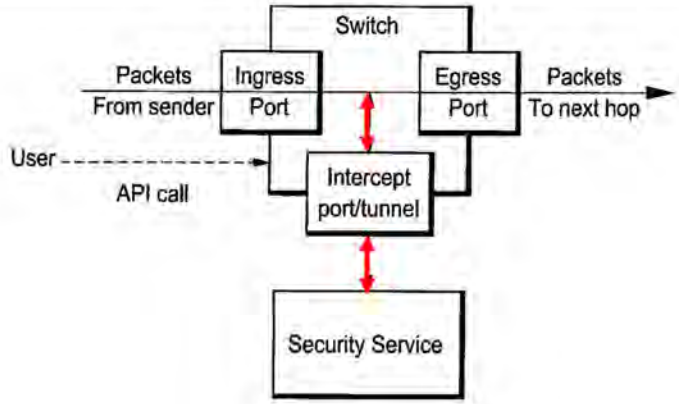


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 607 of 1100

| No. | '111 Patent Claim 4 | Lin '400 |
|---|---|---|
| | | Lin '400 3:11-24 ("Network security vendors provide network security services, such as firewall or deep packet inspection (DPI). Generally speaking, to provide network security services, packets of network traffic are intercepted for inspection. One way of intercepting network traffic is to place the security service in the middle of the packet forwarding path. This is illustrated in FIG. 3, where packets from a sender component (e.g., a sender computer) are received in an ingress port of a switch, forwarded to an egress port of the switch, and forwarded to the ingress port of a security component, such as a security service. The security service may inspect the packets, and forward the packets to an egress port of the switch toward the next hop, which may be another switch or a destination component (e.g., destination computer), for example.").<br><br>Lin '400 3:25-33 ("Another way of intercepting network traffic is to mirror the packets to be inspected on a switch that provides vendor specific mirroring application programming interface (API) as shown in FIG. 4. A user may make an API call such that particular packets that enter the ingress port of the switch are redirected or mirrored to the security service by way of a connection tunnel or a mirror port. The security service may forward the redirected or mirrored packets back to an egress port of the switch after inspection.").<br><br>  <br>FIG. 3<br><br>Fig. 3 (annotation added) |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 608 of 1100

| No. | '111 Patent Claim 4 | Lin '400 |
|---|---|---|
| | |  <br>*FIG. 4*<br>Fig. 4 (annotation added) |

| No. | '111 Patent Claim 5 | Lin '400 |
|---|---|---|
| 5 | The method according to claim 1, further comprising responsive to the packet satisfying the criterion and to the instruction, sending the packet or a portion thereof, by the network node, to the controller. | Lin '400 discloses the method according to claim 1, further comprising responsive to the packet satisfying the criterion and to the instruction, sending the packet or a portion thereof, by the network node, to the controller.<br><br>For example, Lin '400 discloses a command to send a packet to a security service for further inspection upon positive identification. A person of ordinary skill in the art would understand that the security service could be located in the SDN controller. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Lin '400 is found to not meet this limitation, responsive to the packet satisfying the criterion and to the instruction, sending the packet or a portion thereof, by the network node, to the controller would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>Lin '400 6:40-63 ("After the redirect flow rules for creating the SDN pipe are inserted in the flow tables 621, any packet received by the SDN switch 620 in the ingress port 623-1 will be |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 609 of 1100

| No. | '111 Patent Claim 5 | Lin '400 |
|---|---|---|
| | | identified as to be forwarded to the redirect port 623-2, and any packet received by the SDN switch 620 in the redirect port 623-2 will be identified as to be forwarded to the ingress port 623-1 (see arrow 602). This allows the security service 630 to receive from the redirect port 623-2 all outgoing packets sent by the sender component 622 to the ingress port 623-1. The security service 630 may inspect the outgoing packets for compliance with security policies. The security service 630 may drop, or perform other security response, to packets that do not pass inspection (e.g., packets that do not meet firewall policies, packets containing prohibited payload, packets with malicious content, etc.). The security service 630 may forward those packets that pass inspection toward their destination by re-injecting the packets back into the SDN switch 620 by way of the re-inject port 623-3. Once back in the SDN switch 620 by way of the re-inject port 623-3, the flow rules that govern packets received in the ingress port 623-1 and the redirect port 623-2 no longer apply. Accordingly, the re-injected packets are forwarded to the egress port 623-4 (or some other port) toward the next hop in accordance with the L2 switching logic of the SDN computer network 600.").<br><br>Lin '400 5:45-55 ("The security service 630 may inspect packets for compliance/non-compliance with security policies, such as for presence of malicious code, compliance with firewall rules and access control lists, network intrusion detection, and other computer network security services. The security service 630 may employ conventional packet inspection algorithms. The security service 630 may comprise the Trend Micro Deep Security™ service, for example. The security service 630 may also comprise a physical machine, e.g., a server computer, an appliance, a gateway computer, etc.").<br><br>Lin '400 7:10-22 ("Re-injecting packets that pass inspection consume bandwidth, as the packets will have to be transmitted by the security service 630 to the re-inject port 623-3. For optimization, the SDN switch 620 may be configured to copy packets that are redirected to the security service 630 for inspection. This way, the security service 630 simply has to inform the SDN switch 620 an action to take on packets based on the result of the inspection (see arrow 604). For example, the security service 630 may send an index identifying the packets and an action on how to manipulate the packets. The action may instruct the SDN switch 620 to drop the copied packets, forward the copied packets to their destinations, quarantine the copied packets, etc."). |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 610 of 1100

| No. | '111 Patent Claim 5 | Lin '400 |
|---|---|---|
| | | Lin '400 8:33-45 ("The security service 630 receives the outgoing packets from the redirect port 623-2 (see arrow 654) and inspects the outgoing packets. After inspection, the security service 630 re-injects the outgoing packets (e.g., outgoing packets that passed inspection) back into the SDN switch 620 by way of the re-inject port 623-3 (see arrow 655). The SDN switch 620 receives the outgoing packets on the re-inject port 623-3. The SDN switch 620 forwards the outgoing packets from the re-inject port 623-3 to the egress port 623-4 in accordance with the L2 switching logic of the SDN computer network 600 (see arrow 657). The outgoing packets exit the SDN switch 620 through the egress port 623-4 (see arrow 658) and move towards their destination."). <br><br> Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Lin '400 in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 5 of Exhibit E-4 renders the claim, including the present limitation, obvious. Below is an example. <br><br> For example, Copeland discloses sending packets and sampled packet headers to the intrusion detection engine on the monitoring appliance based on matching predetermined values associated with a concern index. <br><br> Copeland at [0067] ("The host servers 130 are directly or indirectly coupled to one or more network devices 135 such as routers or switches that support providing a sampled data stream such as that provided by sFlow. In a typical preferred configuration for the present invention, a monitoring appli-ance 150 operating a flow-based intrusion detection engine 155 is receiving sampled packet headers from one or more network devices 135. The monitoring appliance 150 moni-tors the communications between the host server 130 and other hosts 120, 110 in the attempt to detect intrusion activity.") <br><br> Copeland [0079] ("Large packets tend to be fragmented by networks that cannot handle a large packet size. A 16-bit packet identification is used to reassemble fragmented packets. Three one-bit set of fragmentation flags control whether a packet is or may be fragmented. The 13-bit fragment offset is a sequence number for the 4-byte words in the packet when reassembled. In a series of fragments, the first offset will be zero.") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 611 of 1100

| No. | '111 Patent Claim 5 | Lin '400 |
|---|---|---|
| | | Copeland at [0097] ("The described TCP session 300 of FIG. 3 is a generic TCP session in which a network might engage. In accordance with the invention, flow data is collected about the session to help determine if the communication is abnormal. In the preferred embodiment, information such as the total number of packets sent, the total amount of data sent, the session start time and duration, and the TCP flags set in all of the packets, are collected, stored in the database 160, and analyzed to determine if the communication was suspicious. If a communication is deemed suspicious, i.e. it meets predetermined criteria, a predetermined concern index value associated with a determined category of suspicious activity is added to the cumulated CI value associated with the host that made the communication.")<br><br>Copeland at [0120] ("The sampled packet headers sent from the sFlow agent are captured and processed by the sample packet collector 505 in order to create a "Packet Data" data struc-ture that includes the sFlow agent source of the packets, the header of the sampled packets, and other information avail-able from the sFlow data stream that may be important. For example, one data field that is optionally available pr vides the username of the user using the computer at the time of the communications. This information is extremely useful in some environments subject to regulatory requirements and monitoring of the communications on the network. In this case the username will be stored as "supplementary infor-mation" for auditing purposes in the flow data. Other infor-mation, including the sampling device and the physical port on which the communications was detected may also be retained for other uses such as mitigation, where a host may be removed from the network.")<br><br>Copeland at [0126]-[0129] ("If a particular packet 101 being processed by the packet classifier 510 matches a particular entry or record in the flow data structure 162, data from that particular packet 101 is used to update the statistics in the corresponding flow data structure record. A packet 101 is considered to match to a flow data structure record if both IP numbers match and the source of the sampled packet matches and:<br><br>(1) both port numbers match and no port is marked as the "server" port, or<br>(2) the port number previously marked as the "server" port matches, or<br>(3) one of the port numbers matches, but the other does not, and the neither port number has been marked as the server port (in this case the matching port number is marked as the "server" port).") |

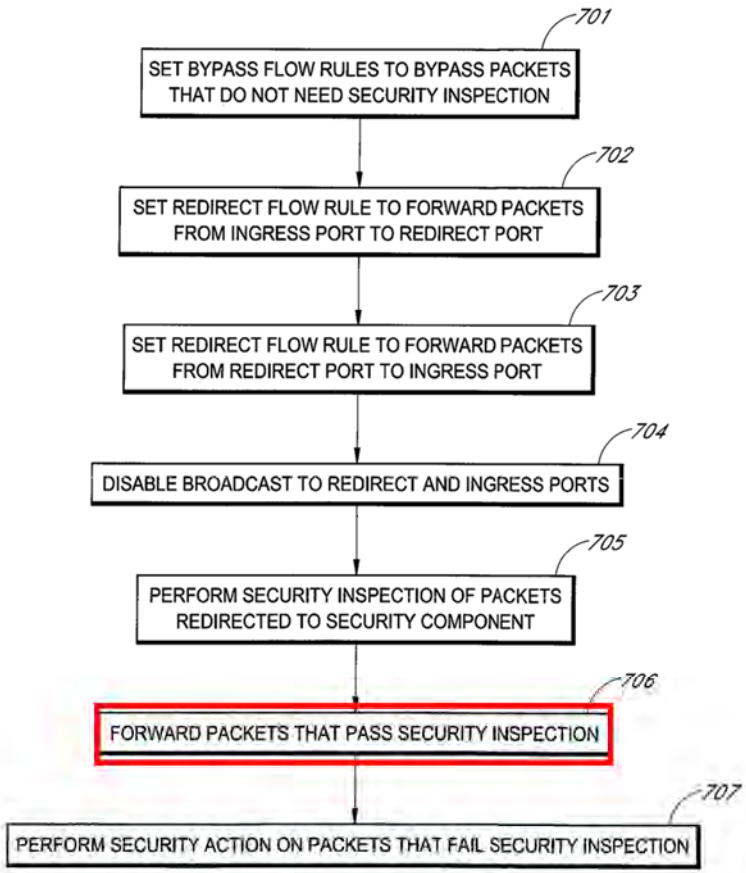


~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 612 of 1100

| No. | '111 Patent Claim 5 | Lin '400 |
|---|---|---|
| | | Copeland at [0144] ("Concern index (CI) values calculated from packet anomalies also add to a host's accumulated concern index value. Table II of FIG. 7 shows one scheme for assigning concern index values due to other events revealed by the flow analysis. For example, there are many combinations of TCP flag bits that are rarely or never seen in valid TCP connections. When the packet classifier thread 510 recog-nizes one of these combinations, it directly adds a predeter-mined value to the sending host's accumulated concern index value. When the packet classifier thread 510 searches along the flow linked-list (i.e. flow data 162) for a match to the current packet 101, it keeps count of the number of flows active with matching IP addresses but no matching port number. If this number exceeds a predetermined threshold value (e.g., 4) and is greater than the previous number noticed, CI is added for an amount corresponding to a "port scan." A bit in the host record is set to indicate that the host has received CI for "port scanning."")<br><br>Copeland at [0150] ("A preferred hardware configuration 800 of an embodiment that executes the functions of the above-described flow-based engine is described in reference to FIG. 8. FIG. 8 illustrates a typically hardware configuration 800 for a network intrusion detection system. A monitoring appliance 150 serves as a pass-by filter of network traffic. A network device 135, such as a router or switch supporting sFlow provides the location for connecting the monitoring appliance 150 to the network 899 for monitoring the network traffic.")<br><br>Copeland at [0159]-[0162] ("A packet 101 is considered to match to a flow data structure record if both IP numbers match and the source of the sampled data matches and:<br><br>(a). both port numbers match and no port is marked as the "server" port, or<br>(b). the port number previously marked as the "server" port matches, or<br>(c). one of the port numbers matches, but the other does not, and the neither port number has been marked as the server port (in this case the matching port number is marked as the "server" port).") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 613 of 1100

| No. | '111 Patent Claim 5 | Lin '400 |
|---|---|---|
| | |  <br>*FIG. 9*<br><br>Figure 9 (annotation added)<br><br>Lin '400 4:53-67 ("The SDN switch 620 may comprise one or more flow tables 621. The flow tables 621 may comprise one or more flow rules (labeled as 624) that indicate how to manipulate or process packets that are passing through the SDN switch 620. As a particular |

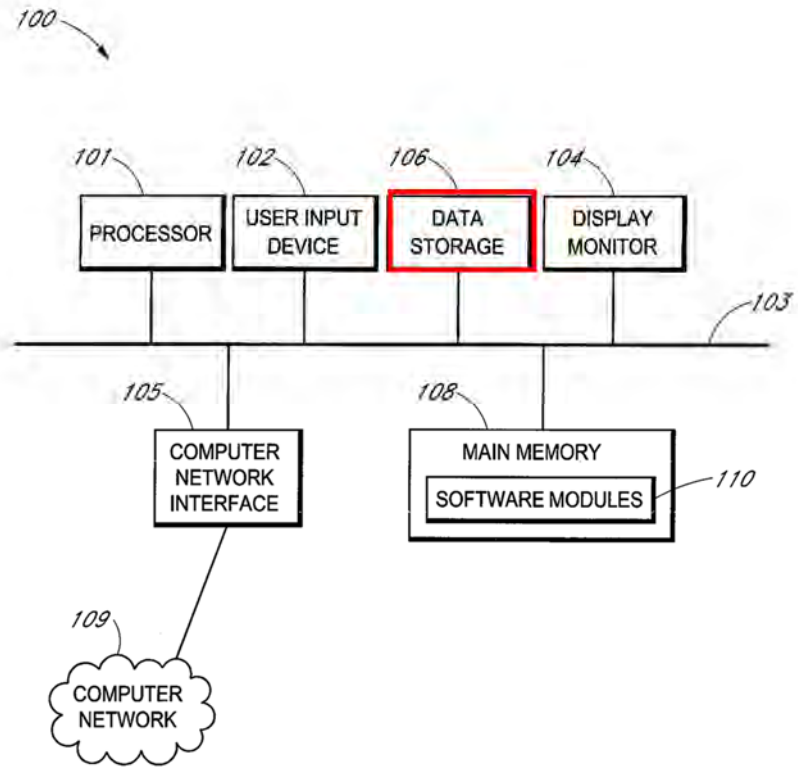


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 614 of 1100

| No. | '111 Patent Claim 5 | Lin '400 |
|---|---|---|
| | | example, a flow rule may indicate that a packet received in the ingress port 623-1 is to be forwarded to the redirect port 623-2. Another flow rule may indicate that a packet received in the redirect port 623-2 is to be forwarded to the ingress port 623-1. The just mentioned pair of flow rules are redirect flow rules that create an SDN pipe between the sender component 622 and the security service 630, allowing the security service 630 to inspect packets sent by or going to the sender component 622. Table 1 shows an example flow table with flow rules that create an SDN pipe between the security service 630 and the sender component 622.").

| No. | '111 Patent Claim 6 | Lin '400 |
|---|---|---|
| 6 | The method according to claim 5, further comprising storing the received packet or a portion thereof, by the controller, in a memory. | Lin '400 discloses the method according to claim 5, further comprising storing the received packet or a portion thereof, by the controller, in a memory.<br><br>For example, Lin '400 discloses a main memory and a data storage device that could be used to store the packet or a portion thereof.<br><br>Lin '400 2:47-65 ("FIG. 2 shows a schematic diagram of a computer system 100 that may be employed with embodiments of the present invention. The computer system 100 may be employed as a control plane and/or a data plane, for example. As another example, the computer system 100 may be employed to host a virtualization environment that supports a plurality of virtual machines. The computer system 100 may have fewer or more components to meet the needs of a particular application. The computer system 100 may include one or more processors 101. The computer system 100 may have one or more buses 103 coupling its various components. The computer system 100 may include one or more user input devices 102 (e.g., keyboard, mouse), one or more data storage devices 106 (e.g., hard drive, optical disk, Universal Serial Bus memory), a display monitor 104 (e.g., liquid crystal display, flat panel monitor), a computer network interface 105 (e.g., network adapter, modem), and a main memory 108 (e.g., random access memory). The computer network interface 105 may be coupled to a computer network 109.")<br><br>Lin '400 2:67-3:10 ("The computer system 100 is a particular machine as programmed with software modules 110. The software modules 110 comprise computer-readable program code stored non-transitory in the main memory 108 for execution by the processor 101. The |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 615 of 1100

| No. | '111 Patent Claim 6 | Lin '400 |
|---|---|---|
| | | computer system 100 may be configured to perform its functions by executing the software modules 110. The software modules 110 may be loaded from the data storage device 106 to the main memory 108. An article of manufacture may be embodied as computer-readable storage medium including instructions that when executed by a computer causes the computer to be operable to perform the functions of the software modules 110.")<br>  <br><br>FIG. 2<br>Fig. 2 (annotation added) |

| No. | '111 Patent Claim 7 | Lin '400 |
|---|---|---|
| 7 | The method according to claim 5, further comprising responsive | Lin '400 discloses the method according to claim 5, further comprising responsive to the packet satisfying the criterion and to the instruction, sending a portion of the packet, by the network node, to the controller. |

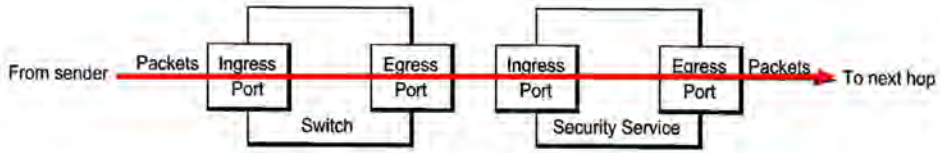
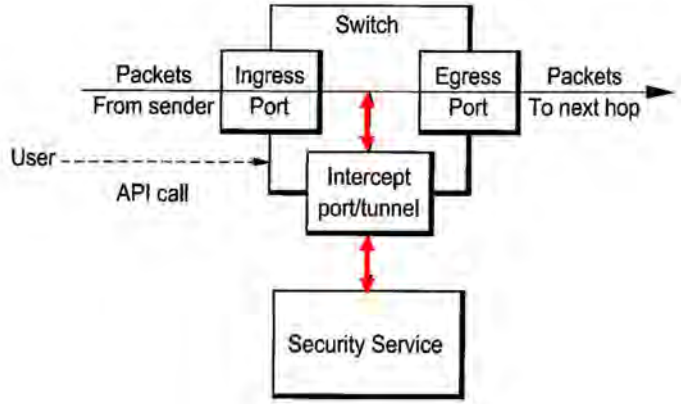


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 616 of 1100

| No. | '111 Patent Claim 7 | Lin '400 |
|---|---|---|
| | to the packet satisfying the criterion and to the instruction, sending a portion of the packet, by the network node, to the controller. | For example, Lin '400 discloses a command to send a packet to a security service for further inspection upon positive identification A person of ordinary skill in the art would understand that the security service could be located in the SDN controller. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Lin '400 is found to not meet this limitation, responsive to the packet satisfying the criterion and to instruction, sending a portion of the packet, by the network node, to the controller would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>*See supra* at Claim 5.<br><br>Lin '400 3:11-24 ("Network security vendors provide network security services, such as firewall or deep packet inspection (DPI). Generally speaking, to provide network security services, packets of network traffic are intercepted for inspection. One way of intercepting network traffic is to place the security service in the middle of the packet forwarding path. This is illustrated in FIG. 3, where packets from a sender component (e.g., a sender computer) are received in an ingress port of a switch, forwarded to an egress port of the switch, and forwarded to the ingress port of a security component, such as a security service. The security service may inspect the packets, and forward the packets to an egress port of the switch toward the next hop, which may be another switch or a destination component (e.g., destination computer), for example.").<br><br>Lin '400 3:25-33 ("Another way of intercepting network traffic is to mirror the packets to be inspected on a switch that provides vendor specific mirroring application programming interface (API) as shown in FIG. 4. A user may make an API call such that particular packets that enter the ingress port of the switch are redirected or mirrored to the security service by way of a connection tunnel or a mirror port. The security service may forward the redirected or mirrored packets back to an egress port of the switch after inspection."). |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 617 of 1100

| No. | '111 Patent Claim 7 | Lin '400 |
|---|---|---|
| | |  <br>*FIG. 3*<br>Fig. 3 (annotation added)<br>  <br>*FIG. 4*<br>Fig. 4 (annotation added)<br><br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Lin '400 in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 5 of Exhibit E-4 renders the claim, including the present limitation, obvious. Below is an example. |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 618 of 1100

| No. | '111 Patent Claim 7 | Lin '400 |
|---|---|---|
| | | For example, Copeland discloses sending packets and sampled packet headers to the intrusion detection engine on the monitoring appliance based on matching predetermined values associated with a concern index.<br><br>Copeland at [0067] ("The host servers 130 are directly or indirectly coupled to one or more network devices 135 such as routers or switches that support providing a sampled data stream such as that provided by sFlow. In a typical preferred configuration for the present invention, a monitoring appli-ance 150 operating a flow-based intrusion detection engine 155 is receiving sampled packet headers from one or more network devices 135. The monitoring appliance 150 moni-tors the communications between the host server 130 and other hosts 120, 110 in the attempt to detect intrusion activity.")<br><br>Copeland [0079] ("Large packets tend to be fragmented by networks that cannot handle a large packet size. A 16-bit packet identification is used to reassemble fragmented packets. Three one-bit set of fragmentation flags control whether a packet is or may be fragmented. The 13-bit fragment offset is a sequence number for the 4-byte words in the packet when reassembled. In a series of fragments, the first offset will be zero.")<br><br>Copeland at [0097] ("The described TCP session 300 of FIG. 3 is a generic TCP session in which a network might engage. In accordance with the invention, flow data is collected about the session to help determine if the communication is abnormal. In the preferred embodiment, information such as the total number of packets sent, the total amount of data sent, the session start time and duration, and the TCP flags set in all of the packets, are collected, stored in the database 160, and analyzed to determine if the communication was suspicious. If a communication is deemed suspicious, i.e. it meets predetermined criteria, a predetermined concern index value associated with a determined category of suspicious activity is added to the cumulated CI value associated with the host that made the communication.")<br><br>Copeland at [0120] ("The sampled packet headers sent from the sFlow agent are captured and processed by the sample packet collector 505 in order to create a "Packet Data" data struc-ture that includes the sFlow agent source of the packets, the header of the sampled packets, and other information avail-able from the sFlow data stream that may be important. For |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 619 of 1100

| No. | ’111 Patent Claim 7 | Lin ’400 |
|---|---|---|
| | | example, one data field that is optionally available pr vides the username of the user using the computer at the time of the communications. This information is extremely useful in some environments subject to regulatory requirements and monitoring of the communications on the network. In this case the username will be stored as "supplementary infor-mation" for auditing purposes in the flow data. Other infor-mation, including the sampling device and the physical port on which the communications was detected may also be retained for other uses such as mitigation, where a host may be removed from the network.”)<br><br>Copeland at [0126]-[0129] (“If a particular packet 101 being processed by the packet classifier 510 matches a particular entry or record in the flow data structure 162, data from that particular packet 101 is used to update the statistics in the corresponding flow data structure record. A packet 101 is considered to match to a flow data structure record if both IP numbers match and the source of the sampled packet matches and:<br><br>(1) both port numbers match and no port is marked as the "server" port, or<br>(2) the port number previously marked as the "server" port matches, or<br>(3) one of the port numbers matches, but the other does not, and the neither port number has been marked as the server port (in this case the matching port number is marked as the "server" port).”)<br><br>Copeland at [0144] (“Concern index (CI) values calculated from packet anomalies also add to a host's accumulated concern index value. Table II of FIG. 7 shows one scheme for assigning concern index values due to other events revealed by the flow analysis. For example, there are many combinations of TCP flag bits that are rarely or never seen in valid TCP connections. When the packet classifier thread 510 recog-nizes one of these combinations, it directly adds a predeter-mined value to the sending host's accumulated concern index value. When the packet classifier thread 510 searches along the flow linked-list (i.e. flow data 162) for a match to the current packet 101, it keeps count of the number of flows active with matching IP addresses but no matching port number. If this number exceeds a predetermined threshold value (e.g., 4) and is greater than the previous number noticed, CI is added for an amount corresponding to a "port scan." A bit in the host record is set to indicate that the host has received CI for "port scanning."”) |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 620 of 1100

| No. | '111 Patent Claim 7 | Lin '400 |
|---|---|---|
| | | Copeland at [0150] ("A preferred hardware configuration 800 of an embodiment that executes the functions of the above-described flow-based engine is described in reference to FIG. 8. FIG. 8 illustrates a typically hardware configuration 800 for a network intrusion detection system. A monitoring appliance 150 serves as a pass-by filter of network traffic. A network device 135, such as a router or switch supporting sFlow provides the location for connecting the monitoring appliance 150 to the network 899 for monitoring the network traffic.")<br><br>Copeland at [0159]-[0162] ("A packet 101 is considered to match to a flow data structure record if both IP numbers match and the source of the sampled data matches and:<br><br>(a). both port numbers match and no port is marked as the "server" port, or<br>(b). the port number previously marked as the "server" port matches, or<br>(c). one of the port numbers matches, but the other does not, and the neither port number has been marked as the server port (in this case the matching port number is marked as the "server" port).") |

| No. | '111 Patent Claim 8 | Lin '400 |
|---|---|---|
| 8[a] | The method according to claim 7, wherein the portion of the packet consists of multiple consecutive bytes, and | Lin '400 discloses the method according to claim 7, wherein the portion of the packet consists of multiple consecutive bytes.<br><br>For example, Lin '400 discloses particular packets with specific byte counts. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Lin '400 is found to not meet this limitation, wherein the portion of the packet consists of multiple consecutive bytes would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>Lin '400 5:8-25 ("A flow table may include columns that indicate one or more conditions, a column that indicates an action to take when the conditions are met, and a column for statistics. A row on the flow table may comprise a flow rule. In the example of Table 1, the "Action" column indicates an action to take when conditions are met, and the "Count" column indicates statistics, such as byte count. The rest of the columns of Table 1 indicate conditions. For example, "IN_PORT", "MAC src" (media access control (MAC) address of the source |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 621 of 1100

| No. | '111 Patent Claim 8 | Lin '400 |
|---|---|---|
| | | of the packet), "MAC dst" (MAC address of the destination of the packet), "IP src" (Internet Protocol (IP) address of the source of the packet), "IP dst" (IP address of the destination of the packet), etc. are conditions that identify a particular packet. When the conditions are met, i.e., the particular packet is identified, the action indicated in the corresponding "Action" column is performed on the packet. The asterisks in Table 1 indicate an irrelevant condition."). <br><br> Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Lin '400 in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 8(a) of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references. <br><br> For example, Kempf discloses consecutive bytes of a packet header field. <br><br> Kempf at [0081] ("In one embodiment, OpenFlow is modified to pro-vide rules for GTP TEID Routing. FIG. 17 is a diagram of one embodiment of the OpenFlow flow table modification for GTP TEID routing. An OpenFlow switch that supports TEID routing matches on the 2 byte (16 bit) collection of header fields and the 4 byte (32 bit) GTP TEID, in addition to other OpenFlow header fields, in at least one flow table ( e.g., the first flow table). The GTP TEID flag can be wildcarded (i.e. matches are "don't care"). In one embodiment, the EPC pro-tocols do not assign any meaning to TEIDs other than as an endpoint identifier for tunnels, like ports in standard UDP/ TCP transport protocols. In other embodiments, the TEIDs can have a correlated meaning or semantics. The GTP header flags field can also be wildcarded, this can be partially matched by combining the following bitmasks: 0xFF00- Match the Message Type field; 0xe0-Match the Version field; 0xl0-Match the PT field; 0x04-Match the E field; 0x02-Match the S field; and 0x0l-Match the PN field.") <br><br> Kempf at [0083] ("If a packet either needs encapsulation or arrives encapsulated with nonzero header flags, header extensions, and/or the GTP-U packet is not a G-PDU packet (i.e. it is a GTP-U control packet), the processing must proceed via the gateway's slow path (software) control plane. GTP-C and GTP' packets directed to the gateway's IP address are a result of mis-configuration and are in error. They must be sent to the OpenFlow controller, since these packets are handled by the S-GW-C and P-GW-C control plane entities in the cloud. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 622 of 1100

| No. | '111 Patent Claim 8 | Lin '400 |
|---|---|---|
| | | computing system or to the billing entity handling GTP' and not the S-GW-D and P-GW-D data plane switches.")<br><br>Kempf at [0087] ("In one embodiment, slow path support for GTP is implemented with an OpenFlow gateway switch. An Open-Flow mobile gateway switch also contains support on the software control plane for slow path packet processing. This path is taken by G-PDU (message type 255) packets with nonzero header fields or extension headers, and user data plane packets requiring encapsulation with such fields or addition of extension headers, and by G TP-U control packets. For this purpose, the switch supports three local ports in the software control plane: LOCAL_GTP _CONTROL-the switch fast path forwards GTP encapsulated packets directed to the gateway IP address that contain GTP-U control mes-sages and the local switch software control plane initiates local control plane actions depending on the GTP-U control message; LOCAL_GTP _U_DECAP-the switch fast path forwards G-PDU packets to this port that have nonzero header fields or extension headers (i.e. E!=0, S!=0, or PN!=0). These packets require specialized handling. The local switch software slow path processes the packets and performs the specialized handling; and LOCAL_GTP _U_ENCAP-the switch fast path forwards user data plane packets to this port that require encapsulation in a GTP tunnel with nonzero header fields or extension headers (i.e. E!=0, S!=0, or PN!=0). These packets require specialized handling. The local switch software slow path encapsulates the packets and performs the specialized handling. In addition to forwarding the packet, the switch fast path makes the OpenFlow metadata field avail-able to the slow path software.")<br><br>Kempf at [0091] ("When a packet header matches a rule associated with the virtual port, the GTP TEID is written into the lower 32 bits of the metadata and the packet is directed to the virtual port. The virtual port calculates the hash of the TEID and looks up the tunnel header information in the tunnel header table. If no such tunnel information is present, the packet is forwarded to the controller with an error indication. Other-wise, the virtual port constructs a GTP tunnel header and encapsulates the packet. Any DSCP bits or VLAN priority bits are additionally set in the IP or MAC tunnel headers, and any VLAN tags or MPLS labels are pushed onto the packet. The encapsulated packet is forwarded out the physical port to which the virtual port is bound.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 623 of 1100

| No. | '111 Patent Claim 8 | Lin '400 |
|---|---|---|
| | | Kempf at [0092] ("In one embodiment, the system implements a GTP fast path decapsulation virtual port. When requested by the S-GW and P-GW control plane software running in the cloud computing system, the gateway switch installs rules and actions for routing GTP encapsulated packets out of GTP tunnels. The rules match the GTP header flags and the GTP TEID for the packet, in the modified OpenFlow flow table shown in FIG. 17 as follows: the IP destination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); the UDP destination port is the GTP-U destination port (2152); and the header fields and message type field is wildcarded with the flag 0XFFF0 and the upper two bytes of the field match the G-PDU message type (255) while the lower two bytes match 0x30, i.e. the packet is a GTP packet not a GTP' packet and the version number is 1.")<br><br>Kempf at [0098] ("The header flags and message type fields for the three rules are wildcarded with the following bitmasks and match as follows: bitmask 0xFFF4 and the upper two bytes match the G-PDU message type (255) while the lower two bytes are Ox34, indicating that the version number is 1, the packet is a GTP packet, and there is an extension header present; bitmask 0xFFFF2 and the upper two bytes match the G-PDU message type (255) while the lower two bytes are 0x32, indicating that the version number is 1, the packet is a GTP packet, and there is a sequence number bitmask 0xFF0l and the upper two bytes match the G-PDU message type (255) while the lower two bytes are 0x31, indicating that the version number is 1, the packet is a GTP packet, and a N-PDU is present.")<br><br>Kempf at [0101] ("In one embodiment, the system implements han-dling of user data plane packets requiring GTP-U encapsula-tion with extension headers, sequence numbers, and N-PDU numbers. User data plane packets that require extension head-ers, sequence numbers, or N-PDU numbers during GTP encapsulation require special handling by the software slow path. For these packets, the OpenFlow controller programs a rule matching the 4 tuple: IP source address; IP destination address; UDP/TCP/SCTP source port; and UDP/TCP/SCTP destination port. The instructions for matching packets are:<br><br>Write-Metadata ( GTP-TEID, 0x FFFFFFFF)<br>Apply-Actions (Set-Output-Port LOCAL_GTP _U_ENCAP )") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 624 of 1100

| No. | '111 Patent Claim 8 | Lin '400 |
|---|---|---|
| | | For example, Copeland discloses fragmenting packets into smaller byte sizes, including headers and flags. Copeland further discloses sending sampled packet headers, consisting of fragmented packets of consecutive bytes to the monitoring device.<br><br>Copeland [0079] ("Large packets tend to be fragmented by networks that cannot handle a large packet size. A 16-bit packet identification is used to reassemble fragmented packets. Three one-bit set of fragmentation flags control whether a packet is or may be fragmented. The 13-bit fragment offset is a sequence number for the 4-byte words in the packet when reassembled. In a series of fragments, the first offset will be zero.") |
| 8[b] | wherein the instruction comprises identification of the consecutive bytes in the packet. | Lin '400 discloses wherein the instruction comprises identification of the consecutive bytes in the packet.<br><br>For example, Lin '400 discloses a flow table with a flow rule that indicates particular packets and identifies byte counts. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Lin '400 is found to not meet this limitation, wherein the instruction comprises identification of the consecutive bytes in the packet would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>Lin '400 5:8-25 ("A flow table may include columns that indicate one or more conditions, a column that indicates an action to take when the conditions are met, and a column for statistics. A row on the flow table may comprise a flow rule. In the example of Table 1, the "Action" column indicates an action to take when conditions are met, and the "Count" column indicates statistics, such as byte count. The rest of the columns of Table 1 indicate conditions. For example, "IN_PORT", "MAC src" (media access control (MAC) address of the source of the packet), "MAC dst" (MAC address of the destination of the packet), "IP src" (Internet Protocol (IP) address of the source of the packet), "IP dst" (IP address of the destination of the packet), etc. are conditions that identify a particular packet. When the conditions are met, i.e., the particular packet is identified, the action indicated in the corresponding "Action" column is performed on the packet. The asterisks in Table 1 indicate an irrelevant condition."). |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 625 of 1100

| No. | '111 Patent Claim 8 | Lin '400 |
|---|---|---|
| | | Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Lin '400 in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 8(b) of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references.<br><br>For example, Kempf discloses rules in which the flow table includes matching to the consecutive bytes of a packet header.<br><br>Kempf at [0081] ("In one embodiment, OpenFlow is modified to pro-vide rules for GTP TEID Routing. FIG. 17 is a diagram of one embodiment of the OpenFlow flow table modification for GTP TEID routing. An OpenFlow switch that supports TEID routing matches on the 2 byte (16 bit) collection of header fields and the 4 byte (32 bit) GTP TEID, in addition to other OpenFlow header fields, in at least one flow table ( e.g., the first flow table). The GTP TEID flag can be wildcarded (i.e. matches are "don't care"). In one embodiment, the EPC pro-tocols do not assign any meaning to TEIDs other than as an endpoint identifier for tunnels, like ports in standard UDP/ TCP transport protocols. In other embodiments, the TEIDs can have a correlated meaning or semantics. The GTP header flags field can also be wildcarded, this can be partially matched by combining the following bitmasks: 0xFF00- Match the Message Type field; 0xe0-Match the Version field; 0xl0-Match the PT field; 0x04-Match the E field; 0x02-Match the S field; and 0x0l-Match the PN field.")<br><br>Kempf at [0083] ("If a packet either needs encapsulation or arrives encapsulated with nonzero header flags, header extensions, and/or the GTP-U packet is not a G-PDU packet (i.e. it is a GTP-U control packet), the processing must proceed via the gateway's slow path (software) control plane. GTP-C and GTP' packets directed to the gateway's IP address are a result of mis-configuration and are in error. They must be sent to the OpenFlow controller, since these packets are handled by the S-GW-C and P-GW-C control plane entities in the cloud computing system or to the billing entity handling GTP' and not the S-GW-D and P-GW-D data plane switches.")<br><br>Kempf at [0087] ("In one embodiment, slow path support for GTP is implemented with an OpenFlow gateway switch. An Open-Flow mobile gateway switch also contains support on the software control plane for slow path packet processing. This path is taken by G-PDU |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 626 of 1100

| No. | '111 Patent Claim 8 | Lin '400 |
|---|---|---|
| | | (message type 255) packets with nonzero header fields or extension headers, and user data plane packets requiring encapsulation with such fields or addition of extension headers, and by G TP-U control packets. For this purpose, the switch supports three local ports in the software control plane: LOCAL_GTP _CONTROL-the switch fast path forwards GTP encapsulated packets directed to the gateway IP address that contain GTP-U control mes-sages and the local switch software control plane initiates local control plane actions depending on the GTP-U control message; LOCAL_GTP _U_DECAP-the switch fast path forwards G-PDU packets to this port that have nonzero header fields or extension headers (i.e. E!=0, S!=0, or PN!=0). These packets require specialized handling. The local switch software slow path processes the packets and performs the specialized handling; and LOCAL_GTP _U_ENCAP-the switch fast path forwards user data plane packets to this port that require encapsulation in a GTP tunnel with nonzero header fields or extension headers (i.e. E!=0, S!=0, or PN!=0). These packets require specialized handling. The local switch software slow path encapsulates the packets and performs the specialized handling. In addition to forwarding the packet, the switch fast path makes the OpenFlow metadata field avail-able to the slow path software.")<br><br>Kempf at [0091] ("When a packet header matches a rule associated with the virtual port, the GTP TEID is written into the lower 32 bits of the metadata and the packet is directed to the virtual port. The virtual port calculates the hash of the TEID and looks up the tunnel header information in the tunnel header table. If no such tunnel information is present, the packet is forwarded to the controller with an error indication. Other-wise, the virtual port constructs a GTP tunnel header and encapsulates the packet. Any DSCP bits or VLAN priority bits are additionally set in the IP or MAC tunnel headers, and any VLAN tags or MPLS labels are pushed onto the packet. The encapsulated packet is forwarded out the physical port to which the virtual port is bound.")<br><br>Kempf at [0092] ("In one embodiment, the system implements a GTP fast path decapsulation virtual port. When requested by the S-GW and P-GW control plane software running in the cloud computing system, the gateway switch installs rules and actions for routing GTP encapsulated packets out of GTP tunnels. The rules match the GTP header flags and the GTP TEID for the packet, in the modified OpenFlow flow table shown in FIG. 17 as follows: the IP destination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); the UDP destination port is the GTP-U destination port (2152); |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 627 of 1100

| No. | '111 Patent Claim 8 | Lin '400 |
|---|---|---|
| | | and the header fields and message type field is wildcarded with the flag 0XFFF0 and the upper two bytes of the field match the G-PDU message type (255) while the lower two bytes match 0x30, i.e. the packet is a GTP packet not a GTP' packet and the version number is 1.")<br><br>Kempf at [0098] ("The header flags and message type fields for the three rules are wildcarded with the following bitmasks and match as follows: bitmask 0xFFF4 and the upper two bytes match the G-PDU message type (255) while the lower two bytes are Ox34, indicating that the version number is 1, the packet is a GTP packet, and there is an extension header present; bitmask 0xFFFF2 and the upper two bytes match the G-PDU message type (255) while the lower two bytes are 0x32, indicating that the version number is 1, the packet is a GTP packet, and there is a sequence number bitmask 0xFF0l and the upper two bytes match the G-PDU message type (255) while the lower two bytes are 0x31, indicating that the version number is 1, the packet is a GTP packet, and a N-PDU is present.")<br><br>Kempf at [0101] ("In one embodiment, the system implements han-dling of user data plane packets requiring GTP-U encapsula-tion with extension headers, sequence numbers, and N-PDU numbers. User data plane packets that require extension head-ers, sequence numbers, or N-PDU numbers during GTP encapsulation require special handling by the software slow path. For these packets, the OpenFlow controller programs a rule matching the 4 tuple: IP source address; IP destination address; UDP/TCP/SCTP source port; and UDP/TCP/SCTP destination port. The instructions for matching packets are:<br><br>Write-Metadata ( GTP-TEID, 0x FFFFFFFF)<br>Apply-Actions (Set-Output-Port LOCAL_GTP _U_ENCAP )")<br><br>For example, Copeland discloses identifying the sampled packet headers comprised of fragmented packets of smaller byte sizes.<br><br>Copeland [0079] ("Large packets tend to be fragmented by networks that cannot handle a large packet size. A 16-bit packet identification is used to reassemble fragmented packets. Three one-bit set of fragmentation flags control whether a packet is or may be fragmented. The 13-bit fragment offset is a sequence number for the 4-byte words in the packet when reassembled. In a series of fragments, the first offset will be zero.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 628 of 1100

| No. | '111 Patent Claim 8 | Lin '400 |
|---|---|---|
| | | Copeland at [0080] ("After the fragmentation information, an 8-bit time to live field specifies the remaining life of a packet and is decremented each time the packet is relayed. If this field is 0, the packet is destroyed. Next is an 8-bit protocol field that specifies the transport protocol used in the data portion. The following 16-bit field is a header checksum on the header only. Finally, the last two fields illustrated contain the 32-bit source address and 32-bit destination address. IP packet data follows the address information.")<br><br>Copeland at [0081] ("In a TCP/IP datagram 210, the initial data of the IP datagram is the TCP header 230 information. The initial TCP header 230 information includes the 16-bit source and 16-bit destination port numbers. A 32-bit sequence number for the data in the packet follows the port numbers. Following the sequence number is a 32-bit acknowledgement number. If an ACK flag (discussed below) is set, this number is the next sequence number the sender of the packet expects to receive. Next is a 4-bit data offset, which is the number of 32-bit words in the TCP header. A 6-bit reserved field follows.") |

| No. | '111 Patent Claim 9 | Lin '400 |
|---|---|---|
| 9 | The method according to claim 5, further comprising responsive to receiving the packet, analyzing the packet, by the controller. | Lin '400 the method according to claim 5, further comprising discloses responsive to receiving the packet, analyzing the packet, by the controller.<br><br>For example, Lin '400 discloses security service that inspects the packets. A person of ordinary skill in the art would understand that the security service could be located in the SDN controller. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Lin '400 is found to not meet this limitation, responsive to receiving the packet, analyzing the packet, by the controller would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>*See supra* at Claim 5.<br><br>Lin '400 3:11-24 ("Network security vendors provide network security services, such as firewall or deep packet inspection (DPI). Generally speaking, to provide network security services, packets of network traffic are intercepted for inspection. One way of intercepting network traffic is to place the security service in the middle of the packet forwarding path |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 629 of 1100

| No. | '111 Patent Claim 9 | Lin '400 |
|---|---|---|
| | | This is illustrated in FIG. 3, where packets from a sender component (e.g., a sender computer) are received in an ingress port of a switch, forwarded to an egress port of the switch, and forwarded to the ingress port of a security component, such as a security service. The security service may inspect the packets, and forward the packets to an egress port of the switch toward the next hop, which may be another switch or a destination component (e.g., destination computer), for example.").<br><br>Lin '400 5:37-55 ("The SDN computer network 600 may include a security component in the form of the security service 630. The security service 630 may comprise a virtual machine that provides computer network security services, such as packet inspection, for the sender component 622 and other virtual machines. For example, the security service 630 may comprise a virtual machine with a virtual network interface card that is coupled to the redirect port 623-2 and re-inject port 623-3 of the SDN switch 620. The security service 630 may inspect packets for compliance/non-compliance with security policies, such as for presence of malicious code, compliance with firewall rules and access control lists, network intrusion detection, and other computer network security services. The security service 630 may employ conventional packet inspection algorithms. The security service 630 may comprise the Trend Micro Deep Security™ service, for example. The security service 630 may also comprise a physical machine, e.g., a server computer, an appliance, a gateway computer, etc.").<br><br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Lin '400 in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 9 of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references.<br><br>For example, Swenson discloses the network controller comprising a flow analyzer for analyzing and inspecting the packet.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 630 of 1100

| No. | '111 Patent Claim 9 | Lin '400 |
|---|---|---|
| | | 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0028] ("In contrast to conventional inline TCP throughput monitoring devices that monitor every single data packets transmitted and received, the network controller 140 is an "out-of-band" computer server that interfaces with the steer-ing device 130 to selectively inspect user flows of interest. The network controller 140 may further identify user flows (e.g., among the flows of interest) for optimization. In one embodiment, the network controller 140 may be imple-mented at the steering device 130 to monitor traffic. In other embodiments, the network controller 140 is coupled to and communicates with the steering device 130 for traffic moni-toring and optimization. When queried by the steering device 130, the network controller 140 determines if a given network flow should be ignored, monitored further or optimized. Opti-mization of a flow is often decided at the beginning of the flow because it is rarely possible to switch to optimized content mid-stream once non-optimized content delivery has begun. However, the network controller 140 may determine that existing flows associated with a particular subscriber or other entity should be optimized. In turn, new flows ( e.g., resulting from seek requests in media, new media requests, resume after pause, etc.) determined to be associated with the entity may be optimized. The network controller 140 uses the net-work state as well as historical traffic data in its decision for monitoring and optimization. Knowledge on the current net-work state, such as congestion, deems critical when it comes to data optimization.")<br><br>Swenson at [0029] ("As a flow is sent to the network controller 140 for inspection, historical network traffic data stored at the net-work controller 140 may be searched. The historical network traffic data includes information such as subscriber informa-tion, the cell towers to which the user devices attached, rout-ers through which the traffic is passing, geography regions, the backhaul segments, and time-of-day of the flows. For example, in a mobile network, the cell tower to which a user device is attached can be most useful, since it is the location where most congestion occurs due to limited bandwidth and high cost of the radio access network infrastructure. The network controller 140 looks into the historical traffic data |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 631 of 1100

| No. | ’111 Patent Claim 9 | Lin ’400 |
|---|---|---|
| | | for the average of the bandwidth per user at the particular cell tower. The network controller 140 can then estimate the amount of bandwidth or degree of congestion for the new flow based on the historical record.”)<br><br>Swenson at [0038] (“Turning back to FIG. 1, the network controller 140 allows network operators to apply fine granular optimization policies to ensure high quality of experience (QoE) based on cell tower congestion, device types, subscriber profiles and service plans with lower hardware and software costs. The architecture of the network controller 140 provides an excel-lent fit for the net neutrality guideline of "reasonable network management", and better compliance to the copyright law (DMCA) than solutions that rely on long-term caching. Hav-ing the ability of monitoring network traffic on a per sub-scriber, per flow, or per video file basis, the network controller 140 also selectively monitors and optimizes only a subset of traffic that benefits from optimization the most, thus achiev-ing both scalability and efficiency for optimization at a com-petitive price-point. The core element of the network control-ler 140 lies in its mechanisms for congestion detection and mitigation, which allows optimization resources to be utilized in the most efficient and surgical manner.”)<br><br>Swenson at [0039] (“Referring now to FIG. 3, it illustrates one embodi-ment of an example architecture of the network controller 140 for providing selective real-time network monitoring and subscriber identification. The network controller 140 com-prises a flow analyzer 312, a policy engine 314, a steering device interface 316, a video optimizer redirector 318, a flow cache 322, and a subscriber log 324. In other embodiments, the network controller 140 may include additional, fewer, or different components for various applications. Conventional components such as network interfaces, security functions, failover servers, management and network operations con-soles, and the like are not shown so as to not obscure the details of the system architecture.”)<br><br>Swenson at [0059] (“In one embodiment, as the steering device 130 monitors network responses, it is looking for flows that match one or more signatures for video and images. When a match-ing flow is detected, the steering device 130 forwards the HTTP request and a portion of the HTTP response to the network controller 140 over the ICAP client interface 404. After receiving the request and the portion of response at the ICAP server interface 406, the flow analyzer 312 of the net-work controller 140 performs a deep flow inspection to deter-mine if the flow is worth bandwidth monitoring and/or user detection. For example, the |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 632 of 1100

| No. | '111 Patent Claim 9 | Lin '400 |
|---|---|---|
| | | flow inspection performed by the flow analyzer 312 may determine if the flow indeed contains large or medium object ( e.g., larger than 50 kB), and/or if the source IP address of the flow is from a user or a group of users that are required to be monitored by policies. The flow ana-lyzer 312 may also determine if the flow needs to be opti-mized based on historical flow statistical data.")<br><br>Swenson at [0060] ("If the flow is deemed of interest, the steering device 130 is notified to steer the flow through the network controller 140. This is known as the "continue" working mode for bandwidth monitoring. In the "continue" mode, the network controller 140 interfaces with the steering device 130 to func-tion, on-demand, as a traditional inline network element for flows deemed of interest. Thus, the network controller 140 ingests the network flow for inspection and subsequently forwards the network flow on the network response path. For example, for this particular flow, the origin server 160 responds to the user request by sending video or images over the network link 413 to the steering device 130, which for-wards the video or images to the network controller 140 over a network link 414. After the network controller 140 updates the flow statistics, the video or images are returned to the steering device 130 over a network link 415, which transmits the video or images to the user device 110 over the network link 416.")<br><br>Swenson at [0061] ("Once a flow is reported to the network controller 140, a flow cache entry is created for the flow in the flow cache 322. The flow cache entry keeps track of the flow and its associated bandwidth. For a flow that is marked in "continue" mode, each time the steering device 130 forwards a next portion of the flow payload to the network controller 140, the flow cache 3 22 updates the number of bytes for transmitted in the flow. By monitoring the number of bytes per flow over time, the flow analyzer 312 is capable of determining an estimate value of bandwidth associated with flow. Further-more, since the steering device 130 does not have infinite packet buffers, if congestion happens on the network link 416 from the steering device 130 to the user device 110, the TCP congestion control mechanism kicks in at the steering device 130, which may slows down and/or eventually stop receiving data over the network link 413 from origin server 160. During the congestion, the steering device 130 would not forward any data to the network controller 140, since the link 416 is congested and the network controller 140 would not be able to transmit data to the user device 110. Therefore, as an inline element, the network controller 140 can detect network con-gestions and estimate bandwidth associated with any flows of interest selected by the network |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 633 of 1100

| No. | ’111 Patent Claim 9 | Lin ’400 |
|---|---|---|
| | | controller 140. However, in the "continue" mode, the network controller 140 does not modify and transform the HTTP messaged it receives over the ICAP interface. The network controller 140 simply updates the flow statistics and returns the video or images to the steering device 130 for transmission to the user device 110.”)<br><br>Swenson at [0065] (“FIG. 5 is a block diagram illustrating an example event trace of "continue" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 512 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 512 to the intended origin server 160 and receives a response 514 back from the origin server 160. The steering device 130 then sends an ICAP request message 516 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 518. Upon receiving the instruc-tion, the steering device 130 re-writes the response 514 to an HTTP redirect response 520, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 520 directly to the user device 110. In case the flow dose not contain video or image objects, or the network controller 140 determines not to monitor the flow, the steering device 13 0 would forward the response to the user device 110.”)<br><br>Swenson at [0069] (“FIG. 6 is a block diagram illustrating an example event trace of "counting" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 612 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 612 to the intended origin server 160 and receives a response 614 back from the origin server 160. The steering device 130 then sends an ICAP request message 616 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 618. |

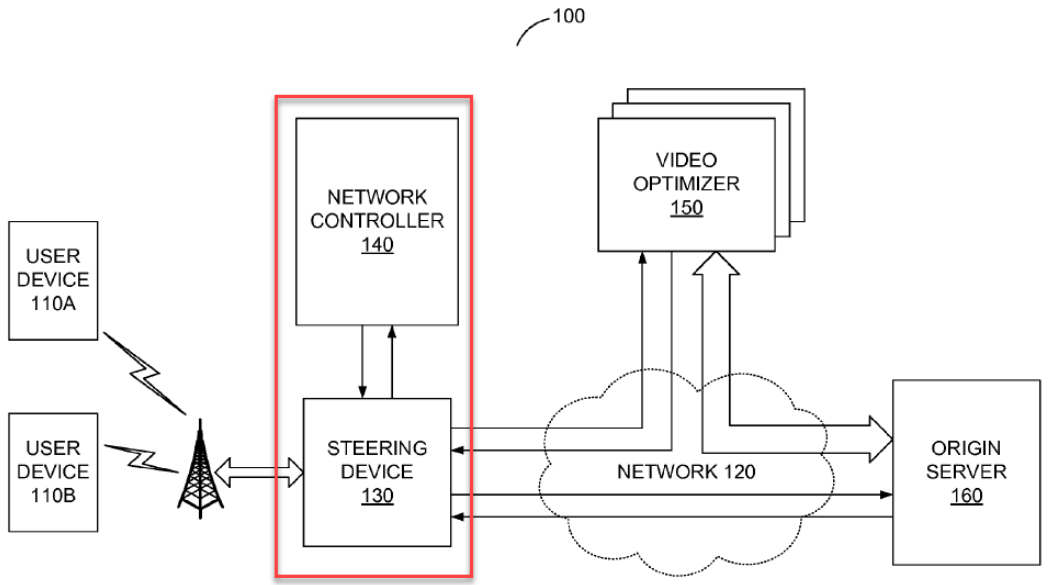


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 634 of 1100

| No. | '111 Patent Claim 9 | Lin '400 |
|---|---|---|
| | | Upon receiving the instruc-tion, the steering device 130 re-writes the response 614 to an HTTP redirect response 620, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 620 directly to the user device 110. In case the flow dose not contain video or image objects that need to be redirected, the steering device 130 would forward the response to the user device 110.")<br><br>Swenson at [0071] ("After receiving the request, the video optimizer 150 forwards the video HTTP GET requests 622 to the origin server 160 and in return, receives a video file 624 from the origin server 160. The video optimizer 150 transcodes the video file to a format usable by the client device 110 based on network bandwidth available to the user device 110. The optimized video 626 is then transmitted from the video opti-mizer 150 to the steering device 130. In one embodiment, the steering device 130 intercepts the optimized video 626. The steering device 130 will then send an ICAP request to the network controller 140 for inspection. The network controller 140 deems this flow to be monitored and sends ICAP response 630. The steering device 130 then allows the flow to go through to the user device 110. The steering device 130 next sends periodic ICAP "counting" updates 632 to the network controller 140 until the flow completes. As such, the client receives the optimized video 626 for substantially real-time playback on an application executing on the user device 110.")<br><br>Swenson at Figure 1 (annotation added) |

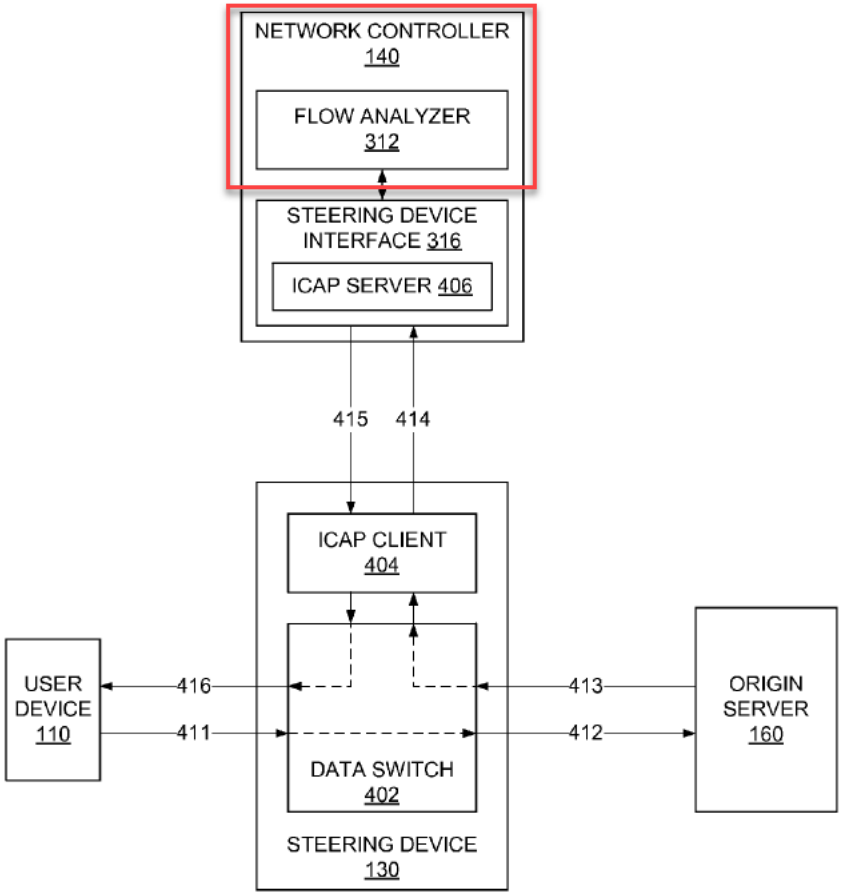


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 635 of 1100

| No. | '111 Patent Claim 9 | Lin '400 |
|---|---|---|
| | |  <br>FIG. 1<br>Swenson at Figure 4A (annotation added) |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 636 of 1100

| No. | '111 Patent Claim 9 | Lin '400 |
|---|---|---|
| | |  <br>FIG. 4A<br>For example, Copeland discloses analyzing packets received by the intrusion detection engine on the monitoring appliance. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 637 of 1100

| No. | '111 Patent Claim 9 | Lin '400 |
|---|---|---|
| | | Copeland at [0021] ("The present invention provides an accurate and reliable method for detecting network attacks through the use of sampled packet headers that are provided by a source such as that as defined in sFlow and further based in large part on "flows" as opposed to signatures or anomalies. By utilizing the host and flow information structures that are inherent with flow-based analysis and applying rules of statistical significance and analysis, the intrusion detection system can operate with sampled data such as provided by sFlow in order to provide a hybrid solution that combines many of the benefits of a packet capture implementation with the distributed nature of an IDS that operates on Netflow, thus providing an enhanced wide-area IDS solu-tion.")<br><br>Copeland at [0023] ("According to one aspect of the invention, the detection system works by assigning sampled data packets to various client/server ( C/S) flows. Statistics are collected for each determined flow. Then, the flow statistics are analyzed to determine if the flow appears to be legitimate traffic or possible suspicious activity. A value, referred to as a "concern index," is assigned to each flow that appears suspicious. By assigning a value to each flow that appears suspicious and adding that value to an accumulated concern index associated with the responsible host, it is possible to identify hosts that are engaged in intruder activity without generation of significant unwarranted false alarms. When the concern index value of a host exceeds a preset alarm value, an alert is issued and appropriate action can be taken.")<br><br>Copeland at [0024] ("Generally speaking, the intrusion detection system analyzes network communication traffic for potential detrimental activity. The system collects flow data from sampled packet headers between two hosts or Internet Protocol (IP) addresses. Collecting flow data from packet headers asso-ciated with a single service where at least one port remains constant allows for more efficient analysis of the flow data. The collected flow data is analyzed to assign a concern index value to the flow based upon a probability that the flow was not normal for data communications. A host list is main-tained containing an accumulated concern index derived from the flows associated with the host. Once the accumu-lated concern index has exceeded an alarm threshold value, an alarm signal is generated.")<br><br>Copeland at [0027] ("According to one aspect of the invention, the detection system works by assigning sampled data packets to various client/server ( C/S) flows. Statistics are collected for each determined flow. Then, the flow statistics are analyzed to determine if the flow |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 638 of 1100

| No. | ’111 Patent Claim 9 | Lin ’400 |
|---|---|---|
| | | appears to be legitimate traffic or possible suspicious activity. A value, referred to as a "concern index," is assigned to each flow that appears suspicious. By assigning a value to each flow that appears suspicious and adding that value to an accumulated concern index associated with the responsible host, it is possible to identify hosts that are engaged in intruder activity without generation of significant unwarranted false alarms. When the concern index value of a host exceeds a preset alarm value, an alert is issued and appropriate action can be taken.”)<br><br>Copeland at [0028] (“Generally speaking, the intrusion detection system analyzes network communication traffic for potential detri-mental activity. The system collects flow data from sampled packet headers between two hosts or Internet Protocol (IP) addresses. Collecting flow data from packet headers asso-ciated with a single service where at least one port remains constant allows for more efficient analysis of the flow data. The collected flow data is analyzed to assign a concern index value to the flow based upon a probability that the flow was not normal for data communications. A host list is main-tained containing an accumulated concern index derived from the flows associated with the host. Once the accumu-lated concern index has exceeded an alarm threshold value, an alarm signal is generated.”)<br><br>Copeland at [0063] (“Consequently, abnormal flows and/or events iden-tified by the intrusion detection engine 155 will raise the concern index (CI) for the associated host. The intrusion detection engine 155 analyzes the data flow between IP devices. However, different types of services have different flow characteristics associated with that service. Therefore, a C/S flow can be determined by the packets exchanged between the two hosts dealing with the same service.”)<br><br>Copeland at [0065] (“The intrusion detection engine 155 analyzes the flow data 160 to determine if the flow appears to be legitimate traffic or possible suspicious activity. Flows with suspicious activity are assigned a predetermined concern index (CI) value based upon a heuristically predetermined assessment of the significance of the threat of the particular traffic or flow or suspicious activity. The flow concern index values have been derived heuristically from extensive net-work traffic analysis. Concern index values are associated with particular hosts and stored in the host data structure 166 (FIG. 1). Exemplary concern index values for various exemplary flow-based events and other types of events are illustrated in connection with FIGS. 6 and 7.) |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 639 of 1100

| No. | '111 Patent Claim 9 | Lin '400 |
|---|---|---|
| | | Copeland at [0069] ("It will now be appreciated that the disclosed meth-odology of intrusion detection is accomplished at least in part by analyzing communication flows to determine if such communications have the flow characteristics of probes or attacks. By analyzing communications for abnormal flow characteristics, attacks can be determined without the need for resource-intensive packet data analysis. A flow can be determined from the packets 101 that are transmitted between two hosts utilizing a single service. The addresses and port numbers of communications are easily discerned by analysis of the header information in a datagram.")<br><br>Copeland at [0087] ("As previously stated, the flow-based engine 155 does not analyze the data segments of packets for signature identification. Instead, the engine 155 associates all packets with a flow. It analyzes certain statistical data and assigns a concern index value to abnormal activity. The engine 155 builds a concern index for suspicious hosts by detecting suspicious activities on the network. An alarm is generated when those hosts build enough concern (in the form of a cumulated CI value) to cross the network administrator's predetermined threshold.")<br><br>Copeland at [0097] ("The described TCP session 300 of FIG. 3 is a generic TCP session in which a network might engage. In accordance with the invention, flow data is collected about the session to help determine if the communication is abnormal. In the preferred embodiment, information such as the total number of packets sent, the total amount of data sent, the session start time and duration, and the TCP flags set in all of the packets, are collected, stored in the database 160, and analyzed to determine if the communication was suspicious. If a communication is deemed suspicious, i.e. it meets predetermined criteria, a predetermined concern index value associated with a determined category of suspicious activity is added to the cumulated CI value associated with the host that made the communication.")<br><br>Copeland at [0111] ("As shown, the packets exchanged between two hosts associated with a single service can determine a flow. A port number designates a service application that is associated with the particular port. Communications utiliz-ing differing protocols or services provide differing flow characteristics. Consequently, the flow engine 155 analyzes each of the services separately.") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 640 of 1100

| No. | '111 Patent Claim 9 | Lin '400 |
|---|---|---|
| | | Copeland at [0150] ("A preferred hardware configuration 800 of an embodiment that executes the functions of the above-described flow-based engine is described in reference to FIG. 8. FIG. 8 illustrates a typically hardware configuration 800 for a network intrusion detection system. A monitoring appliance 150 serves as a pass-by filter of network traffic. A network device 135, such as a router or switch supporting sFlow provides the location for connecting the monitoring appliance 150 to the network 899 for monitoring the network traffic.") |

| No. | '111 Patent Claim 12 | Lin '400 |
|---|---|---|
| 12 | The method according to claim 9, wherein the analyzing comprises applying security or data analytic application. | Lin '400 discloses the method according to claim 9, wherein the analyzing comprises applying security or data analytic application.<br><br>For example, Lin '400 discloses a security processing function that employs conventional packet inspection algorithms.<br><br>Lin '400 3:11-24 ("Network security vendors provide network security services, such as firewall or deep packet inspection (DPI). Generally speaking, to provide network security services, packets of network traffic are intercepted for inspection. One way of intercepting network traffic is to place the security service in the middle of the packet forwarding path. This is illustrated in FIG. 3, where packets from a sender component (e.g., a sender computer) are received in an ingress port of a switch, forwarded to an egress port of the switch, and forwarded to the ingress port of a security component, such as a security service. The security service may inspect the packets, and forward the packets to an egress port of the switch toward the next hop, which may be another switch or a destination component (e.g., destination computer), for example.").<br><br>Lin '400 5:37-55 ("The SDN computer network 600 may include a security component in the form of the security service 630. The security service 630 may comprise a virtual machine that provides computer network security services, such as packet inspection, for the sender component 622 and other virtual machines. For example, the security service 630 may comprise a virtual machine with a virtual network interface card that is coupled to the redirect port 623-2 and re-inject port 623-3 of the SDN switch 620. The security service 630 may inspect packets for compliance/non-compliance with security policies, such as for presence of malicious code, compliance with firewall rules and access control lists, network intrusion |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 641 of 1100

| No. | '111 Patent Claim 12 | Lin '400 |
|---|---|---|
| | | detection, and other computer network security services. The security service 630 may employ conventional packet inspection algorithms. The security service 630 may comprise the Trend Micro Deep Security™ service, for example. The security service 630 may also comprise a physical machine, e.g., a server computer, an appliance, a gateway computer, etc."). |

| No. | '111 Patent Claim 13 | Lin '400 |
|---|---|---|
| 13 | The method according to claim 9, wherein the analyzing comprises applying security application that comprises firewall or intrusion detection functionality. | Lin '400 discloses the method according to claim 9, wherein the analyzing comprises applying security application that comprises firewall or intrusion detection functionality.<br><br>For example, Lin '400 discloses network security service application, such as a firewall or DPI.<br><br>Lin '400 3:11-24 ("Network security vendors provide network security services, such as firewall or deep packet inspection (DPI). Generally speaking, to provide network security services, packets of network traffic are intercepted for inspection. One way of intercepting network traffic is to place the security service in the middle of the packet forwarding path. This is illustrated in FIG. 3, where packets from a sender component (e.g., a sender computer) are received in an ingress port of a switch, forwarded to an egress port of the switch, and forwarded to the ingress port of a security component, such as a security service. The security service may inspect the packets, and forward the packets to an egress port of the switch toward the next hop, which may be another switch or a destination component (e.g., destination computer), for example.").<br><br>Lin '400 5:37-55 ("The SDN computer network 600 may include a security component in the form of the security service 630. The security service 630 may comprise a virtual machine that provides computer network security services, such as packet inspection, for the sender component 622 and other virtual machines. For example, the security service 630 may comprise a virtual machine with a virtual network interface card that is coupled to the redirect port 623-2 and re-inject port 623-3 of the SDN switch 620. The security service 630 may inspect packets for compliance/non-compliance with security policies, such as for presence of malicious code, compliance with firewall rules and access control lists, network intrusion |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 642 of 1100

| No. | '111 Patent Claim 13 | Lin '400 |
|---|---|---|
| | | detection, and other computer network security services. The security service 630 may employ conventional packet inspection algorithms. The security service 630 may comprise the Trend Micro Deep Security™ service, for example. The security service 630 may also comprise a physical machine, e.g., a server computer, an appliance, a gateway computer, etc."). |

| No. | '111 Patent Claim 14 | Lin '400 |
|---|---|---|
| 14 | The method according to claim 9, wherein the analyzing comprises performing Deep Packet Inspection (DPI) or using a DPI engine on the packet. | Lin '400 discloses the method according to claim 9, wherein the analyzing comprises performing Deep Packet Inspection (DPI) or using a DPI engine on the packet.<br><br>For example, Lin '400 discloses a security service on packets which includes deep packet inspection.<br><br>*See supra* Claim 9.<br><br>Lin '400 3:11-24 ("Network security vendors provide network security services, such as firewall or deep packet inspection (DPI). Generally speaking, to provide network security services, packets of network traffic are intercepted for inspection. One way of intercepting network traffic is to place the security service in the middle of the packet forwarding path. This is illustrated in FIG. 3, where packets from a sender component (e.g., a sender computer) are received in an ingress port of a switch, forwarded to an egress port of the switch, and forwarded to the ingress port of a security component, such as a security service. The security service may inspect the packets, and forward the packets to an egress port of the switch toward the next hop, which may be another switch or a destination component (e.g., destination computer), for example."). |

| No. | '111 Patent Claim 15 | Lin '400 |
|---|---|---|
| 15[a] | The method according to claim 9, wherein the packet comprises distinct header and payload fields, and | Lin '400 discloses the method according to claim 9, wherein the packet comprises distinct header and payload fields.<br><br>For example, Lin '400 discloses flow rules that check the source address and destination address of a packet, which is part of a packet header. Lin '400 further discloses inspection of |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 643 of 1100

| No. | '111 Patent Claim 15 | Lin '400 |
|---|---|---|
| | | packets to determine if a packet payload is prohibited or malicious. A person of ordinary skill in the art would understand that data traffic is made up of packets comprised of header and payload fields. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Lin '400 is found to not meet this limitation, wherein the packet comprises distinct header and payload fields would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>*See supra* Claim 9.<br><br>Lin '400 5:8-25 ("A flow table may include columns that indicate one or more conditions, a column that indicates an action to take when the conditions are met, and a column for statistics. A row on the flow table may comprise a flow rule. In the example of Table 1, the "Action" column indicates an action to take when conditions are met, and the "Count" column indicates statistics, such as byte count. The rest of the columns of Table 1 indicate conditions. For example, "IN_PORT", "MAC src" (media access control (MAC) address of the source of the packet), "MAC dst" (MAC address of the destination of the packet), "IP src" (Internet Protocol (IP) address of the source of the packet), "IP dst" (IP address of the destination of the packet), etc. are conditions that identify a particular packet. When the conditions are met, i.e., the particular packet is identified, the action indicated in the corresponding "Action" column is performed on the packet. The asterisks in Table 1 indicate an irrelevant condition.").<br><br>Lin '400 6:40-54 ("After the redirect flow rules for creating the SDN pipe are inserted in the flow tables 621, any packet received by the SDN switch 620 in the ingress port 623-1 will be identified as to be forwarded to the redirect port 623-2, and any packet received by the SDN switch 620 in the redirect port 623-2 will be identified as to be forwarded to the ingress port 623-1 (see arrow 602). This allows the security service 630 to receive from the redirect port 623-2 all outgoing packets sent by the sender component 622 to the ingress port 623-1. The security service 630 may inspect the outgoing packets for compliance with security policies. The security service 630 may drop, or perform other security response, to packets that do not pass inspection (e.g., packets that do not meet firewall policies, packets containing prohibited payload, packets with malicious content, etc.)."). |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 644 of 1100

| No. | '111 Patent Claim 15 | Lin '400 |
|---|---|---|
| | | Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Lin '400 in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 15(a) of Exhibit E-4 renders the claim, including the present limitation, obvious. Below is an example.<br><br>For example, Swenson discloses packet flows with header and payload fields.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0040] ("The flow analyzer 312 monitors large flows in the network, analyzes collected flow statistics to determine net-work throughput, and accordingly selects flows to be opti-mized. The flow analyzer 312 does not need to see all the flows in order to make an accurate estimate of network con-ditions. The flow analyzer 312 processes the traffic statistics stored in the flow cache 3 22 and user information stored in the subscriber log 324, for example, by associating network flows identified by source IP addresses to a mobile subscriber or user, which is identified by his or her current subscriber ID or device ID. The user flows are also mapped to a congestion level at the current sub-network (e.g., a cell with which the user devices are associated), so that an optimization decision can be made at the beginning of the data transmission.")<br><br>Swenson at [0049] ("The policy engine 314 defines policies for optimiz-ing large flows with media objects to mitigate network con-gestion. Detecting and acting on congestion in the network, the design focus of the network controller 140 is built on this very flexible policy engine. The policy engine 314 is capable of taking virtually any input, either deduced from |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 645 of 1100

| No. | '111 Patent Claim 15 | Lin '400 |
|---|---|---|
| | | HTTP headers and payload ( e.g., through RADIUS/Gx interface), or provided by the network infrastructure via API, and making decisions on how to apply optimization based on individual or a combination of these inputs. The optimization policies can be applied to large flows all the time or on a time-of-day basis, a per user basis, and/or depending on the network condition.")<br><br>Swenson at [0061] ("Once a flow is reported to the network controller 140, a flow cache entry is created for the flow in the flow cache 322. The flow cache entry keeps track of the flow and its associated bandwidth. For a flow that is marked in "continue" mode, each time the steering device 130 forwards a next portion of the flow payload to the network controller 140, the flow cache 3 22 updates the number of bytes for transmitted in the flow. By monitoring the number of bytes per flow over time, the flow analyzer 312 is capable of determining an estimate value of bandwidth associated with flow. Further-more, since the steering device 130 does not have infinite packet buffers, if congestion happens on the network link 416 from the steering device 130 to the user device 110, the TCP congestion control mechanism kicks in at the steering device 130, which may slows down and/or eventually stop receiving data over the network link 413 from origin server 160. During the congestion, the steering device 130 would not forward any data to the network controller 140, since the link 416 is congested and the network controller 140 would not be able to transmit data to the user device 110. Therefore, as an inline element, the network controller 140 can detect network con-gestions and estimate bandwidth associated with any flows of interest selected by the network controller 140. However, in the "continue" mode, the network controller 140 does not modify and transform the HTTP messaged it receives over the ICAP interface. The network controller 140 simply updates the flow statistics and returns the video or images to the steering device 130 for transmission to the user device 110.")<br><br>Swenson at [0064] (Similar to the "continue" mode, after receiving the initial HTTP messages of a flow and determining to monitor the flow, the network controller 140 notify the steering device 130 to work in a "counting" mode for bandwidth monitoring. In contrast to the "continue" mode, when a matching flow is detected for "counting" mode, the steering device 130 for-wards the HTTP response directly to the user device 110. While at the same time, the steering device 130 send a cus-tomized ICAP message to the network controller 140 over the network link 425. In one embodiment, the customized ICAP message contains the HTTP request and response headers, as well as a count of payload size of the current flow. After |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 646 of 1100

| No. | ’111 Patent Claim 15 | Lin ’400 |
|---|---|---|
| | | updating the flow statistics, the network controller 140 may acknowledge the gateway over the network line 426. In the "counting" mode, the network controller 140 does not join the network response path as an inline network element, but simply listens to the counting of flow size. The benefit of the "counting" mode is to off-load the network controller 140 from ingesting and forwarding the network flow on the net- work response path, while still enabling the detection of con-gestions and estimation of bandwidth associated with the flows of interest.”)<br><br>Swenson at [0065] (“FIG. 5 is a block diagram illustrating an example event trace of "continue" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 512 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 512 to the intended origin server 160 and receives a response 514 back from the origin server 160. The steering device 130 then sends an ICAP request message 516 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 518. Upon receiving the instruc-tion, the steering device 130 re-writes the response 514 to an HTTP redirect response 520, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 520 directly to the user device 110. In case the flow dose not contain video or image objects, or the network controller 140 determines not to monitor the flow, the steering device 13 0 would forward the response to the user device 110.”)<br><br>Swenson at [0069] (“FIG. 6 is a block diagram illustrating an example event trace of "counting" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 612 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 612 to the intended origin server 160 and receives a response 614 back from the origin server 160. The steering device 130 then sends an ICAP request message 616 comprising the HTTP GET request |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 647 of 1100

| No. | '111 Patent Claim 15 | Lin '400 |
|---|---|---|
| | | header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 618. Upon receiving the instruc-tion, the steering device 130 re-writes the response 614 to an HTTP redirect response 620, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 620 directly to the user device 110. In case the flow dose not contain video or image objects that need to be redirected, the steering device 130 would forward the response to the user device 110.")<br><br>Swenson at [0073] ("FIG. 7 is a block diagram illustrating one embodi-ment of an example of internal components of the flow cache. The flow cache map 700 comprises a plurality of flow cache entries, such as flow cache entries 710 and 712 indexed by a hash. Not shown in the example diagram is a possible linked list behind each flow cache entry which allows chaining of flow cache entries for a given hash index. The hash into the flow cache may be based on source IP address, MAC address, subscriber ID, or other identifier indicative of a given sub-scriber, group of subscribers or subscriber's device.")<br><br>Swenson at [0079] ("In the bandwidth calculation, flows are categorized into buckets based on the size of the objects being transferred. Small objects may not be factored into the bandwidth calcu-lation since they may come and go within a single interval. For example, flows with payload size less than 50 kB may be ignored because a transfer of 50 kB may never reach the full potential throughput of the link. While larger flows may reach the full throughput of the link for a long period of time intervals, they are grouped into 50-75 kB, 75-100 kB and 100 kB+ buckets because the characteristics of these flow sizes can be different, hence the bandwidth for each of the buckets is measured and calculated separately. In other embodiments, the flow size ranges (e.g., 50-75 kB, 75-100 kB and l00kB+) of the buckets may be altered depending on the network traffic and size of objects transmitted. Furthermore, the bucket sizes can also be adjusted based on network topology, such as buffer size, prior to transmission to the client. The calculated bandwidth per bucket is stored in a queue structure that allows for the computing and updating of minimum, maximum, and/or average measurements for each bucket. In one embodiment, the 100 kB+ bucket's current tail entry is checked against the average bandwidth for the 100 kB+ bucket. If the current entry is less than the average multiplied by the number of entries in the queue, the current entry is added |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 648 of 1100

| No. | '111 Patent Claim 15 | Lin '400 |
|---|---|---|
| | | to the bandwidth calculation for the current interval. This scheme can filter out large bursts of data from tempo-rarily idle flows. If the bandwidth exceeds the value, a number of bytes (e.g., 125 kB) will be subtracted from the current entry to account for TCP buffers in the network.")<br><br>Swenson at [0083] ("When a new flow is observed, flow cache entries are searched by matching source IP address 722 if the subscriber id or other identifiers of the flow are not available. In case of multiple users sharing an IP address, the flow analyzer 312 needs to find patterns or other identifiers in the flows to map them to particular subscribers. Flows without identified sub-scribers are added to the flow cache block under the default user flows 726, which is a default holding place for the new flows. The flow analyzer 312 later will scan through the default user flows that contain cookies or other identifiers that may be used to determine a real user or subscriber associated with the flow. If a flow contains identifiers not associated with an existing real user, a new user or subscriber is created and the user flow block is moved to newly created (or mapped) user or subscriber.") |
| 15[b] | wherein the analyzing comprises checking part of, or whole of, the payload field. | Lin '400 discloses wherein the analyzing comprises checking part of, or whole of, the payload field.<br><br>For example, Lin '400 discloses inspection of packets to determine if a packet payload is prohibited or malicious. A person of ordinary skill in the art would understand that inspection of packets occurs at the payload field. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Lin '400 is found to not meet this limitation, wherein the analyzing comprises checking part of, or whole of, the payload field would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>Lin '400 5:8-25 ("A flow table may include columns that indicate one or more conditions, a column that indicates an action to take when the conditions are met, and a column for statistics. A row on the flow table may comprise a flow rule. In the example of Table 1, the "Action" column indicates an action to take when conditions are met, and the "Count" column indicates statistics, such as byte count. The rest of the columns of Table 1 indicate conditions. For example, "IN_PORT", "MAC src" (media access control (MAC) address of the source of the packet), "MAC dst" (MAC address of the destination of the packet), "IP src" (Internet |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 649 of 1100

| No. | '111 Patent Claim 15 | Lin '400 |
|---|---|---|
| | | Protocol (IP) address of the source of the packet), "IP dst" (IP address of the destination of the packet), etc. are conditions that identify a particular packet. When the conditions are met, i.e., the particular packet is identified, the action indicated in the corresponding "Action" column is performed on the packet. The asterisks in Table 1 indicate an irrelevant condition.").<br><br>Lin '400 6:40-54 ("After the redirect flow rules for creating the SDN pipe are inserted in the flow tables 621, any packet received by the SDN switch 620 in the ingress port 623-1 will be identified as to be forwarded to the redirect port 623-2, and any packet received by the SDN switch 620 in the redirect port 623-2 will be identified as to be forwarded to the ingress port 623-1 (see arrow 602). This allows the security service 630 to receive from the redirect port 623-2 all outgoing packets sent by the sender component 622 to the ingress port 623-1. The security service 630 may inspect the outgoing packets for compliance with security policies. The security service 630 may drop, or perform other security response, to packets that do not pass inspection (e.g., packets that do not meet firewall policies, packets containing prohibited payload, packets with malicious content, etc.).").<br><br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Lin '400 in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 15(b) of Exhibit E-4 renders the claim, including the present limitation, obvious. Below is an example.<br><br>For example, Swenson discloses inspecting the payload of a packet flow.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 650 of 1100

| No. | '111 Patent Claim 15 | Lin '400 |
|---|---|---|
| | | 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0040] ("The flow analyzer 312 monitors large flows in the network, analyzes collected flow statistics to determine net-work throughput, and accordingly selects flows to be opti-mized. The flow analyzer 312 does not need to see all the flows in order to make an accurate estimate of network con-ditions. The flow analyzer 312 processes the traffic statistics stored in the flow cache 3 22 and user information stored in the subscriber log 324, for example, by associating network flows identified by source IP addresses to a mobile subscriber or user, which is identified by his or her current subscriber ID or device ID. The user flows are also mapped to a congestion level at the current sub-network (e.g., a cell with which the user devices are associated), so that an optimization decision can be made at the beginning of the data transmission.")<br><br>Swenson at [0049] ("The policy engine 314 defines policies for optimiz-ing large flows with media objects to mitigate network con-gestion. Detecting and acting on congestion in the network, the design focus of the network controller 140 is built on this very flexible policy engine. The policy engine 314 is capable of taking virtually any input, either deduced from HTTP headers and payload ( e.g., through RADIUS/Gx interface), or provided by the network infrastructure via API, and making decisions on how to apply optimization based on individual or a combination of these inputs. The optimization policies can be applied to large flows all the time or on a time-of-day basis, a per user basis, and/or depending on the network condition.")<br><br>Swenson at [0061] ("Once a flow is reported to the network controller 140, a flow cache entry is created for the flow in the flow cache 322. The flow cache entry keeps track of the flow and its associated bandwidth. For a flow that is marked in "continue" mode, each time the steering device 130 forwards a next portion of the flow payload to the network controller 140, the flow cache 3 22 updates the number of bytes for transmitted in the flow. By monitoring the number of bytes per flow over time, the flow analyzer 312 is capable of determining an estimate value of bandwidth associated with flow. Further-more, since the steering device 130 does not have infinite packet buffers, if congestion happens on the network link 416 from the steering device 130 to the user device 110, the TCP congestion control mechanism kicks in at the steering device 130, which may slows down and/or eventually stop receiving data over |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 651 of 1100

| No. | ’111 Patent Claim 15 | Lin ’400 |
|---|---|---|
| | | the network link 413 from origin server 160. During the congestion, the steering device 130 would not forward any data to the network controller 140, since the link 416 is congested and the network controller 140 would not be able to transmit data to the user device 110. Therefore, as an inline element, the network controller 140 can detect network con-gestions and estimate bandwidth associated with any flows of interest selected by the network controller 140. However, in the "continue" mode, the network controller 140 does not modify and transform the HTTP messaged it receives over the ICAP interface. The network controller 140 simply updates the flow statistics and returns the video or images to the steering device 130 for transmission to the user device 110.”)<br><br>Swenson at [0064] (Similar to the "continue" mode, after receiving the initial HTTP messages of a flow and determining to monitor the flow, the network controller 140 notify the steering device 130 to work in a "counting" mode for bandwidth monitoring. In contrast to the "continue" mode, when a matching flow is detected for "counting" mode, the steering device 130 for-wards the HTTP response directly to the user device 110. While at the same time, the steering device 130 send a cus-tomized ICAP message to the network controller 140 over the network link 425. In one embodiment, the customized ICAP message contains the HTTP request and response headers, as well as a count of payload size of the current flow. After updating the flow statistics, the network controller 140 may acknowledge the gateway over the network line 426. In the "counting" mode, the network controller 140 does not join the network response path as an inline network element, but simply listens to the counting of flow size. The benefit of the "counting" mode is to off-load the network controller 140 from ingesting and forwarding the network flow on the net- work response path, while still enabling the detection of con-gestions and estimation of bandwidth associated with the flows of interest.”)<br><br>Swenson at [0065] (“FIG. 5 is a block diagram illustrating an example event trace of "continue" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 512 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 512 to the intended origin server 160 and receives a response 514 back from the origin server 160. The steering device 130 then sends an ICAP request message 516 comprising the HTTP GET request |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 652 of 1100

| No. | '111 Patent Claim 15 | Lin '400 |
|---|---|---|
| | | header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 518. Upon receiving the instruc-tion, the steering device 130 re-writes the response 514 to an HTTP redirect response 520, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 520 directly to the user device 110. In case the flow dose not contain video or image objects, or the network controller 140 determines not to monitor the flow, the steering device 13 0 would forward the response to the user device 110.")<br><br>Swenson at [0069] ("FIG. 6 is a block diagram illustrating an example event trace of "counting" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 612 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 612 to the intended origin server 160 and receives a response 614 back from the origin server 160. The steering device 130 then sends an ICAP request message 616 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 618. Upon receiving the instruc-tion, the steering device 130 re-writes the response 614 to an HTTP redirect response 620, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 620 directly to the user device 110. In case the flow dose not contain video or image objects that need to be redirected, the steering device 130 would forward the response to the user device 110.")<br><br>Swenson at [0073] ("FIG. 7 is a block diagram illustrating one embodi-ment of an example of internal components of the flow cache. The flow cache map 700 comprises a plurality of flow cache entries, such as flow cache entries 710 and 712 indexed by a hash. Not shown in the example diagram is a possible linked list behind each flow cache entry which allows chaining of flow cache entries for a given hash index. The hash into the flow cache may be |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 653 of 1100

| No. | '111 Patent Claim 15 | Lin '400 |
|---|---|---|
| | | based on source IP address, MAC address, subscriber ID, or other identifier indicative of a given sub-scriber, group of subscribers or subscriber's device.")<br><br>Swenson at [0079] ("In the bandwidth calculation, flows are categorized into buckets based on the size of the objects being transferred. Small objects may not be factored into the bandwidth calcu-lation since they may come and go within a single interval. For example, flows with payload size less than 50 kB may be ignored because a transfer of 50 kB may never reach the full potential throughput of the link. While larger flows may reach the full throughput of the link for a long period of time intervals, they are grouped into 50-75 kB, 75-100 kB and 100 kB+ buckets because the characteristics of these flow sizes can be different, hence the bandwidth for each of the buckets is measured and calculated separately. In other embodiments, the flow size ranges (e.g., 50-75 kB, 75-100 kB and l00kB+) of the buckets may be altered depending on the network traffic and size of objects transmitted. Furthermore, the bucket sizes can also be adjusted based on network topology, such as buffer size, prior to transmission to the client. The calculated bandwidth per bucket is stored in a queue structure that allows for the computing and updating of minimum, maximum, and/or average measurements for each bucket. In one embodiment, the 100 kB+ bucket's current tail entry is checked against the average bandwidth for the 100 kB+ bucket. If the current entry is less than the average multiplied by the number of entries in the queue, the current entry is added to the bandwidth calculation for the current interval. This scheme can filter out large bursts of data from tempo-rarily idle flows. If the bandwidth exceeds the value, a number of bytes (e.g., 125 kB) will be subtracted from the current entry to account for TCP buffers in the network.")<br><br>Swenson at [0083] ("When a new flow is observed, flow cache entries are searched by matching source IP address 722 if the subscriber id or other identifiers of the flow are not available. In case of multiple users sharing an IP address, the flow analyzer 312 needs to find patterns or other identifiers in the flows to map them to particular subscribers. Flows without identified sub-scribers are added to the flow cache block under the default user flows 726, which is a default holding place for the new flows. The flow analyzer 312 later will scan through the default user flows that contain cookies or other identifiers that may be used to determine a real user or subscriber associated with the flow. If a flow contains identifiers not associated with an existing real user, a new user or subscriber is created and the user flow block is moved to newly created (or mapped) user or subscriber.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 654 of 1100

| No. | '111 Patent Claim 15 | Lin '400 |
|---|---|---|
| | | |

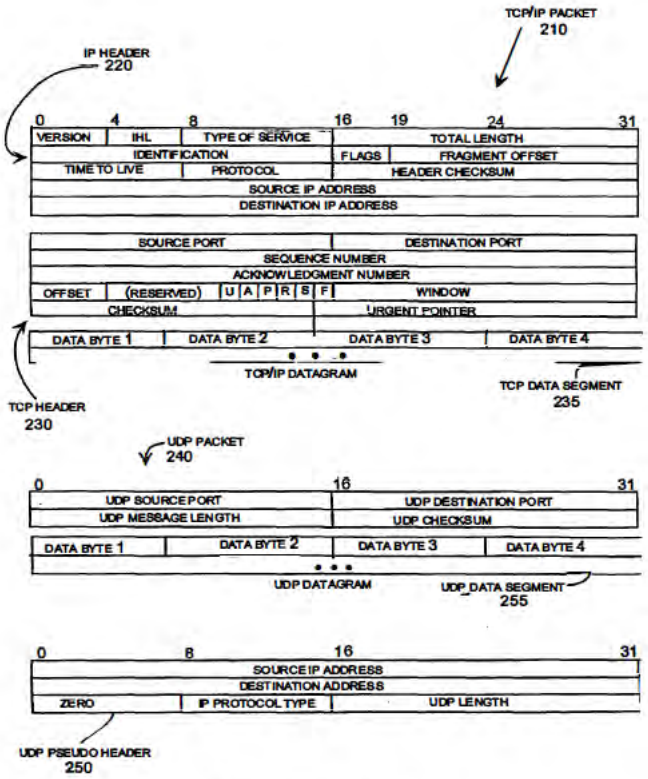
| No. | '111 Patent Claim 16 | Lin '400 |
|---|---|---|
| 16[a] | The method according to claim 1, wherein the packet comprises distinct header and payload fields, | Lin '400 discloses the method according to claim 1, wherein the packet comprises distinct header and payload fields.<br><br>*See supra* Claim 1, 15[a]. |
| 16[b] | the header comprises one or more flag bits, and | Lin '400 discloses the header comprises one or more flag bits.<br><br>For example, Lin '400 discloses flow rules that check the source address and destination address of a packet, which is part of a packet header. A person of ordinary skill in the art would understand that header fields can comprise one or more flag bits. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Lin '400 is found to not meet this limitation, the header comprises one or more flag bits would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>Lin '400 5:8-25 ("A flow table may include columns that indicate one or more conditions, a column that indicates an action to take when the conditions are met, and a column for statistics. A row on the flow table may comprise a flow rule. In the example of Table 1, the "Action" column indicates an action to take when conditions are met, and the "Count" column indicates statistics, such as byte count. The rest of the columns of Table 1 indicate conditions. For example, "IN_PORT", "MAC src" (media access control (MAC) address of the source of the packet), "MAC dst" (MAC address of the destination of the packet), "IP src" (Internet Protocol (IP) address of the source of the packet), "IP dst" (IP address of the destination of the packet), etc. are conditions that identify a particular packet. When the conditions are met, i.e., the particular packet is identified, the action indicated in the corresponding "Action" |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 655 of 1100

| No. | '111 Patent Claim 16 | Lin '400 |
|---|---|---|
| | | column is performed on the packet. The asterisks in Table 1 indicate an irrelevant condition."). <br><br> Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Lin '400 in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 16[b] of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references. <br><br> For example, Copeland discloses packet headers with flag bits. <br><br> Copeland at Figure 2 |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 656 of 1100

| No. | '111 Patent Claim 16 | Lin '400 |
|---|---|---|
| | |  <br>FIG. 2<br><br>Copeland at [0076] ("FIG. 2 illustrates an exemplary TCP/IP packet or datagram 210 and an exemplary UDP datagram 240. In a typical TCP/IP packet like 210, each packet typically includes a header portion comprising an IP header 220 and a TCP header 230, followed by a data portion that contains the information to be communicated in the packet. The information in the IP header 220 contained in a TCP/IP packet 210, or any other IP packet, contains the IP addresses and assures that the packet is delivered to the right host. The transport layer protocol (TCP) header follows the Internet protocol header and specifies the port numbers for the associated service.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 657 of 1100

| No. | '111 Patent Claim 16 | Lin '400 |
|---|---|---|
| | | Copeland at [0077] ("The header portion in the typical TCP/IP datagram 210 is 40 bytes including 20 bytes of IP header 220 information and 20 bytes of TCP header 230 information. The data portion or segment associated with the packet 210 follows the header information.")<br><br>Copeland at [0078] ("In regards to a typical IP packet 210, the first 4 bits of the IP header 220 identify the Internet protocol (IP) version. The following 4 bits identify the IP header length in 32 bit words. The next 8 bits differentiate the type of service by describing how the packet should be handled in transit. The following 16 bits convey the total packet length.")<br><br>Copeland at [0081] ("In a TCP/IP datagram 210, the initial data of the IP datagram is the TCP header 230 information. The initial TCP header 230 information includes the 16-bit source and 16-bit destination port numbers. A 32-bit sequence number for the data in the packet follows the port numbers. Following the sequence number is a 32-bit acknowledgement number. If an ACK flag (discussed below) is set, this number is the next sequence number the sender of the packet expects to receive. Next is a 4-bit data offset, which is the number of 32-bit words in the TCP header. A 6-bit reserved field follows.")<br><br>Copeland at [0082] ("Following the reserved field, the next 6 bits are a series of one-bit flags, shown in FIG. 2 as flags U, A, P, R, S, F. The first flag is the urgent flag (U). If the U flag is set, it indicates that the urgent pointer is valid and points to urgent data that should be acted upon as soon as possible. The next flag is the A ( or ACK or "acknowledgment") flag. The ACK flag indicates that an acknowledgment number is valid, and acknowledges that data has been received. The next flag, the push (P) flag, tells the receiving end to push all buffered data to the receiving application. The reset (R) flag is the following flag, which terminates both ends of the TCP connection. Next, the S (or SYN for "synchronize") flag is set in the initial packet of a TCP connection where both ends have to synchronize their TCP buffers. Following the SYN flag is the F (for FIN or "finish") flag. This flag signifies that the sending end of the communication and the host will not send any more data but still may acknowledge data that is received.")<br><br>Copeland at [0083] ("Following the TCP flag bits is a 16-bit receive window size field that specifies the amount of space avail-able in the receive buffer for the TCP connection. The |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 658 of 1100

| No. | '111 Patent Claim 16 | Lin '400 |
|---|---|---|
| | | checksum of the TCP header is a 16-bit field. Following the checksum is a 16 bit urgent pointer that points to the urgent data. The TCP/IP datagram data follows the TCP header.")<br><br>Copeland at [0116] ("These steps generally require manipulations of quantities such as IP addresses, packet length, header length, start times, end times, port numbers, and other packet related information. Usually, though not necessarily, these quanti-ties take the form of electrical, magnetic, or optical signals capable of being stored, transferred, combined, compared, or otherwise manipulated. It is conventional for those skilled in the art to refer to these signals as bits, bytes, words, values, elements, symbols, characters, terms, numbers, points, records, objects, images, files or the like. It should be kept in mind, however, that these and similar terms should be associated with appropriate quantities for computer opera-tions and that these terms are merely conventional labels applied to quantities that exist within and during operation of the computer.")<br><br>As another example, Kempf discloses packet headers with flag bits.<br><br>Kempf at [0081] ("In one embodiment, OpenFlow is modified to pro-vide rules for GTP TEID Routing. FIG. 17 is a diagram of one embodiment of the OpenFlow flow table modification for GTP TEID routing. An OpenFlow switch that supports TEID routing matches on the 2 byte (16 bit) collection of header fields and the 4 byte (32 bit) GTP TEID, in addition to other OpenFlow header fields, in at least one flow table ( e.g., the first flow table). The GTP TEID flag can be wildcarded (i.e. matches are "don't care"). In one embodiment, the EPC pro-tocols do not assign any meaning to TEIDs other than as an endpoint identifier for tunnels, like ports in standard UDP/ TCP transport protocols. In other embodiments, the TEIDs can have a correlated meaning or semantics. The GTP header flags field can also be wildcarded, this can be partially matched by combining the following bitmasks: 0xFF00- Match the Message Type field; 0xe0-Match the Version field; 0xl0-Match the PT field; 0x04-Match the E field; 0x02-Match the S field; and 0x0l-Match the PN field.")<br><br>Kempf at [0082] ("In one embodiment, OpenFlow can be modified to support virtual ports for fast path GTP TEID encapsulation and decapsulation. An OpenFlow mobile gateway can be used to support GTP encapsulation and decapsulation with virtual ports. The GTP encapsulation and decapsulation virtual ports can be used for fast encapsulation and decapsulation of user data packets within GTP-U tunnels, and can be designed simply enough |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 659 of 1100

| No. | '111 Patent Claim 16 | Lin '400 |
|---|---|---|
| | | that they can be implemented in hardware or firmware. For this reason, GTP virtual ports may have the following restrictions on traffic they will handle: Protocol Type (PT) field= 1, where GTP encapsulation ports only sup-port GTP, not GTP' (PT field=0); Extension Header flag (E)=0, where no extension headers are supported, Sequence Number flag (S)=0, where no sequence numbers are sup-ported; N-PDU flag (PN)=0; and Message type=255, where Only G-PDU messages, i.e. tunneled user data, is supported in the fast path.")<br><br>Kempf at [0083] ("If a packet either needs encapsulation or arrives encapsulated with nonzero header flags, header extensions, and/or the GTP-U packet is not a G-PDU packet (i.e. it is a GTP-U control packet), the processing must proceed via the gateway's slow path (software) control plane. GTP-C and GTP' packets directed to the gateway's IP address are a result of mis-configuration and are in error. They must be sent to the OpenFlow controller, since these packets are handled by the S-GW-C and P-GW-C control plane entities in the cloud computing system or to the billing entity handling GTP' and not the S-GW-D and P-GW-D data plane switches.")<br><br>Kempf at [0088] ("To support slow path encapsulation, the software control plane on the switch maintains a hash table with keys calculated from the GTP-U TEID. The TEID hash keys are calculated using a suitable hash algorithm with low collision frequency, for example SHA-1. The flow table entries contain a record of how the packet header, including the GTP encap-sulation header, should be configured. This includes: the same header fields as for the hardware or firmware encapsu-lation table in FIG.18; values for the GTP header flags (PT, E, S, and PN); the sequence number and/or the N-PDU number if any; if the E flag is 1, then the flow table contains a list of the extension headers, including their types, which the slow path should insert into the GTP header.")<br><br>Kempf at [0092] ("In one embodiment, the system implements a GTP fast path decapsulation virtual port. When requested by the S-GW and P-GW control plane software running in the cloud computing system, the gateway switch installs rules and actions for routing GTP encapsulated packets out of GTP tunnels. The rules match the GTP header flags and the GTP TEID for the packet, in the modified OpenFlow flow table shown in FIG. 17 as follows: the IP destination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); the UDP destination port is the GTP-U destination port (2152); and the header fields and message type field is wildcarded with the flag 0XFFF0 and the upper |

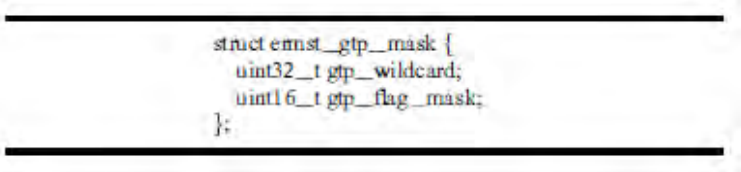


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 660 of 1100

| No. | '111 Patent Claim 16 | Lin '400 |
|---|---|---|
| | | two bytes of the field match the G-PDU message type (255) while the lower two bytes match 0x30, i.e. the packet is a GTP packet not a GTP' packet and the version number is 1.")<br><br>Kempf at [0094] ("In one embodiment, the system implements han-dling of GTP-U control packets. The OpenFlow controller programs the gateway switch flow tables with 5 rules for each gateway switch IP address used for GTP traffic. These rules contain specified values for the following fields: the IP des-tination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); the UDP destination port is the GTP-U destination port (2152); the GTP header flags and message type field is wildcarded with 0xFFF0; the value of the header flags field is 0x30, i.e. the version number is 1 and the PT field is 1; and the value of the message type field is one of 1 (Echo Request), 2 (Echo Response), 26 (Error Indication), 31 (Support for Extension Headers Notification), or 254 (End Marker).")<br><br>Kempf at [0098] ("The header flags and message type fields for the three rules are wildcarded with the following bitmasks and match as follows: bitmask 0xFFF4 and the upper two bytes match the G-PDU message type (255) while the lower two bytes are Ox34, indicating that the version number is 1, the packet is a GTP packet, and there is an extension header present; bitmask 0xFFFF2 and the upper two bytes match the G-PDU message type (255) while the lower two bytes are 0x32, indicating that the version number is 1, the packet is a GTP packet, and there is a sequence number present; and bitmask 0xFF0l and the upper two bytes match the G-PDU message type (255) while the lower two bytes are 0x31, indicating that the version number is 1, the packet is a GTP packet, and a N-PDU is present.")<br><br>Kempf at [0114] ("The gtp_type_n_flags field contains the GTP mes-sage type in the upper 8 bits and the GTP header flags in the lower 8 bits. The gtp_teid field contains the GTP TEID. The gtp_ wildcard field indicates whether the GTP type and flags and TEID should be matched. If the lower four bits are 1, the type and flags field should be ignored, while if the upper four bits are 1, the TEID should be ignored. If the lower bits are 0, the type and fields flag should be matched subject to the flags in the gtp_flag_mask field, while if the upper bits are 0 the TEID should be matched. The mask is combined with the message type and header field of the packet using logical AND; the result becomes the value of the match. Only those parts of the field in which the mask has a 1 value are matched.") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 661 of 1100

| No. | '111 Patent Claim 16 | Lin '400 |
|---|---|---|
| | | Kempf at [0117] ("The gtp_type_n_flags field contains the GTP mes-sage type in the upper 8 bits and the GTP header flags in the lower 8 bits. The gtp_<br>teid field contains the GRP TEID. When the value of the oxm_type ( oxm_class+oxm_field is GTP _ MATCH and the HM bit is zero, the flaw's GTP header must match these values exactly. If the HM flag is one, the value contains an ersmt_gtp_match field and an ermst_gtp_mask field, as specified by the OpenFlow 1.2 specification. We define ermst_gtp_mask field for selecting flows based on the settings of flag bits:<br>  <br>struct ermst_gtp_mask {<br>uint32_t gtp_wildcard;<br>uint16_t gtp_flag_mask;<br>};<br><br>Kempf at [0118] ("The gtp_ wildcard field indicates whether the TEID should be matched. If the value is 0xFFFFFFFF, the TEID should be matched and not the flags, if the value is 0x00000000, the flags should be matched and not the TEID. If the gtp_ wildcard indicates the flags should be matched, the gtp_flag_mask is combined with the message type and header field of the packet using logical AND, the result becomes the value of the match. Only those parts of the field in which the mask has a 1 value are matched.") |
| 16[c] | wherein the packet-applicable criterion is that one or more of the flag bits is set. | Lin '400 discloses wherein the packet-applicable criterion is that one or more of the flag bits is set.<br><br>For example, Lin '400 discloses flow rules that check the source address and destination address of a packet, which is part of a packet header. A person of ordinary skill in the art would understand that header fields can comprise one or more flag bits. A person of ordinary skill in the art would further understand that whether a condition is met in the flow table could depend on whether the one or more flag bits of a header is set. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Lin '400 is found to not meet this limitation, wherein the packet applicable criterion is that one or more of the flag bits is set would have been obvious to a person having ordinary skill in the art, as explained below. |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 662 of 1100

| No. | '111 Patent Claim 16 | Lin '400 |
|---|---|---|
| | | Lin '400 5:8-25 ("A flow table may include columns that indicate one or more conditions, a column that indicates an action to take when the conditions are met, and a column for statistics. A row on the flow table may comprise a flow rule. In the example of Table 1, the "Action" column indicates an action to take when conditions are met, and the "Count" column indicates statistics, such as byte count. The rest of the columns of Table 1 indicate conditions. For example, "IN_PORT", "MAC src" (media access control (MAC) address of the source of the packet), "MAC dst" (MAC address of the destination of the packet), "IP src" (Internet Protocol (IP) address of the source of the packet), "IP dst" (IP address of the destination of the packet), etc. are conditions that identify a particular packet. When the conditions are met, i.e., the particular packet is identified, the action indicated in the corresponding "Action" column is performed on the packet. The asterisks in Table 1 indicate an irrelevant condition.").<br><br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Lin '400 in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 16[c] of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references.<br><br>For example, Copeland discloses packet specific characteristics including flag bits that are set.<br><br>Copeland at [0081] ("In a TCP/IP datagram 210, the initial data of the IP datagram is the TCP header 230 information. The initial TCP header 230 information includes the 16-bit source and 16-bit destination port numbers. A 32-bit sequence number for the data in the packet follows the port numbers. Following the sequence number is a 32-bit acknowledgement number. If an ACK flag (discussed below) is set, this number is the next sequence number the sender of the packet expects to receive. Next is a 4-bit data offset, which is the number of 32-bit words in the TCP header. A 6-bit reserved field follows.")<br><br>Copeland at [0082] ("Following the reserved field, the next 6 bits are a series of one-bit flags, shown in FIG. 2 as flags U, A, P, R, S, F. The first flag is the urgent flag (U). If the U flag is set, it indicates that the urgent pointer is valid and points to urgent data that should be acted upon as soon as possible. The next flag is the A ( or ACK or "acknowledgment") flag. The |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 663 of 1100

| No. | '111 Patent Claim 16 | Lin '400 |
|---|---|---|
| | | ACK flag indicates that an acknowledgment number is valid, and acknowledges that data has been received. The next flag, the push (P) flag, tells the receiving end to push all buffered data to the receiving application. The reset (R) flag is the following flag, which terminates both ends of the TCP connection. Next, the S (or SYN for "synchronize") flag is set in the initial packet of a TCP connection where both ends have to synchronize their TCP buffers. Following the SYN flag is the F (for FIN or "finish") flag. This flag signifies that the sending end of the communication and the host will not send any more data but still may acknowledge data that is received.")<br><br>Copeland at [0083] ("Following the TCP flag bits is a 16-bit receive window size field that specifies the amount of space avail-able in the receive buffer for the TCP connection. The checksum of the TCP header is a 16-bit field. Following the checksum is a 16 bit urgent pointer that points to the urgent data. The TCP/IP datagram data follows the TCP header.")<br><br>Copeland at [0089] ("FIG. 3 illustrates an exemplary TCP/IP session 300. As discussed in reference to FIG. 2, the SYN flag is set whenever one host initiates a session with another host. In the initial packet, Hostl sends a message with only the SYN flag set. The SYN flag is designed to establish a TCP connection and allow both ends to synchronize their TCP buffers. Hostl provides the sequence of the first data packet it will send.")<br><br>Copeland at [0125] ("For purposes of the description, which follows, the IP address with the lower value, when considered as a 32-bit unsigned integer, is designated ip[0] and the corresponding port number is designated pt[0]. The higher IP address is designated ip[l] and the corresponding TCP or UDP port number is designated pt[l]. At some point, either pt[0] or pt[l] may be designated the "server" port by setting an appropriate bit in a bit map that is part of the flow record (record "state", bit 1 or 2 is set).")<br><br>Copeland at [0145] ("A list IP of addresses contacted or probed by each host can be maintained. When this list indicates that more than a threshold number of other hosts (e.g., 8) have been contacted in the same subnet, CI is added to the to the host and a bit in the host record is set to indicate that the host has received CI for "address scanning." Note that the number of hosts to designate a scan is not required to be a fixed value, but could be adjusted based on the sample rate or other means to enhance the accuracy making the number of hosts |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 664 of 1100

| No. | ’111 Patent Claim 16 | Lin ’400 |
|---|---|---|
| | | scanned "statistically significant". These and other values of concern index are shown for non-flow based events in FIG. 7.”)<br><br>As another example, Kempf flow table matches in which the flag bits is set,<br><br>Kempf at [0081] (“In one embodiment, OpenFlow is modified to pro-vide rules for GTP TEID Routing. FIG. 17 is a diagram of one embodiment of the OpenFlow flow table modification for GTP TEID routing. An OpenFlow switch that supports TEID routing matches on the 2 byte (16 bit) collection of header fields and the 4 byte (32 bit) GTP TEID, in addition to other OpenFlow header fields, in at least one flow table ( e.g., the first flow table). The GTP TEID flag can be wildcarded (i.e. matches are "don't care"). In one embodiment, the EPC pro-tocols do not assign any meaning to TEIDs other than as an endpoint identifier for tunnels, like ports in standard UDP/ TCP transport protocols. In other embodiments, the TEIDs can have a correlated meaning or semantics. The GTP header flags field can also be wildcarded, this can be partially matched by combining the following bitmasks: 0xFF00- Match the Message Type field; 0xe0-Match the Version field; 0xl0-Match the PT field; 0x04-Match the E field; 0x02-Match the S field; and 0x0l-Match the PN field.”)<br><br>Kempf at [0082] (“In one embodiment, OpenFlow can be modified to support virtual ports for fast path GTP TEID encapsulation and decapsulation. An OpenFlow mobile gateway can be used to support GTP encapsulation and decapsulation with virtual ports. The GTP encapsulation and decapsulation virtual ports can be used for fast encapsulation and decapsulation of user data packets within GTP-U tunnels, and can be designed simply enough that they can be implemented in hardware or firmware. For this reason, GTP virtual ports may have the following restrictions on traffic they will handle: Protocol Type (PT) field= 1, where GTP encapsulation ports only sup-port GTP, not GTP' (PT field=0); Extension Header flag (E)=0, where no extension headers are supported, Sequence Number flag (S)=0, where no sequence numbers are sup-ported; N-PDU flag (PN)=0; and Message type=255, where Only G-PDU messages, i.e. tunneled user data, is supported in the fast path.”)<br><br>Kempf at [0083] (“If a packet either needs encapsulation or arrives encapsulated with nonzero header flags, header extensions, and/or the GTP-U packet is not a G-PDU packet (i.e. it is a GTP-U control packet), the processing must proceed via the gateway's slow path (software) control plane. GTP-C and GTP' packets directed to the gateway's IP address are a result of |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 665 of 1100

| No. | '111 Patent Claim 16 | Lin '400 |
|---|---|---|
| | | mis-configuration and are in error. They must be sent to the OpenFlow controller, since these packets are handled by the S-GW-C and P-GW-C control plane entities in the cloud computing system or to the billing entity handling GTP' and not the S-GW-D and P-GW-D data plane switches.")<br><br>Kempf at [0088] ("To support slow path encapsulation, the software control plane on the switch maintains a hash table with keys calculated from the GTP-U TEID. The TEID hash keys are calculated using a suitable hash algorithm with low collision frequency, for example SHA-1. The flow table entries contain a record of how the packet header, including the GTP encap-sulation header, should be configured. This includes: the same header fields as for the hardware or firmware encapsu-lation table in FIG.18; values for the GTP header flags (PT, E, S, and PN); the sequence number and/or the N-PDU number if any; if the E flag is 1, then the flow table contains a list of the extension headers, including their types, which the slow path should insert into the GTP header.")<br><br>Kempf at [0092] ("In one embodiment, the system implements a GTP fast path decapsulation virtual port. When requested by the S-GW and P-GW control plane software running in the cloud computing system, the gateway switch installs rules and actions for routing GTP encapsulated packets out of GTP tunnels. The rules match the GTP header flags and the GTP TEID for the packet, in the modified OpenFlow flow table shown in FIG. 17 as follows: the IP destination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); the UDP destination port is the GTP-U destination port (2152); and the header fields and message type field is wildcarded with the flag 0XFFF0 and the upper two bytes of the field match the G-PDU message type (255) while the lower two bytes match 0x30, i.e. the packet is a GTP packet not a GTP' packet and the version number is 1.")<br><br>Kempf at [0094] ("In one embodiment, the system implements han-dling of GTP-U control packets. The OpenFlow controller programs the gateway switch flow tables with 5 rules for each gateway switch IP address used for GTP traffic. These rules contain specified values for the following fields: the IP des-tination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); the UDP destination port is the GTP-U destination port (2152); the GTP header flags and message type field is wildcarded with 0xFFF0; the value of the header flags field is 0x30, i.e. the version number is 1 and the PT field is 1; and the value of the message type field is one of 1 (Echo Request), 2 (Echo |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 666 of 1100

| No. | '111 Patent Claim 16 | Lin '400 |
|---|---|---|
| | | Response), 26 (Error Indication), 31 (Support for Extension Headers Notification), or 254 (End Marker).")<br><br>Kempf at [0098] ("The header flags and message type fields for the three rules are wildcarded with the following bitmasks and match as follows: bitmask 0xFFF4 and the upper two bytes match the G-PDU message type (255) while the lower two bytes are Ox34, indicating that the version number is 1, the packet is a GTP packet, and there is an extension header present; bitmask 0xFFFF2 and the upper two bytes match the G-PDU message type (255) while the lower two bytes are 0x32, indicating that the version number is 1, the packet is a GTP packet, and there is a sequence number present; and bitmask 0xFF0l and the upper two bytes match the G-PDU message type (255) while the lower two bytes are 0x31, indicating that the version number is 1, the packet is a GTP packet, and a N-PDU is present.")<br><br>Kempf at [0114] ("The gtp_type_n_flags field contains the GTP mes-sage type in the upper 8 bits and the GTP header flags in the lower 8 bits. The gtp_teid field contains the GTP TEID. The gtp_ wildcard field indicates whether the GTP type and flags and TEID should be matched. If the lower four bits are 1, the type and flags field should be ignored, while if the upper four bits are 1, the TEID should be ignored. If the lower bits are 0, the type and fields flag should be matched subject to the flags in the gtp_flag_mask field, while if the upper bits are 0 the TEID should be matched. The mask is combined with the message type and header field of the packet using logical AND; the result becomes the value of the match. Only those parts of the field in which the mask has a 1 value are matched.")<br><br>Kempf at [0117] ("The gtp_type_n_flags field contains the GTP mes-sage type in the upper 8 bits and the GTP header flags in the lower 8 bits. The gtp_teid field contains the GRP TEID. When the value of the oxm_type ( oxm_class+oxm_field is GTP _ MATCH and the HM bit is zero, the flaw's GTP header must match these values exactly. If the HM flag is one, the value contains an ersmt_gtp_match field and an ermst_gtp_mask field, as specified by the OpenF!ow 1.2 specification. We define ermst_gtp_mask field for selecting flows based on the settings of flag bits: |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 667 of 1100

| No. | '111 Patent Claim 16 | Lin '400 |
|---|---|---|

```
struct ernst_gtp_mask {
  uint32_t gtp_wildcard;
  uint16_t gtp_flag_mask;
};
```

Kempf at [0118] ("The gtp_ wildcard field indicates whether the TEID should be matched. If the value is 0xFFFFFFFF, the TEID should be matched and not the flags, if the value is 0x00000000, the flags should be matched and not the TEID. If the gtp_ wildcard indicates the flags should be matched, the gtp_flag_mask is combined with the message type and header field of the packet using logical AND, the result becomes the value of the match. Only those parts of the field in which the mask has a 1 value are matched.")

Kempf at Figure 10

| Octets | Bits 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
|---|---|---|---|---|---|---|---|---|
| 1 | Version | | | PT | (*) | E | S | PN |
| 2 | Message Type | | | | | | | |
| 3 | Length (1st Octet) | | | | | | | |
| 4 | Length (2nd Octet) | | | | | | | |
| 5 | Tunnel Endpoint Identifier (1st Octet) | | | | | | | |
| 6 | Tunnel Endpoint Identifier (2nd Octet) | | | | | | | |
| 7 | Tunnel Endpoint Identifier (3rd Octet) | | | | | | | |
| 8 | Tunnel Endpoint Identifier (4th Octet) | | | | | | | |
| 9 | Sequence Number (1st Octet) | | | | | | | |
| 10 | Sequence Number (2nd Octet) | | | | | | | |
| 11 | N-PDU Number | | | | | | | |
| 12 | Next Extension Header Type | | | | | | | |

NOTE 0: (*) This bit is a spare bit. It shall be sent as '0'. The receiver shl not evaluate this bit,
NOTE 1: 1) This field shall only be evaluated when indicated by the S flag set to 1.
NOTE 2: 2) This field shall only be evaluated when indicated by the PN flag set to 1.
NOTE 3: 3) This field shall only be evaluated when indicated by the E flag set to 1.
NOTE 4: 4) This field shall be present if and only if any one or more of the S. PN and E flags are set.

FIG. 10



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 668 of 1100

| No. | '111 Patent Claim 16 | Lin '400 |
|---|---|---|
| | | |

| No. | '111 Patent Claim 17 | Lin '400 |
|---|---|---|
| 17[a] | The method according to claim 16, wherein the packet is an Transmission Control Protocol (TCP) packet, and | Lin '400 discloses the method according to claim 16, wherein the packet is a Transmission Control Protocol (TCP) packet.<br><br>For example, Lin '400 discloses TCP packets entering and exiting the switch. Lin '400 further discloses a transport protocol (TCP) port.<br><br>TABLE 2<br><br>IN_PORT . . . IP src **TCP** src port **TCP** dst port . . . Action Count<br><br>Ingress_port_ID * * * 80 * Egress port 120<br>Egress_port_ID * * 80 * * Ingress port 120<br>Ingress_port_ID * * * * * Redirect port 10<br>Redirect_port_ID * * * * * Ingress port 10<br><br>Lin '400 7:39-50 ("In the example of Table 2, the first two rows are bypass rules for bypassing packets coming from or going to a transport control protocol (TCP) port 80. More specifically, hypertext transfer protocol (HTTP) packets, i.e., port 80 packets, that are received in the ingress port with the Ingress_port_ID (i.e., ingress port 623-1) are forwarded directly to the egress port (i.e., egress port 623-4), instead of being redirected to the redirect port 623-2 for inspection by the security service 630. Similarly, HTTP packets received in the egress port with the Egress_port_ID (i.e., egress port 623-4) are forwarded directly to the ingress port 623-1 without being redirected to the security service 630."). |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 669 of 1100

| No. | '111 Patent Claim 17 | Lin '400 |
|---|---|---|
| | | TABLE 3<br><br>IN_PORT . . . IP src | **TCP** src port | **TCP** dst port . . . Action Count<br><br>Ingress_port_ID * * * 80 * Redirect port 10<br>Redirect_port_ID * * 80 * * Ingress port 10<br>Ingress_port_ID * * * * * Egress port 130<br>Egress_port_ID * * * * * Ingress port 130<br><br>Lin '400 8:10-18 ("In the example of Table 3, the top two rows are redirect flow rules for redirecting HTTP packets to the security service 630 for inspection, while the bottom two rows are bypass flow rules for all packets. Because the redirect flow rules are at higher priority than the bypass flow rules, HTTP packets are sent through the SDN pipe formed in the SDN switch 620 between the sender component 622 and the security service 630. All other packets bypass the SDN pipe, and are accordingly not inspected by the security service 630.<br><br>Lin '400 Claim 8 ("The SDN computer network of claim 7, wherein the specified packets are packets having a particular transport control protocol (TCP) source or destination port."). |
| 17[b] | wherein the one or more flag bits comprises comprise a SYN flag bit, an ACK flag bit, a FIN flag bit, a RST flag bit, or any combination thereof. | Lin '400 discloses wherein the one or more flag bits comprises comprise a SYN flag bit, an ACK flag bit, a FIN flag bit, a RST flag bit, or any combination thereof.<br><br>For example, Lin '400 discloses flow rules that check the source address and destination address of a packet, which is part of a packet header. A person of ordinary skill in the art would understand that header fields can comprise one or more flag bits that can comprise a SYN flag bit, an ACK flag bit, a FIN flag bit, a RST flag bit, or any combination thereof. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Lin '400 is found to not meet this limitation, wherein the one or more flag bits comprises comprise a SYN flag bit, an ACK flag bit, a FIN flag bit, a RST flag bit, or any combination thereof would have been obvious to a person having ordinary skill in the art, as explained below. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 670 of 1100

| No. | '111 Patent Claim 17 | Lin '400 |
|---|---|---|
| | | Lin '400 5:8-25 ("A flow table may include columns that indicate one or more conditions, a column that indicates an action to take when the conditions are met, and a column for statistics. A row on the flow table may comprise a flow rule. In the example of Table 1, the "Action" column indicates an action to take when conditions are met, and the "Count" column indicates statistics, such as byte count. The rest of the columns of Table 1 indicate conditions. For example, "IN_PORT", "MAC src" (media access control (MAC) address of the source of the packet), "MAC dst" (MAC address of the destination of the packet), "IP src" (Internet Protocol (IP) address of the source of the packet), "IP dst" (IP address of the destination of the packet), etc. are conditions that identify a particular packet. When the conditions are met, i.e., the particular packet is identified, the action indicated in the corresponding "Action" column is performed on the packet. The asterisks in Table 1 indicate an irrelevant condition."). |
| | | Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Lin '400 in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 17[b] of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references. |
| | | For example, Copeland discloses TCP packets with flag bits including SYN, ACK, FIN, and R flag bits. |
| | | Copeland at [0081] ("In a TCP/IP datagram 210, the initial data of the IP datagram is the TCP header 230 information. The initial TCP header 230 information includes the 16-bit source and 16-bit destination port numbers. A 32-bit sequence number for the data in the packet follows the port numbers. Following the sequence number is a 32-bit acknowledgement number. If an ACK flag (discussed below) is set, this number is the next sequence number the sender of the packet expects to receive. Next is a 4-bit data offset, which is the number of 32-bit words in the TCP header. A 6-bit reserved field follows.") |
| | | Copeland at [0082] ("Following the reserved field, the next 6 bits are a series of one-bit flags, shown in FIG. 2 as flags U, A, P, R, S, F. The first flag is the urgent flag (U). If the U flag is set, it indicates that the urgent pointer is valid and points to urgent data that should be acted |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 671 of 1100

| No. | ’111 Patent Claim 17 | Lin ’400 |
|---|---|---|
| | | upon as soon as possible. The next flag is the A ( or ACK or "acknowledgment") flag. The ACK flag indicates that an acknowledgment number is valid, and acknowledges that data has been received. The next flag, the push (P) flag, tells the receiving end to push all buffered data to the receiving application. The reset (R) flag is the following flag, which terminates both ends of the TCP connection. Next, the S (or SYN for "synchronize") flag is set in the initial packet of a TCP connection where both ends have to synchronize their TCP buffers. Following the SYN flag is the F (for FIN or "finish") flag. This flag signifies that the sending end of the communication and the host will not send any more data but still may acknowledge data that is received.”)<br><br>Copeland at [0089] (“FIG. 3 illustrates an exemplary TCP/IP session 300. As discussed in reference to FIG. 2, the SYN flag is set whenever one host initiates a session with another host. In the initial packet, Hostl sends a message with only the SYN flag set. The SYN flag is designed to establish a TCP connection and allow both ends to synchronize their TCP buffers. Hostl provides the sequence of the first data packet it will send.”)<br><br>Copeland at [0090] (“Host2 responds with a SYN-ACK packet. In this message, both the SYN flag and the ACK flag are set. Host2 provides the initial sequence number for its data to Hostl. Host2 also sends to Hostl the acknowledgment number that is the next sequence number Host2 expects to receive from host 1. In the SYN-ACK packet sent by Host2, the acknowl-edgment number is the initial sequence number of Hostl plus 1, which should be the next sequence number received.”)<br><br>Copeland at [0091] (“Hostl responds to the SYN-ACK with a packet with just the ACK flag set. Hostl acknowledges that the next packet of information received from Host2 will be Host2's initial sequence number plus 1. The three-way handshake is complete and data is transferred.”)<br><br>Copeland at [0092] (“Host2 responds to ACK packet with its own ACK packet. Host2 acknowledges the data it has received from Hostl by sending an acknowledgment number one greater than its last received data sequence number. Both hosts send packets with the ACK flag set until the session is to end although the P and U flags may also be set, if warranted.”) |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 672 of 1100

| No. | '111 Patent Claim 17 | Lin '400 |
|---|---|---|
| | | Copeland at [0093] ("As illustrated, when Hostl terminates its end of the session, it sends a packet with the FIN and ACK flags set. The FIN flag informs Host2 that Hostl will send no more data. The ACK flag acknowledges the last data received by Hostl by informing Host2 of the next sequence number it expects to receive.")<br><br>Copeland at [0094] ("Host2 acknowledges the FIN packet by sending its own ACK packet. The ACK packet has the acknowledge-ment number one greater than the sequence number of Hostl's FIN-ACK packet. ACK packets are still delivered between the two hosts, except that HOSTl's packets have no data appended to the TCP/IP end of the headers.")<br><br>Copeland at [0095] ("When Host 2 is ready to terminate the session, it sends its own packet with the FIN and ACK flags set. Hostl responds that it has received the final packet with an ACK packet providing to Host2 an acknowledgment number one greater than the sequence number provided in the FIN-ACK packet of Host2.")<br><br>As another example, Uchida discloses the TCP (Transmission Control Protocol) FIN flag, RST flag, and SYN flag.<br><br>Uchida at [0040] ("A flow end can be detected by various methods as below. For example, in one method, a protocol end message is checked. For example, in the TCP (Transmission Control Protocol), a FIN flag is checked. In this way, the end of communication, that is, the end of a flow using communica-tion, can be detected. In practice, after a FIN flag, communi-cation with an ACK packet is generated in a reverse-direction flow (a flow in which the source and the destination are reversed). Thus, by detecting the ACK flag in the reverse-direction flow after the FIN packet, a flow end can be deter-mined. Further, since the TCP is used in bidirectional com-munication, the forward- and reverse-direction flows can be used as a pair to determine a flow end. Namely, if the end of a flow is detected, a process rule corresponding to the reverse-direction flow of the flow can also be determined to be unnec-essary. Alternatively, a communication end can also be deter-mined when a predetermined time elapses after reception of a SYN packet and a timeout is determined. Still alternatively, a communication end can be determined by reception of a RST packet. These methods will be described in more detail later as specific examples.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 673 of 1100

| No. | '111 Patent Claim 17 | Lin '400 |
|---|---|---|
| | | Uchida at [0050] ("The flow end check unit can use at least one of a TCP (Transmission Control Protocol) FIN flag, RST flag, and SYN flag extracted by the end determination information extraction unit to determine a flow end.")<br><br>Uchida at [0055] ("In the process rule update method, a flow end can be determined by at least one of a TCP (Transmission Control Protocol) FIN flag, RST flag, and SYN flag.")<br><br>Uchida at [0102] ("Next, specific examples 1 to 3 will be described. In the examples 1 to 3, a flow end is determined by combining features of the above individual exemplary embodiments and using TCP (Transmission Control Protocol) flags.")<br><br>Uchida at [0103] ("FIG. 6 is a state transition diagram of TCP connec-tion. "CLOSED" at the top of FIG. 6 represents the end of TCP communication, and portions connected thereto repre-sent states prior to the end of TCP communication. Approxi-mately 2MSL (MSL: Maximum Segment Lifetime) is the maximum amount of time required to reach the above "CLOSED," that is, if the packet forwarding apparatus stands by for approximately 2MSL after both FINs flow, the above "CLOSED" is reached. Thus, after a FIN is confirmed in either direction, if this 2MSL elapses, basically, a communi-cation end can be determined. Even if the state does not change smoothly because of packet loss or the like (for example, even if an ACK packet does not arrive after "CLOS-ING"), a retransmitted packet is forwarded immediately after this 2MSL. Thus, the end of TCP communication can be determined if a new FIN packet is not received within the time corresponding to the 2MSL and a margin (2MSL+a) at long-est.")<br><br>Uchida at [0104] ("Hereinafter, the description will be made, assuming that a packet forwarding apparatus Cl according to the present invention relays TCP communication between a com-puter (client) Dl 0 and a server D20 that use network configu-rations illustrated in FIG. 7. In the example of FIG. 7, the computer Dl0 belongs to a network represented by 192.168. 0./24 and is set by 192.168.0.10. The server D20 belongs to a network represented by 192.168.1./24 and is set by 192.168. 1.10. As in the case of the OpenFlow controller described in Non-Patent Documents 1 and 2, a control apparatus ( control-ler) Dl is connected to the packet forwarding apparatus Cl via a dedicated channel and manages connection between the two networks. In the following description, the control appa-ratus (controller) Dl controls the packet forwarding appara-tus Cl so that connection from other |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 674 of 1100

| No. | '111 Patent Claim 17 | Lin '400 |
|---|---|---|
| | | networks appears as communication from network number 1 (192.168.1.1) of the respective networks (see process rule actions in FIG. 19). In addition, in the present specific example, since FIN packets are monitored, the end determination information extraction unit Cl 7 monitors a protocol stack, including: fields in which the TCP is determined; and the FIN flag in the TCP header.")<br><br>Uchida at [0105] ("FIG. 8 is a flow chart of a flow end determination process using FIN flags. In FIG. 8, steps relating to a timeout determination are added to steps Slll to S116 in the flow chart in FIG. 3. Thus, the flow chart in FIG. 8 includes more detailed steps than the flow chart of FIG. 3. Hereinafter, operations will be described with reference to FIGS. 3, 6, and 8 and FIGS. 9 to 13. In practice, prior to TCP/IP communi-cation, ARP (Address Resolution Protocol) communication is executed, and a process rule may be set in that stage. However, for ease of description, description of the ARP communication will be omitted. The following description will be made based on communication at the TCP/IP level.")<br><br>Uchida at [0106] ("First, the computer Dl0 starts communication with the server D20. For an initial establishment of communica-tion, a packet (SYN) is inputted to the packet forwarding apparatus Cl (start of ACTIVE OPEN through SYN forward-ing in FIG. 6). The packet reception unit Cl0 receives and stores this first packet in the packet storage unit Cll (steps SlOl to S102 in FIG. 3).")<br><br>Uchida at [0107] ("The packet reception unit C10 notifies the packet process information extraction unit C12 and the end determination information extraction unit C17 of reception of the packet. The packet process information extraction unit C12 refers to the packet storage unit C11 and extracts information such as IP source and destination information that is necessary to search for a process rule (step S103 in FIG. 3). Hereinafter, a process corresponding to steps S103 to S110 in FIG. 3 will be executed.")<br><br>Uchida at [0115] ("Upon receiving a notification that the packet has been received by the packet reception unit Cl 0, the end deter-mination information extraction unit Cl 7 refers to the packet storage unit Cll, monitors a TCP FIN flag, and finds a FIN flag (step S201 in FIG. 8).") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 675 of 1100

| No. | '111 Patent Claim 17 | Lin '400 |
|---|---|---|
| | | Uchida at [0116] ("Since a FIN flag is set, the end determination infor-mation extraction unit Cl 7 determines that the packet includes information necessary for determining a flow end. Thus, the end determination information extraction unit Cl 7 extracts information for identifying a process rule to be deleted (the ingress port is 1; the source address is 192.168. 0.10; the destination is 192.168.1.10; and the protocol is TCP (the type is Ox0006)) and stands by until forwarding of the packet. Upon receiving a notification that the packet has been transmitted by the packet forwarding unit C16, the end deter-mination information extraction unit Cl 7 further extracts information for identifying a process rule to be deleted from the packet storage unit Cll. Since the IP address is replaced, the extracted information for identifying a process rule to be deleted represents that the source address is 192.168.1.1; the destination is 192.168.1.1 0; and the protocol is TCP (the type is 0x0006). The information is used for marking of the reverse flow. The end determination information extraction unit Cl 7 notifies the flow end check unit C18 of the notification that the FIN packet has been received and these items of information (step S202 in FIG. 8).")<br><br>Uchida at [0117] ("Upon receiving the above information from the end determination information extraction unit Cl 7, the flow end check unit C18 checks whether or not a FIN flag is set in a predetermined packet header position (step S203). These steps correspond to steps Slll to S114 in FIG. 3.")<br><br>Uchida at [0121] ("Next, after an ACK reply in response to the FIN packet from the computer DlO is forwarded from the server D20 in the same way as the above normal packet (start of PASSIVE CLOSE in FIG. 6), the server D20 transmits a FIN packet to the computer DlO. When this FIN packet is inputted to the packet forwarding apparatus Cl, the flow end determi-nation process from steps Slll to S116 is started, as in the case of the above start of ACTIVE CLOSE.")<br><br>Uchida at [0122] ("Upon receiving a notification that the packet has been received from the packet reception unit Cl0, the end determination information extraction unit Cl 7 refers to the packet storage unit Cll, monitors a TCP FIN flag, and finds a FIN packet (step S201 in FIG. 8).")<br><br>Uchida at [0123] ("Since a FIN flag is set, the end determination infor-mation extraction unit Cl 7 determines that the packet includes information necessary for determining a flow end. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 676 of 1100

| No. | '111 Patent Claim 17 | Lin '400 |
|---|---|---|
| | | Thus, the end determination information extraction unit Cl 7 extracts information for identifying a process rule to be deleted (the ingress port is 2; the source address is 192.168. 1.10; the destination is 192.168.1.1; and the protocol is TCP (the type is Ox.0006)) and stands by until the packet is trans-mitted. Upon receiving a notification that the packet has been transmitted from the packet forwarding unit C16, the end determination information extraction unit Cl 7 further extracts information for identifying a modified process rule from the packet storage unit Cll. Since the IP address is replaced, the extracted information for identifying a modified process rule represents that the source address is 192.168.1. 10; the destination is 192.168.0.10; and the protocol is TCP (the type is 0x0006). The information is used for marking of the reverse flow. The end determination information extrac-tion unit Cl 7 notifies the flow end check unit C18 of the notification that the FIN packet has been received and these items of information (step S202 in FIG. 8).")<br><br>Uchida at [0124] ("Upon receiving the above information from the end determination information extraction unit Cl 7, the flow end check unit C18 checks whether or not a FIN flag is set in a predetermined packet header position (step S203 in FIG. 8). These steps correspond to steps Slll to S114 in FIG. 3.")<br><br>Uchida at [0125] ("At this point, since a FIN packet has been transmit-ted, the flow end check unit C18 uses the information for identifying a process rule to be deleted as a key, extracts the process rule (process rule corresponding to ingress port 2 in FIG. 11) from the process rule storage unit C13, and marks a FIN packet reception flag (steps S204 to S205 in FIG. 8). This process corresponds to the internal state update process in step S115 in FIG. 3.")<br><br>Uchida at [0134] ("Referring back to the state transition diagram of TCP connection in FIG. 6, there are two cases where "CLOSED" at the top of FIG. 6 is reached without a state transition involving FIN flags. One case arises when the ses-sion is closed from SYN_SENT, which is reached when a SYN packet in which a SYN flag is marked is transmitted. The other case arises when a timeout is generated. In such case, while the packet forwarding apparatus cannot monitor the closed session, the packet forwarding apparatus can con-firm a timeout in the following way. In the present specific example, a flow end is determined by this timeout.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 677 of 1100

| No. | '111 Patent Claim 17 | Lin '400 |
|---|---|---|
| | | Uchida at [0135] ("n the present specific example, if a SYN/ ACK packet does not flow in a direction opposite to the SYN packet flow direction within a predetermined time (from "SYN_ RCVD" to "SYN_SENT" in FIG. 6), a timeout is determined.")<br><br>Uchida at [0136] ("FIG. 14 is a flow chart illustrating a flow end deter-mination process using a SYN flag. Since the basic operations are the same as those of the above specific example 1, the following description will be made with a focus on the dif-ference.")<br><br>Uchida at [0137] ("In FIG. 14, upon receiving a notification that the packet has been received by the packet reception unit ClO, the end determination information extraction unit Cl 7 refers to the packet storage, unit Cll, monitors a TCP SYN flag, and finds a SYN packet (step S301 in FIG. 14).")<br><br>Uchida at [0138] ("Since a SYN flag is set, the end determination infor-mation extraction unit Cl 7 determines that the packet includes information necessary for determining a flow end. Thus, the end determination information extraction unit Cl 7 extracts information for identifying a process rule to be deleted (the ingress port is 2; the source address is 192.168. 1.10; the destination is 192.168.1.1; and the protocol is TCP (the type is Ox.0006)) and stands by until the packet is trans-mitted. Upon receiving a notification that the packet has been transmitted by the packet forwarding unit C16, the end deter-mination information extraction unit Cl 7 further extracts information for identifying a modified process rule from the packet storage unit Cll. Since the IP address is replaced, the extracted information for identifying a process rule repre-sents that the source address is 192.168.1.10; the destination is 192.168.0.10; and the protocol is TCP (the type is 0x0006). The information is used for marking of the reverse flow. The end determination information extraction unit Cl 7 notifies the flow end check unit C18 of the notification that the SYN packet has been received and these items of information (step S302 in FIG. 14).")<br><br>Uchida at [0139] ("Upon receiving the above information from the end determination information extraction unit Cl 7, the flow end check unit C18 checks whether a SYN flag is set in a prede-termined packet header position and an ACK flag is not marked (step S303 in FIG. 14). These steps correspond to steps Slll to S114 in FIG. 3.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 678 of 1100

| No. | '111 Patent Claim 17 | Lin '400 |
|---|---|---|
| | | Uchida at [0148] (" Next, a third specific example in which a flow end determination is executed by using a TCP RST (reset) flag will be described.")<br><br>Uchida at [0149] ("Referring back to the state transition diagram of TCP connection in FIG. 6, there is a transition from "SYN_ RCVD," which is a communication establishment standby state, to "LISTEN," which is a communication standby state. A TCP RST (reset) flag signifies release of connection and retry of communication. Namely, since a RST packet in which this RST flag is set signifies invalidation of communi-cation, by detecting this RST flag, a flow end can be deter-mined.")<br><br>Uchida at [0150] ("FIG. 16 is a first flow chart illustrating a flow end determination process using a RST flag. Since the basic operations are the same as those of the above specific example 1, the following description will be made with a focus on the difference.")<br><br>Uchida at [0151] ("In FIG. 16, upon receiving a notification that the packet has been received by the packet reception unit ClO, the end determination information extraction unit Cl 7 refers to the packet storage unit Cll, monitors a TCP RST flag, and finds a RST packet (step S401 in FIG. 16).")<br><br>Uchida at [0152] ("Since a RST flag is set, the end determination infor-mation extraction unit Cl 7 determines that the packet includes information necessary for determining a flow end. Thus, the end determination information extraction unit Cl 7 extracts information for identifying a process rule to be deleted (the ingress port is 2; the source address is 192.168. 1.10; the destination is 192.168.1.1; and the protocol is TCP (the type is Ox0006)) and stands by until the packet is trans-mitted. Upon receiving a notification that the packet has been transmitted from the packet forwarding unit C16, the end determination information extraction unit Cl 7 notifies the flow end check unit C18 of the notification that the RST packet has been received and these items of information ( step S402 in FIG. 16).")<br><br>Uchida at [0164] ("For example, in a specific example of the present invention, certain TCP flags are monitored. A single packet forwarding apparatus can monitor these flags in a parallel fashion. For example, after a packet that triggers a flow end is detected, the above process may be allowed to branch to the above FIGS. 8, 14, and 16 (17) to realize parallel monitoring.") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 679 of 1100

| No. | '111 Patent Claim 17 | Lin '400 |
|---|---|---|
| | | |

| No. | '111 Patent Claim 18 | Lin '400 |
|---|---|---|
| 18[a] | The method according to claim 1, wherein the packet comprises distinct header and payload fields, | Lin '400 discloses the method according to claim 1, wherein the packet comprises distinct header and payload fields.<br><br>*See supra* Claim 1, 15[a]. |
| 18[b] | the header comprises at least the first and second entities addresses in the packet network, and | Lin '400 discloses the header comprises at least the first and second entities addresses in the packet network.<br><br>For example, Lin '400 discloses flow rules that check the source address and destination address of a packet, which is part of a packet header. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met.<br><br>Lin '400 5:8-25 ("A flow table may include columns that indicate one or more conditions, a column that indicates an action to take when the conditions are met, and a column for statistics. A row on the flow table may comprise a flow rule. In the example of Table 1, the "Action" column indicates an action to take when conditions are met, and the "Count" column indicates statistics, such as byte count. The rest of the columns of Table 1 indicate conditions. For example, "IN_PORT", "MAC src" (media access control (MAC) address of the source of the packet), "MAC dst" (MAC address of the destination of the packet), "IP src" (Internet Protocol (IP) address of the source of the packet), "IP dst" (IP address of the destination of the packet), etc. are conditions that identify a particular packet. When the conditions are met, i.e., the particular packet is identified, the action indicated in the corresponding "Action" column is performed on the packet. The asterisks in Table 1 indicate an irrelevant condition."). |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 680 of 1100

| No. | '111 Patent Claim 18 | Lin '400 |
|---|---|---|
| 18[c] | wherein the packet-applicable criterion is that the first entity address, the second entity address, or both match a predetermined address or addresses. | Lin '400 discloses wherein the packet-applicable criterion is that the first entity address, the second entity address, or both match a predetermined address or addresses.<br><br>For example, Lin '400 discloses flow rules that check the source address and destination address of a packet, which is part of a packet header. Lin '400 further discloses using source and destination addresses of a packet header to determine whether a packet meets a condition in a flow table. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met.<br><br>Lin '400 5:8-25 ("A flow table may include columns that indicate one or more conditions, a column that indicates an action to take when the conditions are met, and a column for statistics. A row on the flow table may comprise a flow rule. In the example of Table 1, the "Action" column indicates an action to take when conditions are met, and the "Count" column indicates statistics, such as byte count. The rest of the columns of Table 1 indicate conditions. For example, "IN_PORT", "MAC src" (media access control (MAC) address of the source of the packet), "MAC dst" (MAC address of the destination of the packet), "IP src" (Internet Protocol (IP) address of the source of the packet), "IP dst" (IP address of the destination of the packet), etc. are conditions that identify a particular packet. When the conditions are met, i.e., the particular packet is identified, the action indicated in the corresponding "Action" column is performed on the packet. The asterisks in Table 1 indicate an irrelevant condition."). |

| No. | '111 Patent Claim 19 | Lin '400 |
|---|---|---|
| 19 | The method according to claim 18, wherein the addresses are Internet Protocol (IP) addresses. | Lin '400 discloses the method according to claim 18, wherein the addresses are Internet Protocol (IP) addresses.<br><br>For example, Lin '400 discloses where the conditions include the IP address of the source or destination of the packet.<br><br>Lin '400 5:8-25 ("A flow table may include columns that indicate one or more conditions, a column that indicates an action to take when the conditions are met, and a column for statistics. A row on the flow table may comprise a flow rule. In the example of Table 1, the "Action" column indicates an action to take when conditions are met, and the "Count" column |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 681 of 1100

| No. | '111 Patent Claim 19 | Lin '400 |
|---|---|---|
| | | indicates statistics, such as byte count. The rest of the columns of Table 1 indicate conditions. For example, "IN_PORT", "MAC src" (media access control (MAC) address of the source of the packet), "MAC dst" (MAC address of the destination of the packet), "IP src" (Internet Protocol (IP) address of the source of the packet), "IP dst" (IP address of the destination of the packet), etc. are conditions that identify a particular packet. When the conditions are met, i.e., the particular packet is identified, the action indicated in the corresponding "Action" column is performed on the packet. The asterisks in Table 1 indicate an irrelevant condition."). |

| No. | '111 Patent Claim 20 | Lin '400 |
|---|---|---|
| 20[a] | The method according to claim 1, wherein the packet is an Transmission Control Protocol (TCP) packet that comprises source and destination TCP ports, a TCP sequence number, and a TCP sequence mask fields, and | Lin '400 discloses the method according to claim 1, wherein the packet is a Transmission Control Protocol (TCP) packet that comprises source and destination TCP ports, a TCP sequence number, and a TCP sequence mask fields.<br><br>For example, Lin '400 discloses conditions relating to the ingress and egress of TCP packets.<br><br>TABLE 2<br><br>IN_PORT . . . IP src **TCP** src port **TCP** dst port . . . Action Count<br><br>Ingress_port_ID * * * 80 * Egress port 120<br>Egress_port_ID * * 80 * * Ingress port 120<br>Ingress_port_ID * * * * * Redirect port 10<br>Redirect_port_ID * * * * * Ingress port 10<br><br>Lin '400 7:39-50 ("In the example of Table 2, the first two rows are bypass rules for bypassing packets coming from or going to a transport control protocol (TCP) port 80. More specifically, hypertext transfer protocol (HTTP) packets, i.e., port 80 packets, that are received in the ingress port with the Ingress_port_ID (i.e., ingress port 623-1) are forwarded directly to the egress port (i.e., egress port 623-4), instead of being redirected to the redirect port 623-2 for |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 682 of 1100

<table>
<tr><th>No.</th><th>’111 Patent Claim 20</th><th>Lin ’400</th></tr>
<tr><td></td><td></td><td>inspection by the security service 630. Similarly, HTTP packets received in the egress port with the Egress_port_ID (i.e., egress port 623-4) are forwarded directly to the ingress port 623-1 without being redirected to the security service 630.”).<br><br>TABLE 3<br><br>
<table>
<tr><th>IN_PORT</th><th>. . .</th><th>IP src</th><th>TCP src port</th><th></th><th></th><th>TCP dst port</th><th></th><th>. . .</th><th>Action</th><th>Count</th></tr>
<tr><td>Ingress_port_ID</td><td></td><td></td><td>*</td><td>*</td><td>*</td><td>80</td><td>*</td><td></td><td>Redirect port</td><td>10</td></tr>
<tr><td>Redirect_port_ID</td><td></td><td></td><td>*</td><td>*</td><td>80</td><td>*</td><td>*</td><td></td><td>Ingress port</td><td>10</td></tr>
<tr><td>Ingress_port_ID</td><td></td><td></td><td>*</td><td>*</td><td>*</td><td>*</td><td>*</td><td></td><td>Egress port</td><td>130</td></tr>
<tr><td>Egress_port_ID</td><td></td><td></td><td>*</td><td>*</td><td>*</td><td>*</td><td>*</td><td></td><td>Ingress port</td><td>130</td></tr>
</table>
<br>Lin ’400 8:10-18 (“In the example of Table 3, the top two rows are redirect flow rules for redirecting HTTP packets to the security service 630 for inspection, while the bottom two rows are bypass flow rules for all packets. Because the redirect flow rules are at higher priority than the bypass flow rules, HTTP packets are sent through the SDN pipe formed in the SDN switch 620 between the sender component 622 and the security service 630. All other packets bypass the SDN pipe, and are accordingly not inspected by the security service 630.<br><br>Lin ’400 Claim 8 (“The SDN computer network of claim 7, wherein the specified packets are packets having a particular transport control protocol (TCP) source or destination port.”).</td></tr>
<tr><td>20[b]</td><td>wherein the packet-applicable criterion is that the source TCP port, the destination TCP port, the TCP sequence number, the TCP sequence mask,</td><td>Lin ’400 discloses wherein the packet-applicable criterion is that the source TCP port, the destination TCP port, the TCP sequence number, the TCP sequence mask, or any combination thereof, matches a predetermined value or values.<br><br>For example, Lin ’400 discloses flow tables containing matching conditions regarding qualifications for the TCP packets.</td></tr>
</table>



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 683 of 1100

| No. | '111 Patent Claim 20 | Lin '400 |
|---|---|---|
| | or any combination thereof, matches a predetermined value or values. | *See supra* Claim 20[a].<br><br>Lin '400 7:24-27 ("In one embodiment, bypass flow rules are inserted in the flow tables 621 such that particular packets that do not need to be inspected are not redirected to the security service 630. This embodiment is explained with reference to example flow tables of Tables 2 and 3.<br><br>TABLE 2<br><br>IN_PORT ... IP src TCP src port TCP dst port ... Action Count<br><br>Ingress_port_ID * * * 80 * Egress port 120<br>Egress_port_ID * * 80 * * Ingress port 120<br>Ingress_port_ID * * * * * Redirect port 10<br>Redirect_port_ID * * * * * Ingress port 10<br><br>Lin '400 7:39-67 ("In the example of Table 2, the first two rows are bypass rules for bypassing packets coming from or going to a transport control protocol (TCP) port 80. More specifically, hypertext transfer protocol (HTTP) packets, i.e., port 80 packets, that are received in the ingress port with the Ingress_port_ID (i.e., ingress port 623-1) are forwarded directly to the egress port (i.e., egress port 623-4), instead of being redirected to the redirect port 623-2 for inspection by the security service 630. Similarly, HTTP packets received in the egress port with the Egress_port_ID (i.e., egress port 623-4) are forwarded directly to the ingress port 623-1 without being redirected to the security service 630.<br>In the example of Table 2, the bottom two rows are redirect flow rules for forming the SDN pipe between the sender component 622 and the security service 630. Because the bypass flow rules are inserted in the flow tables 621 with higher priority than the redirect flow rules, the bypass flow rules are followed by the SDN switch 620 before the redirect flow rules. Accordingly, HTTP packets are not redirected for inspection by the security service 630. Other packets, i.e., non-HTTP packets, are redirected to the security service 630 per the redirect flow rules. Bypass flow rules and redirect flow rules may be set at different priority levels to meet particular packet inspection needs. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 684 of 1100

| No. | ’111 Patent Claim 20 | Lin ’400 |
|---|---|---|
| | | The bypass and redirect flow rules also allow for inspection of particular packets, while allowing all other packets to bypass inspection. This is illustrated in the example flow table of Table 3.”).<br><br>TABLE 3<br><br>IN_PORT . . . IP src TCP src port TCP dst port . . . Action Count<br><br>Ingress_port_ID * * * 80 * Redirect port 10<br>Redirect_port_ID * * 80 * * Ingress port 10<br>Ingress_port_ID * * * * * Egress port 130<br>Egress_port_ID * * * * * Ingress port 130<br><br>Lin ’400 8:10-18 (“In the example of Table 3, the top two rows are redirect flow rules for redirecting HTTP packets to the security service 630 for inspection, while the bottom two rows are bypass flow rules for all packets. Because the redirect flow rules are at higher priority than the bypass flow rules, HTTP packets are sent through the SDN pipe formed in the SDN switch 620 between the sender component 622 and the security service 630. All other packets bypass the SDN pipe, and are accordingly not inspected by the security service 630.”).<br><br>Lin ’400 Claim 8 (“The SDN computer network of claim 7, wherein the specified packets are packets having a particular transport control protocol (TCP) source or destination port.”). |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 685 of 1100

| No. | '111 Patent Claim 21 | Lin '400 |
|---|---|---|
| 21 | The method according to claim 1, wherein the packet network comprises a Wide Area Network (WAN), Local Area Network (LAN), the Internet, Metropolitan Area Network (MAN), Internet Service Provider (ISP) backbone datacenter network, or inter - datacenter network. | Lin '400 discloses , the method according to claim 1, wherein the packet network comprises a Wide Area Network (WAN), Local Area Network (LAN), the Internet, Metropolitan Area Network (MAN), Internet Service Provider (ISP) backbone datacenter network, or inter - datacenter network.<br><br>For example, Lin '400 discloses communicating over a packet network which can comprise of a computer network utilizing the Internet via a network adapter or modem. A person of ordinary skill in the art would understand that a computer network exchanging IP packets would include the Internet.<br><br>Lin '400 2:47-65 ("FIG. 2 shows a schematic diagram of a computer system 100 that may be employed with embodiments of the present invention. The computer system 100 may be employed as a control plane and/or a data plane, for example. As another example, the computer system 100 may be employed to host a virtualization environment that supports a plurality of virtual machines. The computer system 100 may have fewer or more components to meet the needs of a particular application. The computer system 100 may include one or more processors 101. The computer system 100 may have one or more buses 103 coupling its various components. The computer system 100 may include one or more user input devices 102 (e.g., keyboard, mouse), one or more data storage devices 106 (e.g., hard drive, optical disk, Universal Serial Bus memory), a display monitor 104 (e.g., liquid crystal display, flat panel monitor), a computer network interface 105 (e.g., network adapter, modem), and a main memory 108 (e.g., random access memory). The computer network interface 105 may be coupled to a computer network 109.").<br><br>Lin '400 at 3:53-64 ("In one embodiment, the SDN computer network 600 is a virtual computer network that allows for transmission of packets from one virtual machine to another. Accordingly, the SDN controller 610 may comprise a virtual OpenFlow controller and the SDN switch 620 may comprise a virtual OpenFlow switch. The SDN computer network 600 may be implemented in a computer system comprising one or more computers that host a virtualization environment. For example, the SDN computer network 600 may be implemented in the Amazon Web Services virtualization environment. The sender component 622 may be a virtual machine in that embodiment.")<br>Lin '400 5:8-36 ("A flow table may include columns that indicate one or more conditions, a column that indicates an action to take when the conditions are met, and a column for |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 686 of 1100

| No. | '111 Patent Claim 21 | Lin '400 |
|---|---|---|
| | | statistics. A row on the flow table may comprise a flow rule. In the example of Table 1, the "Action" column indicates an action to take when conditions are met, and the "Count" column indicates statistics, such as byte count. The rest of the columns of Table 1 indicate conditions. For example, "IN_PORT", "MAC src" (media access control (MAC) address of the source of the packet), "MAC dst" (MAC address of the destination of the packet), "IP src" (Internet Protocol (IP) address of the source of the packet), "IP dst" (IP address of the destination of the packet), etc. are conditions that identify a particular packet. When the conditions are met, i.e., the particular packet is identified, the action indicated in the corresponding "Action" column is performed on the packet. The asterisks in Table 1 indicate an irrelevant condition. In the example of Table 1, the first and second rows are redirect flow rules for forming an SDN pipe between the sender component 622 and the security service 630. More specifically, the first row of Table 1 is a flow rule instructing the SDN switch 620 to forward packets received in a port having the Ingress_port_ID (e.g., ingress port 623-1) to the redirect port (e.g., redirect port 623-2). Similarly, the second row of Table 1 is a flow rule instructing the SDN switch 620 to forward packets received in a port having a "Redirect_port_ID" to the ingress port.").<br><br>Lin '400 6:1-12 ("The SDN controller 610 may insert flow rules in the flow tables 621 (see arrow 601) to create an SDN pipe (labeled as 625) between the sender component 622 and the security service 630. The SDN pipe allows outgoing packets sent by the sender component 622 or incoming packets going to the sender component 622 to be redirected to the security service 630 for inspection before the packets are sent out of the SDN switch 620. In one embodiment, the SDN pipe is created by creating a first flow rule that forwards packets received in the ingress port 623-1 to the redirect port 623-2, and a second flow rule that forwards packets received in the redirect port 623-2 to the ingress port 623-1.").<br><br>Lin '400 6:40-54 ("After the redirect flow rules for creating the SDN pipe are inserted in the flow tables 621, any packet received by the SDN switch 620 in the ingress port 623-1 will be identified as to be forwarded to the redirect port 623-2, and any packet received by the SDN switch 620 in the redirect port 623-2 will be identified as to be forwarded to the ingress port 623-1 (see arrow 602). This allows the security service 630 to receive from the redirect port 623-2 all outgoing packets sent by the sender component 622 to the ingress port 623-1. The security service 630 may inspect the outgoing packets for compliance with security policies. The security service 630 may drop, or perform other security response, to packets |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 687 of 1100

| No. | '111 Patent Claim 21 | Lin '400 |
|---|---|---|
| | | that do not pass inspection (e.g., packets that do not meet firewall policies, packets containing prohibited payload, packets with malicious content, etc.).").<br><br>TABLE 1<br><br>IN_PORT \| MAC src \| MAC dst \| IP src \| IP dst \| . . . \| Action \| Count<br>Ingress_port_ID \| * * * * * \| Redirect port \| 10<br>Redirect_port_ID \| * * * * * \| Ingress port \| 10 |

| No. | '111 Patent Claim 22 | Lin '400 |
|---|---|---|
| 22 | The method according to claim 1, wherein the first entity is a server device and the second entity is a client device, or wherein the first entity is a client device and the second entity is a server device. | Lin '400 discloses the method according to claim 1, wherein the first entity is a server device and the second entity is a client device, or wherein the first entity is a client device and the second entity is a server device.<br><br>For example, Lin '400 discloses a sender component and a destination component. Lin '400 further discloses that the sender component can be a sender computer. A person of ordinary skill in the art would understand that a destination component can be a server device. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Lin '400 is found to not meet this limitation, wherein the first entity is a server device and the second entity is a client device, or wherein the first entity is a client device and the second entity is a server device would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>Lin '400 at 3:11-24 ("Network security vendors provide network security services, such as firewall or deep packet inspection (DPI). Generally speaking, to provide network security services, packets of network traffic are intercepted for inspection. One way of intercepting network traffic is to place the Security service in the middle of the packet forwarding path. This is illustrated in FIG. 3, where packets from a sender component (e.g., a sender computer) are received in an ingress port of a Switch, forwarded to an egress port of the switch, and forwarded to the ingress port of a security component, Such as a security service. The security service may inspect the packets, and forward the packets to an egress port of the switch toward |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 688 of 1100

| No. | '111 Patent Claim 22 | Lin '400 |
|---|---|---|
| | | the next hop, which may be another Switch or a destination component (e.g., destination computer), for example.")<br><br>Lin '400 at 3:53-64 ("In one embodiment, the SDN computer network 600 is a virtual computer network that allows for transmission of packets from one virtual machine to another. Accordingly, the SDN controller 610 may comprise a virtual OpenFlow controller and the SDN switch 620 may comprise a virtual OpenFlow switch. The SDN computer network 600 may be implemented in a computer system comprising one or more computers that host a virtualization environment. For example, the SDN computer network 600 may be implemented in the Amazon Web Services virtualization environment. The sender component 622 may be a virtual machine in that embodiment.")<br><br>Lin '400 5:37-55 ("The SDN computer network 600 may include a security component in the form of the security service 630. The security service 630 may comprise a virtual machine that provides computer network security services, such as packet inspection, for the sender component 622 and other virtual machines. For example, the security service 630 may comprise a virtual machine with a virtual network interface card that is coupled to the redirect port 623-2 and re-inject port 623-3 of the SDN switch 620. The security service 630 may inspect packets for compliance/non-compliance with security policies, such as for presence of malicious code, compliance with firewall rules and access control lists, network intrusion detection, and other computer network security services. The security service 630 may employ conventional packet inspection algorithms. The security service 630 may comprise the Trend Micro Deep Security™ service, for example. The security service 630 may also comprise a physical machine, e.g., a server computer, an appliance, a gateway computer, etc.").<br><br>Lin '400 8:18-45 ("FIG. 7 schematically illustrates inspection of outgoing packets sent by the sender component 622 in the SDN computer network 600 in accordance with an embodiment of the present invention. In the example of FIG. 7, the sender component 622 (e.g., a virtual machine, a laptop computer, desktop computer, etc.) transmits outgoing packets to the ingress port 623-1 (see arrow 651). The SDN switch 620 receives the outgoing packets in the ingress port 623-1 and follows a flow rule that pertains to the outgoing packets (see arrow 652). In the example of FIG. 7, a redirect flow rule dictates that packets received by the SDN switch 620 in the ingress port 623-1 are to be forwarded to the redirect port 623-2. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 689 of 1100

| No. | '111 Patent Claim 22 | Lin '400 |
|---|---|---|
| | | Accordingly, the SDN switch 620 forwards the outgoing packets to the redirect port 623-2 (see arrow 653), which is connected to the security service 630 (e.g., a virtual machine, server computer, appliance, etc.). The security service 630 receives the outgoing packets from the redirect port 623-2 (see arrow 654) and inspects the outgoing packets. After inspection, the security service 630 re-injects the outgoing packets (e.g., outgoing packets that passed inspection) back into the SDN switch 620 by way of the re-inject port 623-3 (see arrow 655). The SDN switch 620 receives the outgoing packets on the re-inject port 623-3. The SDN switch 620 forwards the outgoing packets from the re-inject port 623-3 to the egress port 623-4 in accordance with the L2 switching logic of the SDN computer network 600 (see arrow 657). The outgoing packets exit the SDN switch 620 through the egress port 623-4 (see arrow 658) and move towards their destination."<br><br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Lin '400 in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 22 of Exhibit E-4 renders the claim, including the present limitation, obvious. Below is an example.<br><br>For example, Swenson discloses a user device and an origin server.<br><br>*See supra* at Claim 1.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 690 of 1100

| No. | '111 Patent Claim 22 | Lin '400 |
|---|---|---|
| | | Swenson at [0030] ("The video optimizer 150 is a computer server that provides video and image optimization and delivers opti-mized video and image content to the user devices 110 via the network 120. The video and image optimization is an on-demand service provided through the transcoding of the video and image content. For example, when a user device attempts to retrieve video from the origin server 160, the network controller 140 may decide that the flow meets certain criteria for content optimization. The network controller 140 then redirected the user devices 110 to the video optimizer 150 to retrieve the optimized content. The video optimizer 150 receives information in the redirect request from the user devices 110 or from the network controller 140 about the video or image content to be optimized and retrieve the video or image content from the corresponding origin server 160 for optimization and subsequent delivery to the user devices 110.")<br><br>Swenson at [0032] ("The video optimizer 150 and the origin server 160 are typically formed of one or more computers. While only one server of each video optimizer 150 and origin server 160 is shown in the environment 100 of FIG. 1, different embodi-ments may include multiple web servers and video servers operated by a single entity or multiple entities. In other embodiments, a single server may also provide different func-tionalities, such as delivering web content as a web server, as well as serving optimized video content.")<br><br>Swenson at [0034] ("The machine may be a server computer, a client computer, a personal computer (PC), a tablet PC, a set-top box (STB), a personal digital assistant (PDA), a cellular tele-phone, a smart phone, a web appliance, a network router, switch or bridge, or any machine capable of executing instructions 224 ( sequential or otherwise) that specify actions to be taken by that machine. Further, while only a single machine is illustrated, the term "machine" shall also be taken to include any collection of machines that individually or jointly execute instructions 224 to perform any one or more of the methodologies discussed herein.")<br><br>Swenson at [0055] ("The video optimizer redirector 318 generates a redi-rect request to a URL pointing to the video optimizer 150 if the video is deemed to be transcoded. In one embodiment, the URL may contain at least one of a video resolution, a video bit rate, a video frame rate divisor, an audio sample rate and number of channels, an audio bit rate, a source URL, a user agent of a client, a source domain cookie and any other authentication data by the video optimizer 150. The video optimizer redirector 318 rewrites the original response with the HTTP redirect and sets the location header to the new URL. This causes the user |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 691 of 1100

| No. | '111 Patent Claim 22 | Lin '400 |
|---|---|---|
| | | devices 110 to issue a new request to the video optimizer 150. The video optimizer redirector 318 also has the logic to look for incoming URLs generated by itself so that they are not intercepted again.")<br><br>Swenson at [0058] ("Referring now to FIG. 4A, network traffic flows from the user device 110 through the steering device 130 and arrive at the origin server 160 over the network request path. For example, a browser on the user device 110 may request web content from the origin server 160. A HTTP request message initiated at the user device 110 is forwarded to the steering device 130 over the network link 411. A data switch 402 inside the steering device 130 then relays the request message to the origin server 160 over the network link 412. On the opposite direction, network traffic originated from the origin server 160 flows through the steering device 130 back to the user device 110 over the network response path. For example, the origin server 160 responds to the user request by sending web content over the network link 413 to the steering device 130, which forwards the web content to the user device 110 over the network link 416. Note that the network links 411 and 416 are two opposite directions on the same physical link, so are the network link pair 414 and 415. On the other hand, the network link pair 412 and 413 may or may not share the same network path because traffic between the steering device 130 and origin server 160 on opposite directions may be routed differently over one or more routers.")<br><br>Swenson at [0070] ("Once the user device 110 receives the HTTP redirect request 620, the user device 110 sends the request over the network to the video optimizer 150. In one embodiment, the network controller 140 monitors the traffic and/or requests from the client device 110 as the HTTP redirect request 620 is routed to the video optimizer 150. In such a configuration, the video optimizer 150 only sees requests for video files that need to be transcoded (i.e., optimized) and are associated with a HTTP redirect request 620. As such, the video optimizer 150 is not burdened with all the requests generated by a user device 110.")<br><br>Swenson at [0095] ("Certain embodiments are described herein as including logic or a number of components, modules, or mechanisms. Modules may constitute either software mod-ules ( e.g., code embodied on a machine-readable medium or in a transmission signal) or hardware modules. A hardware module is tangible unit capable of performing certain opera-tions and may be configured or arranged in a certain manner. In example embodiments, one or more computer systems ( e.g., a standalone, client or server computer system) or one or more |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 692 of 1100

| No. | '111 Patent Claim 22 | Lin '400 |
|---|---|---|
| | | hardware modules of a computer system (e.g., a pro-cessor or a group of processors 102) may be configured by software (e.g., an application or application portion) as a hardware module that operates to perform certain operations as described herein.") |

| No. | '111 Patent Claim 23 | Lin '400 |
|---|---|---|
| 23[a] | The method according to claim 22, wherein the server device comprises a web server, and | Lin '400 discloses the method according to claim 22, wherein the server device comprises a web server.<br><br>For example, Lin '400 discloses a sender component and a destination component. Lin '400 further discloses that the sender component can be a sender computer. A person of ordinary skill in the art would understand that a destination component can be a web server. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Lin '400 is found to not meet this limitation, wherein the server device comprises a web server would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>Lin '400 at 3:11-24 ("Network security vendors provide network security services, such as firewall or deep packet inspection (DPI). Generally speaking, to provide network security services, packets of network traffic are intercepted for inspection. One way of intercepting network traffic is to place the Security service in the middle of the packet forwarding path. This is illustrated in FIG. 3, where packets from a sender component (e.g., a sender computer) are received in an ingress port of a Switch, forwarded to an egress port of the switch, and forwarded to the ingress port of a security component, Such as a security service. The security service may inspect the packets, and forward the packets to an egress port of the switch toward the next hop, which may be another Switch or a destination component (e.g., destination computer), for example.")<br><br>Lin '400 at 3:53-64 ("In one embodiment, the SDN computer network 600 is a virtual computer network that allows for transmission of packets from one virtual machine to another. Accordingly, the SDN controller 610 may comprise a virtual OpenFlow controller and the SDN switch 620 may comprise a virtual OpenFlow switch. The SDN computer network 600 |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 693 of 1100

| No. | '111 Patent Claim 23 | Lin '400 |
|---|---|---|
| | | may be implemented in a computer system comprising one or more computers that host a virtualization environment. For example, the SDN computer network 600 may be implemented in the Amazon Web Services virtualization environment. The sender component 622 may be a virtual machine in that embodiment.")<br><br>Lin '400 5:37-55 ("The SDN computer network 600 may include a security component in the form of the security service 630. The security service 630 may comprise a virtual machine that provides computer network security services, such as packet inspection, for the sender component 622 and other virtual machines. For example, the security service 630 may comprise a virtual machine with a virtual network interface card that is coupled to the redirect port 623-2 and re-inject port 623-3 of the SDN switch 620. The security service 630 may inspect packets for compliance/non-compliance with security policies, such as for presence of malicious code, compliance with firewall rules and access control lists, network intrusion detection, and other computer network security services. The security service 630 may employ conventional packet inspection algorithms. The security service 630 may comprise the Trend Micro Deep Security™ service, for example. The security service 630 may also comprise a physical machine, e.g., a server computer, an appliance, a gateway computer, etc.").<br><br>Lin '400 8:18-45 ("FIG. 7 schematically illustrates inspection of outgoing packets sent by the sender component 622 in the SDN computer network 600 in accordance with an embodiment of the present invention. In the example of FIG. 7, the sender component 622 (e.g., a virtual machine, a laptop computer, desktop computer, etc.) transmits outgoing packets to the ingress port 623-1 (see arrow 651). The SDN switch 620 receives the outgoing packets in the ingress port 623-1 and follows a flow rule that pertains to the outgoing packets (see arrow 652). In the example of FIG. 7, a redirect flow rule dictates that packets received by the SDN switch 620 in the ingress port 623-1 are to be forwarded to the redirect port 623-2. Accordingly, the SDN switch 620 forwards the outgoing packets to the redirect port 623-2 (see arrow 653), which is connected to the security service 630 (e.g., a virtual machine, server computer, appliance, etc.). The security service 630 receives the outgoing packets from the redirect port 623-2 (see arrow 654) and inspects the outgoing packets. After inspection, the security service 630 re-injects the outgoing packets (e.g., outgoing packets that passed inspection) back into the SDN switch 620 by way of the re-inject port 623-3 (see arrow 655). The SDN switch 620 receives the outgoing packets on the re-inject port 623-3. The SDN |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 694 of 1100

| No. | '111 Patent Claim 23 | Lin '400 |
|---|---|---|
| | | switch 620 forwards the outgoing packets from the re-inject port 623-3 to the egress port 623-4 in accordance with the L2 switching logic of the SDN computer network 600 (see arrow 657). The outgoing packets exit the SDN switch 620 through the egress port 623-4 (see arrow 658) and move towards their destination."<br><br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Lin '400 in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 23(a) of Exhibit E-4 renders the claim, including the present limitation, obvious. Below is an example.<br><br>For example, Swenson discloses servers that's provide different functionalities, such as delivering web content as a web server.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0030] ("The video optimizer 150 is a computer server that provides video and image optimization and delivers opti-mized video and image content to the user devices 110 via the network 120. The video and image optimization is an on-demand service provided through the transcoding of the video and image content. For example, when a user device attempts to retrieve video from the origin server 160, the network controller 140 may decide that the flow meets certain criteria for content optimization. The network controller 140 then redirected the user devices 110 to the video optimizer 150 to retrieve the optimized content. The video optimizer 150 receives information in the redirect request from the user devices 110 or from the network controller 140 about the video or image content to be optimized and |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 695 of 1100

| No. | '111 Patent Claim 23 | Lin '400 |
|---|---|---|
| | | retrieve the video or image content from the corresponding origin server 160 for optimization and subsequent delivery to the user devices 110.")<br><br>Swenson at [0032] ("The video optimizer 150 and the origin server 160 are typically formed of one or more computers. While only one server of each video optimizer 150 and origin server 160 is shown in the environment 100 of FIG. 1, different embodi-ments may include multiple web servers and video servers operated by a single entity or multiple entities. In other embodiments, a single server may also provide different func-tionalities, such as delivering web content as a web server, as well as serving optimized video content.")<br><br>Swenson at [0034] ("The machine may be a server computer, a client computer, a personal computer (PC), a tablet PC, a set-top box (STB), a personal digital assistant (PDA), a cellular tele-phone, a smart phone, a web appliance, a network router, switch or bridge, or any machine capable of executing instructions 224 ( sequential or otherwise) that specify actions to be taken by that machine. Further, while only a single machine is illustrated, the term "machine" shall also be taken to include any collection of machines that individually or jointly execute instructions 224 to perform any one or more of the methodologies discussed herein.")<br><br>Swenson at [0058] ("Referring now to FIG. 4A, network traffic flows from the user device 110 through the steering device 130 and arrive at the origin server 160 over the network request path. For example, a browser on the user device 110 may request web content from the origin server 160. A HTTP request message initiated at the user device 110 is forwarded to the steering device 130 over the network link 411. A data switch 402 inside the steering device 130 then relays the request message to the origin server 160 over the network link 412. On the opposite direction, network traffic originated from the origin server 160 flows through the steering device 130 back to the user device 110 over the network response path. For example, the origin server 160 responds to the user request by sending web content over the network link 413 to the steering device 130, which forwards the web content to the user device 110 over the network link 416. Note that the network links 411 and 416 are two opposite directions on the same physical link, so are the network link pair 414 and 415. On the other hand, the network link pair 412 and 413 may or may not share the same network path because traffic between the steering device 130 and origin server 160 on opposite directions may be routed differently over one or more routers.") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 696 of 1100

| No. | '111 Patent Claim 23 | Lin '400 |
|---|---|---|
| 23[b] | wherein the client device comprises a smartphone, a tablet computer, a personal computer, a laptop computer, or a wearable computing device. | Lin '400 discloses wherein the client device comprises a smartphone, a tablet computer, a personal computer, a laptop computer, or a wearable computing device.<br><br>For example, Lin '400 discloses that the sender component can be a virtual machine, a laptop computer, or a desktop computer, etc.<br><br>Lin '400 8:18-45 ("FIG. 7 schematically illustrates inspection of outgoing packets sent by the sender component 622 in the SDN computer network 600 in accordance with an embodiment of the present invention. In the example of FIG. 7, the sender component 622 (e.g., a virtual machine, a laptop computer, desktop computer, etc.) transmits outgoing packets to the ingress port 623-1 (see arrow 651). The SDN switch 620 receives the outgoing packets in the ingress port 623-1 and follows a flow rule that pertains to the outgoing packets (see arrow 652). In the example of FIG. 7, a redirect flow rule dictates that packets received by the SDN switch 620 in the ingress port 623-1 are to be forwarded to the redirect port 623-2. Accordingly, the SDN switch 620 forwards the outgoing packets to the redirect port 623-2 (see arrow 653), which is connected to the security service 630 (e.g., a virtual machine, server computer, appliance, etc.). The security service 630 receives the outgoing packets from the redirect port 623-2 (see arrow 654) and inspects the outgoing packets. After inspection, the security service 630 re-injects the outgoing packets (e.g., outgoing packets that passed inspection) back into the SDN switch 620 by way of the re-inject port 623-3 (see arrow 655). The SDN switch 620 receives the outgoing packets on the re-inject port 623-3. The SDN switch 620 forwards the outgoing packets from the re-inject port 623-3 to the egress port 623-4 in accordance with the L2 switching logic of the SDN computer network 600 (see arrow 657). The outgoing packets exit the SDN switch 620 through the egress port 623-4 (see arrow 658) and move towards their destination." |

| No. | '111 Patent Claim 24 | Lin '400 |
|---|---|---|
| 24 | The method according to claim 22, wherein the communication between the network node and the controller | Lin '400 discloses the method according to claim 22, wherein the communication between the network node and the controller is based on, or uses, a standard protocol.<br><br>For example, Lin '400 discloses using an Openflow protocol for communications between the switch and the security device. |

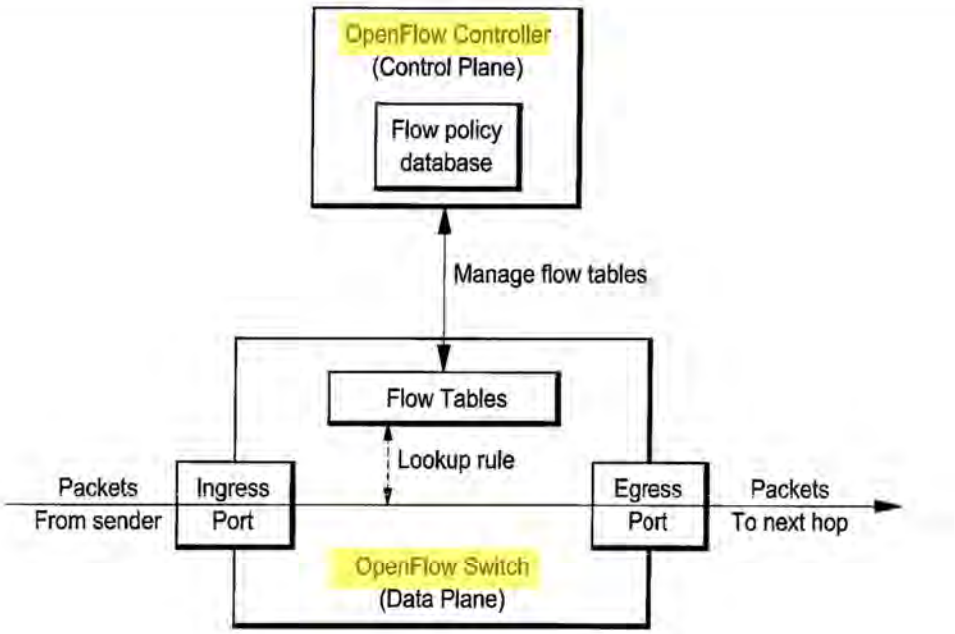


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 697 of 1100

| No. | '111 Patent Claim 24 | Lin '400 |
|---|---|---|
| | is based on, or uses, a standard protocol. | Lin '400 1:17-32 ("The OpenFlow™ protocol is an open protocol for remotely controlling forwarding tables of network switches that are enabled for SDN. Generally speaking, the OpenFlow protocol allows direct access to and manipulation of the forwarding plane of network devices, such as switches and routers. A control plane of an OpenFlow™ protocol-compliant computer network (also referred to as an "OpenFlow™ controller") may communicate with OpenFlow™ switches (i.e., network switches that are compliant with the OpenFlow™ protocol) to set flow policies that specify how the switches should manipulate packets of network traffic. Example packet manipulation actions include forwarding a packet to a specific port, modifying one or more fields of the packet, asking the controller for action to perform on the packet, or dropping the packet."). <br><br> Lin '400 1:33-43 ("FIG. 1 shows a schematic diagram of an SDN computer network that is compliant with the OpenFlow™ protocol. Generally speaking, the OpenFlow™ protocol separates the control plane from the data plane. An OpenFlow™ controller serves as a control plane for making forwarding decisions based on flow policies, which may be stored in a flow policy database. The controller determines flow policies in conjunction with network forwarding setting and network topology. The flow policies may contain a condition and corresponding action to be performed when the condition is met. The action may specify how to manipulate a packet."). <br><br> Lin '400 1:44-54 ("An OpenFlow™ switch serves as the data plane that forwards packets, e.g., from an ingress port to an egress port, according to flow tables maintained by the data plane. The data plane is a replacement of traditional switches. When the data plane does not know how to manipulate a specific packet, the data plane may request the controller to receive a flow rule for the specific packet, and store the flow rule in the flow tables. Other packets that meet the same condition as the specific packet will be processed in accordance with the flow rule. The control plane may also actively insert flow rules into the flow tables."). <br><br> Lin '400 Claim 8 ("The SDN computer network of claim 7, wherein the specified packets are packets having a particular transport control protocol (TCP) source or destination port."). |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 698 of 1100

| No. | '111 Patent Claim 24 | Lin '400 |
|---|---|---|
| | |  <br>*FIG. 1*<br>*(PRIOR ART)*<br>Fig. 1 (annotation added) |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 699 of 1100

| No. | '111 Patent Claim 27 | Lin '400 |
|---|---|---|
| 27 | The method according to claim 1, wherein the network node comprises a router, a switch, or a bridge. | Lin '400 discloses the method according to claim 1, wherein the network node comprises a router, a switch, or a bridge.<br><br>For example, Lin '400 discloses use of a switch or a router as the claimed network node.<br><br>*See supra* at Claim 1.<br><br>Lin '400 1:58-2:4 ("In one embodiment, a software defined networking (SDN) computer network includes an SDN controller and an SDN switch. The SDN controller inserts flow rules in a flow table of the SDN switch to create an SDN pipe between a sender component and a security component. A broadcast function of the SDN switch to the ports that form the SDN pipe may be disabled. The SDN pipe allows outgoing packets sent by the sender component to be received by the security component. The security component inspects the outgoing packets for compliance with security policies and allows the outgoing packets to be forwarded to their destination when the outgoing packets pass inspection. The SDN controller may also insert a flow rule in the flow table of the SDN switch to bypass inspection of specified packets.").<br><br>Lin '400 Claim 1 ("1. A software defined networking (SDN) computer network comprising:<br>an SDN switch comprising a plurality of ports that receives network traffic of an SDN computer network, the SDN switch having a first port coupled to a sender component and a second port coupled to a security component, the SDN switch comprising a flow table that comprises a first flow rule to forward a packet received in the first port to the second port and a second flow rule to forward a packet received in the second port to the first port, the SDN switch receiving outgoing packets from the first port and forwarding the outgoing packets to the second port in accordance with the first flow rule, the outgoing packets being sent by the sender component to a destination component; and<br>an SDN controller that controls forwarding behavior of the SDN switch and inserts the first and second flow rules into the flow table of the SDN switch,<br>wherein the security component receives the outgoing packets from the second port of the SDN switch, inspects the outgoing packets, and allows the outgoing packets to be forwarded to their destination when the outgoing packets pass inspection,<br>wherein the security component allows the outgoing packets to be forwarded to their destination by instructing the SDN switch to release copies of the outgoing packets.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 700 of 1100

| No. | '111 Patent Claim 27 | Lin '400 |
|---|---|---|
| | | Lin '400 Claim 3 ("The SDN computer network of claim 1, wherein the SDN switch comprises a physical packet switch, the SDN controller comprises one or more computers that send flow rules to the SDN switch, and the sender component comprises a computer coupled to the first port of the SDN switch.").<br><br>Lin '400 3:11-24 ("Network security vendors provide network security services, such as firewall or deep packet inspection (DPI). Generally speaking, to provide network security services, packets of network traffic are intercepted for inspection. One way of intercepting network traffic is to place the security service in the middle of the packet forwarding path. This is illustrated in FIG. 3, where packets from a sender component (e.g., a sender computer) are received in an ingress port of a switch, forwarded to an egress port of the switch, and forwarded to the ingress port of a security component, such as a security service. The security service may inspect the packets, and forward the packets to an egress port of the switch toward the next hop, which may be another switch or a destination component (e.g., destination computer), for example.").<br><br>Lin '400 1:17-32 (" "The OpenFlow™ protocol is an open protocol for remotely controlling forwarding tables of network switches that are enabled for SDN. Generally speaking, the OpenFlow protocol allows direct access to and manipulation of the forwarding plane of network devices, such as switches and routers. A control plane of an OpenFlow™ protocol-compliant computer network (also referred to as an "OpenFlow™ controller") may communicate with OpenFlow™ switches (i.e., network switches that are compliant with the OpenFlow™ protocol) to set flow policies that specify how the switches should manipulate packets of network traffic. Example packet manipulation actions include forwarding a packet to a specific port, modifying one or more fields of the packet, asking the controller for action to perform on the packet, or dropping the packet."). |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 701 of 1100

| No. | '111 Patent Claim 28 | Lin '400 |
|---|---|---|
| 28 | The method according to claim 1, wherein the packet network is an Internet Protocol (IP) network, and the packet is an IP packet. | Lin '400 discloses the method according to claim 1, wherein the packet network is an Internet Protocol (IP) network, and the packet is an IP packet.<br><br>For example, Lin '400 discloses the use of IP addresses to identify packets being transmitted across the packet network (IP network).<br><br>*See supra* Claim 1.<br><br>Lin '400 5:8-25 ("A flow table may include columns that indicate one or more conditions, a column that indicates an action to take when the conditions are met, and a column for statistics. A row on the flow table may comprise a flow rule. In the example of Table 1, the "Action" column indicates an action to take when conditions are met, and the "Count" column indicates statistics, such as byte count. The rest of the columns of Table 1 indicate conditions. For example, "IN_PORT", "MAC src" (media access control (MAC) address of the source of the packet), "MAC dst" (MAC address of the destination of the packet), "IP src" (Internet Protocol (IP) address of the source of the packet), "IP dst" (IP address of the destination of the packet), etc. are conditions that identify a particular packet. When the conditions are met, i.e., the particular packet is identified, the action indicated in the corresponding "Action" column is performed on the packet. The asterisks in Table 1 indicate an irrelevant condition."). |

| No. | '111 Patent Claim 29 | Lin '400 |
|---|---|---|
| 29 | The method according to claim 28, wherein the packet network is an Transmission Control Protocol (TCP) network, and the packet is an TCP packet. | Lin '400 discloses the method according to claim 28, wherein the packet network is an Transmission Control Protocol (TCP) network, and the packet is an TCP packet.<br><br>For example, Lin '400 discloses the use of TCP addresses to identify packets being transmitted across the packet network (TCP network).<br><br>*See supra* Claim 28. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 702 of 1100

| No. | '111 Patent Claim 29 | Lin '400 |
|---|---|---|

TABLE 2

| IN_PORT | . . . | IP src | **TCP** src port | | **TCP** dst port | | . . . | Action | Count |
|---|---|---|---|---|---|---|---|---|---|
| Ingress_port_ID | | * | * | * | 80 | * | | Egress port | 120 |
| Egress_port_ID | | * | * | 80 | * | * | | Ingress port | 120 |
| Ingress_port_ID | | * | * | * | * | * | | Redirect port | 10 |
| Redirect_port_ID | | * | * | * | * | * | | Ingress port | 10 |

Lin '400 7:39-50 ("In the example of Table 2, the first two rows are bypass rules for bypassing packets coming from or going to a transport control protocol (TCP) port 80. More specifically, hypertext transfer protocol (HTTP) packets, i.e., port 80 packets, that are received in the ingress port with the Ingress_port_ID (i.e., ingress port 623-1) are forwarded directly to the egress port (i.e., egress port 623-4), instead of being redirected to the redirect port 623-2 for inspection by the security service 630. Similarly, HTTP packets received in the egress port with the Egress_port_ID (i.e., egress port 623-4) are forwarded directly to the ingress port 623-1 without being redirected to the security service 630.").

TABLE 3

| IN_PORT | . . . | IP src | **TCP** src port | | **TCP** dst port | | . . . | Action | Count |
|---|---|---|---|---|---|---|---|---|---|
| Ingress_port_ID | | * | * | * | 80 | * | | Redirect port | 10 |
| Redirect_port_ID | | * | * | 80 | * | * | | Ingress port | 10 |
| Ingress_port_ID | | * | * | * | * | * | | Egress port | 130 |
| Egress_port_ID | | * | * | * | * | * | | Ingress port | 130 |

Lin '400 8:10-18 ("In the example of Table 3, the top two rows are redirect flow rules for redirecting HTTP packets to the security service 630 for inspection, while the bottom two



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 703 of 1100

| No. | '111 Patent Claim 29 | Lin '400 |
|---|---|---|
| | | rows are bypass flow rules for all packets. Because the redirect flow rules are at higher priority than the bypass flow rules, HTTP packets are sent through the SDN pipe formed in the SDN switch 620 between the sender component 622 and the security service 630. All other packets bypass the SDN pipe, and are accordingly not inspected by the security service 630.<br><br>Lin '400 Claim 8 ("The SDN computer network of claim 7, wherein the specified packets are packets having a particular transport control protocol (TCP) source or destination port."). |

| No. | '111 Patent Claim 30 | Lin '400 |
|---|---|---|
| 30[a] | The method according to claim 1, further comprising: receiving, by the network node from the first entity over the packet network, one or more additional packets; | Lin '400 discloses the method according to claim 1, further comprising: receiving, by the network node from the first entity over the packet network, one or more additional packets.<br><br>For example, Lin '400 discloses the switch receiving a packet from the sender component over the network, as it applies to one or more individual packets.<br><br>*See also* Claim 1[c]. |
| 30[b] | checking, by the network node, if any one of the one or more additional packets satisfies the criterion; | Lin '400 discloses checking, by the network node, if any one of the one or more additional packets satisfies the criterion.<br><br>For example, Lin '400 discloses matching the packet to criteria in the flow tables, as it applies to one or more individual packets.<br><br>*See also* Claim 1[d]. |
| 30[c] | responsive to an additional packet not satisfying the criterion, sending, by the network node over the packet network, the | Lin '400 discloses responsive to an additional packet not satisfying the criterion, sending, by the network node over the packet network, the additional packet to the second entity.<br><br>For example, Lin '400 discloses, upon receiving a match of a criteria between the packet and the flow table, sending the packet over the network to the security device or to the destination, as it applies to one or more individual packets. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 704 of 1100

| No. | '111 Patent Claim 30 | Lin '400 |
|---|---|---|
| | additional packet to the second entity; and | *See also* Claim 1[e]. |
| 30[d] | responsive to the additional packet satisfying the criterion, sending the additional packet, by the network node over the packet network, in response to the instruction. | Lin '400 discloses responsive to the additional packet satisfying the criterion, sending the additional packet, by the network node over the packet network, in response to the instruction.<br><br>For example, Lin '400 teaches that the devices check for a specific packet-applicable criterion, where if a packet satisfies this criterion by indication that port 80 is the s destination port, then the SDN switch sends the packet over the packet network to the security device, as it applies to one or more individual packets.<br><br>*See also* Claim 1[f]. |

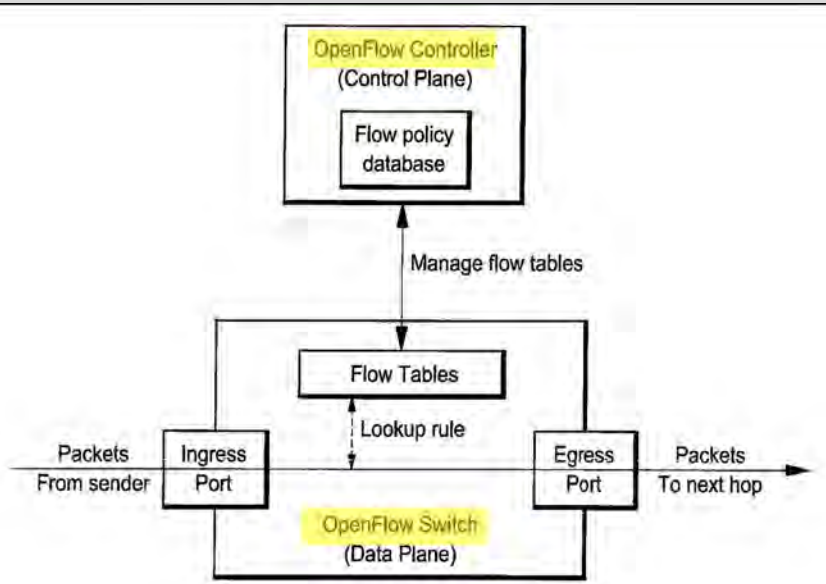
| No. | '111 Patent Claim 31 | Lin '400 |
|---|---|---|
| 31[a] | The method according to claim 1, wherein the packet network is a Software Defined Network (SDN), | Lin '400 discloses the method according to claim 1, wherein the packet network is a Software Defined Network (SDN).<br><br>For example, Lin '400 discloses using a software-defined network as the packet network.<br><br>Lin '400 Claim 1 ("A software defined networking (SDN) computer network comprising:<br>an SDN switch comprising a plurality of ports that receives network traffic of an SDN computer network, the SDN switch having a first port coupled to a sender component and a second port coupled to a security component, the SDN switch comprising a flow table that comprises a first flow rule to forward a packet received in the first port to the second port and a second flow rule to forward a packet received in the second port to the first port, the SDN switch receiving outgoing packets from the first port and forwarding the outgoing packets to the second port in accordance with the first flow rule, the outgoing packets being sent by the sender component to a destination component; and<br>an SDN controller that controls forwarding behavior of the SDN switch and inserts the first and second flow rules into the flow table of the SDN switch, |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 705 of 1100

| No. | '111 Patent Claim 31 | Lin '400 |
|---|---|---|
| | | wherein the security component receives the outgoing packets from the second port of the SDN switch, inspects the outgoing packets, and allows the outgoing packets to be forwarded to their destination when the outgoing packets pass inspection,<br>wherein the security component allows the outgoing packets to be forwarded to their destination by instructing the SDN switch to release copies of the outgoing packets.").<br><br>Lin '400 1:11-17 ("Software defined networking (SDN) is an emerging architecture for computer networking. Unlike traditional computer network architectures, SDN separates the control plane from the data plane. This provides many advantages, including relatively fast experimentation and optimization of switching and routing policies. SDN is applicable to both physical (i.e., real) and virtual computer networks.").<br><br>Lin '400 1:17-32 ("The OpenFlow™ protocol is an open protocol for remotely controlling forwarding tables of network switches that are enabled for SDN. Generally speaking, the OpenFlow protocol allows direct access to and manipulation of the forwarding plane of network devices, such as switches and routers. A control plane of an OpenFlow™ protocol-compliant computer network (also referred to as an "OpenFlow™ controller") may communicate with OpenFlow™ switches (i.e., network switches that are compliant with the OpenFlow™ protocol) to set flow policies that specify how the switches should manipulate packets of network traffic. Example packet manipulation actions include forwarding a packet to a specific port, modifying one or more fields of the packet, asking the controller for action to perform on the packet, or dropping the packet."). |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 706 of 1100

| No. | '111 Patent Claim 31 | Lin '400 |
|---|---|---|
| | |  <br>*FIG. 1*<br>*(PRIOR ART)*<br>Fig. 1 (annotation added) |
| 31[b] | the packet is routed as part of a data plane and | Lin '400 discloses that the packet is routed as part of a data plane.<br><br>For example, Lin '400 discloses that the packet is routed through the switch, where the switch comprises the data plane.<br><br>Lin '400 1:11-17 ("Software defined networking (SDN) is an emerging architecture for computer networking. Unlike traditional computer network architectures, SDN separates the control plane from the data plane. This provides many advantages, including relatively fast experimentation and optimization of switching and routing policies. SDN is applicable to both physical (i.e., real) and virtual computer networks."). |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 707 of 1100

| No. | '111 Patent Claim 31 | Lin '400 |
|---|---|---|
| | | Lin '400 3:40-64 ("Referring now to FIG. 6, there is shown a schematic diagram of an SDN computer network 600 in accordance with an embodiment of the present invention. In one embodiment, the SDN computer network 600 is compliant with the OpenFlow™ protocol. Accordingly, in one embodiment, the SDN controller 610 comprises an OpenFlow™ controller and the SDN switch 620 comprises an OpenFlow™ switch. The SDN controller 610 and the SDN switch 620 comprise the control plane and data plane, respectively, of the SDN computer network 600. The SDN computer network 600 may have a plurality of SDN switches 620 but only one is shown for clarity of illustration. The SDN controller 610 and the SDN switch 620 are logically separate components.").<br><br>Lin '400 1:33-43 ("FIG. 1 shows a schematic diagram of an SDN computer network that is compliant with the OpenFlow™ protocol. Generally speaking, the OpenFlow™ protocol separates the control plane from the data plane. An OpenFlow™ controller serves as a control plane for making forwarding decisions based on flow policies, which may be stored in a flow policy database. The controller determines flow policies in conjunction with network forwarding setting and network topology. The flow policies may contain a condition and corresponding action to be performed when the condition is met. The action may specify how to manipulate a packet.").<br><br>Lin '400 1:44-54 ("An OpenFlow™ switch serves as the data plane that forwards packets, e.g., from an ingress port to an egress port, according to flow tables maintained by the data plane. The data plane is a replacement of traditional switches. When the data plane does not know how to manipulate a specific packet, the data plane may request the controller to receive a flow rule for the specific packet, and store the flow rule in the flow tables. Other packets that meet the same condition as the specific packet will be processed in accordance with the flow rule. The control plane may also actively insert flow rules into the flow tables.").<br><br>Lin '400 1:11-17 ("Software defined networking (SDN) is an emerging architecture for computer networking. Unlike traditional computer network architectures, SDN separates the control plane from the data plane. This provides many advantages, including relatively fast experimentation and optimization of switching and routing policies. SDN is applicable to both physical (i.e., real) and virtual computer networks."). |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 708 of 1100

| No. | '111 Patent Claim 31 | Lin '400 |
|---|---|---|
| | | Lin '400 2:47-65 ("FIG. 2 shows a schematic diagram of a computer system 100 that may be employed with embodiments of the present invention. The computer system 100 may be employed as a control plane and/or a data plane, for example. As another example, the computer system 100 may be employed to host a virtualization environment that supports a plurality of virtual machines. The computer system 100 may have fewer or more components to meet the needs of a particular application. The computer system 100 may include one or more processors 101. The computer system 100 may have one or more buses 103 coupling its various components. The computer system 100 may include one or more user input devices 102 (e.g., keyboard, mouse), one or more data storage devices 106 (e.g., hard drive, optical disk, Universal Serial Bus memory), a display monitor 104 (e.g., liquid crystal display, flat panel monitor), a computer network interface 105 (e.g., network adapter, modem), and a main memory 108 (e.g., random access memory). The computer network interface 105 may be coupled to a computer network 109."). |
| 31[c] | the network node communication with the controller serves as a control plane. | Lin '400 discloses the network node communication with the controller serves as a control plane.<br><br>For example, Lin '400 discloses that the controller comprises the control plane.<br><br>Lin '400 3:40-64 ("Referring now to FIG. 6, there is shown a schematic diagram of an SDN computer network 600 in accordance with an embodiment of the present invention. In one embodiment, the SDN computer network 600 is compliant with the OpenFlow™ protocol. Accordingly, in one embodiment, the SDN controller 610 comprises an OpenFlow™ controller and the SDN switch 620 comprises an OpenFlow™ switch. The SDN controller 610 and the SDN switch 620 comprise the control plane and data plane, respectively, of the SDN computer network 600. The SDN computer network 600 may have a plurality of SDN switches 620 but only one is shown for clarity of illustration. The SDN controller 610 and the SDN switch 620 are logically separate components.").<br><br>Lin '400 1:11-17 ("Software defined networking (SDN) is an emerging architecture for computer networking. Unlike traditional computer network architectures, SDN separates the control plane from the data plane. This provides many advantages, including relatively fast experimentation and optimization of switching and routing policies. SDN is applicable to both physical (i.e., real) and virtual computer networks."). |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 709 of 1100

---

**Chart for U.S. Patent 10,652,111 ("the '111 Patent")**
**U.S. Patent Publication No. 2013/0291088 to Shieh et al. ("Shieh '088")**

As shown in the chart below, all Asserted Claims of the '111 Patent are invalid under (1) AIA-35 U.S.C. § 102 (a) because Shieh '088 meets each element of those claims, and/or (2) 35 U.S.C. § 103 because Shieh '088 renders those claims obvious either alone, or in combination with the knowledge of a person having ordinary skill in the art, and in further combination with the references specifically identified below and in the following claim chart and/or one or more references identified in Defendant's Preliminary Invalidity Contentions. The following quotations and diagrams come from Shieh '088 titled "Cooperative Network Security Inspection", which was filed on Apr. 10, 2013, and published on Oct. 31, 2013.

Motivations to combine the disclosures in Shieh '088 with disclosures in other publications known in the art, as explained in this chart, include at least the similarity in subject matter between the references to the extent they concern methods relating to routing certain network traffic to entities for further analysis and inspection. Insofar as the references cite other patents or publications, or suggest additional changes, one of ordinary skill in the art would look beyond a single reference to other references in the field.

These invalidity contentions are based on Defendant's present understanding of the Asserted Claims, and Orckit's apparent construction of the claims in its November 3, 2022 Disclosure of Asserted Claims and Infringement Contentions Pursuant to P.R. 3-1, and Orckit's January 19, 2023 First Amended Disclosure of Asserted Claims and Infringement Contentions Pursuant to P.R. 3-1 (Orckit's "Infringement Disclosures"), which is deficient at least insofar as it fails to cite any documents or identify accused structures, acts, or materials in the Accused Products with particularity. Defendant does not agree with Orckit's application of the claims, or that the claims satisfy the requirements of 35 U.S.C. § 112. Defendant's contentions herein are not, and should in no way be seen as, admissions or adoptions as to any particular claim scope or construction, or as any admission that any particular element is met by any accused product in any particular way. Defendant objects to any attempt to imply claim construction from this chart. Defendant's prior art invalidity contentions are made in a variety of alternatives and do not represent Defendant's agreement or view as to the meaning, definiteness, written description support for, or enablement of any claim contained therein.



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 710 of 1100

The following contentions are subject to revision and amendment pursuant to Federal Rule of Civil Procedure 26(e), the Local Rules, and the Orders of record in this matter subject to further investigation and discovery regarding the prior art and the Court's construction of the claims at issue.

| No. | '111 Patent Claim 1 | Shieh '088 |
|---|---|---|
| 1[preamble] | A method for use with a packet network including a network node for transporting packets between first and second entities under control of a controller that is external to the network node, the method comprising: | Shieh '088 discloses a method for use with a packet network including a network node for transporting packets between first and second entities under control of a controller that is external to the network node.<br><br>For example, Shieh '088 discloses that it relates to a network system that operates on a packet from a source node destined to a destination mode controlled by a controller. Shieh '088 further discloses that the controller is external to the network access devices. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met.<br><br>Shieh '088 ¶ [0002] ("Embodiments of the present invention relate generally to network security. More particularly, embodiments of the invention relate to enabling network security with network equipment.").<br><br>Shieh '088 ¶ [0017] ("According to some embodiments, a mechanism is utilized to dynamically perform security inspection in a network. In one embodiment, the mechanism includes two functions: 1) an input/output (IO) function that performs the distribution of network traffic; and 2) a security-processing function that performs security processing, including security inspection and policy enforcement. The IO function receives the packets and uses a session table to forward the packets to the security-processing function. A session table is a data structure that stores connection states, including the destination of the security-processing function. In one embodiment, the IO function determines, based on an internal data structure such as a session or flow table, whether the packet should be forwarded to the security processing function for security inspection. The configuration of the IO function to control whether to forward the packets to the security processing function can be set based on a command received from an administrator or alternatively, based on a signal received from the security processing function.").<br><br>Shieh '088 ¶ [0018] ("According to one embodiment, an administrator can configure, for example, via a controller or a management entity, a network access device to set up a set of |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 711 of 1100

| No. | '111 Patent Claim 1 | Shieh '088 |
|---|---|---|
| | | filtering rules specifying whether and/or what types of packets should be forwarded to a security device and which of the security devices for security inspection. In this embodiment, the controller is configured to manage multiple network access devices and/or multiple security devices. Alternatively, a security device may inform a network access device that subsequent packets of a particular session should be forwarded from the network access device for security inspection. In one embodiment, a security device performs the security inspection at the beginning of the flow or session, and at a certain point, the security device decides that it no longer needs to inspect further packets of the same session."). <br><br> Shieh '088 ¶ [0021] ("According to one embodiment, network access device 204 is associated with a distributed firewall 212 that includes various firewall processing modules, for example, each being executed within a virtual machine (VM). In one embodiment, each firewall module is responsible for performing one or more firewall functions, but it does not include all of the firewall functions of a firewall. Examples of the firewall functions include, but are not limited to, network address translation (NAT), virtual private network (VPN), deep packet inspection (DPI), and/or anti-virus, etc. In one embodiment, some of the firewall processing modules are located within network access device 204 (e.g., firewall modules 209) and some are located external to network access device 204 (e.g., firewall modules 210 maintained by firewall processing node(s) 211, which may be a dedicated firewall processing machine. All of the firewall modules 209-210 are managed by firewall controller 208, which may be located within network access device 204, or external to network access device 204, such as, for example, in a public cloud associated with network 203, or in a private cloud associated with network 205. Controller 208 and firewall processing modules 209-210 collectively are referred to herein as distributed firewall 212."). <br><br> Shieh '088 ¶ [0023] ("According to one embodiment, a mechanism is utilized to dynamically perform security inspection in a network. In one embodiment, the mechanism includes two functions: 1) an input/output (IO) function (e.g., firewall module(s) 209) that performs the distribution of network traffic; and 2) a security-processing function (e.g., firewall module(s) 210) that performs security processing, including security inspection and policy enforcement. IO function 209 receives the packets and uses a session table to forward the packets to security-processing function 210. A session table is a data structure that stores connection states, including the destination of security-processing function. In one embodiment, IO function 209 determines, based on an internal data structure such as a session |

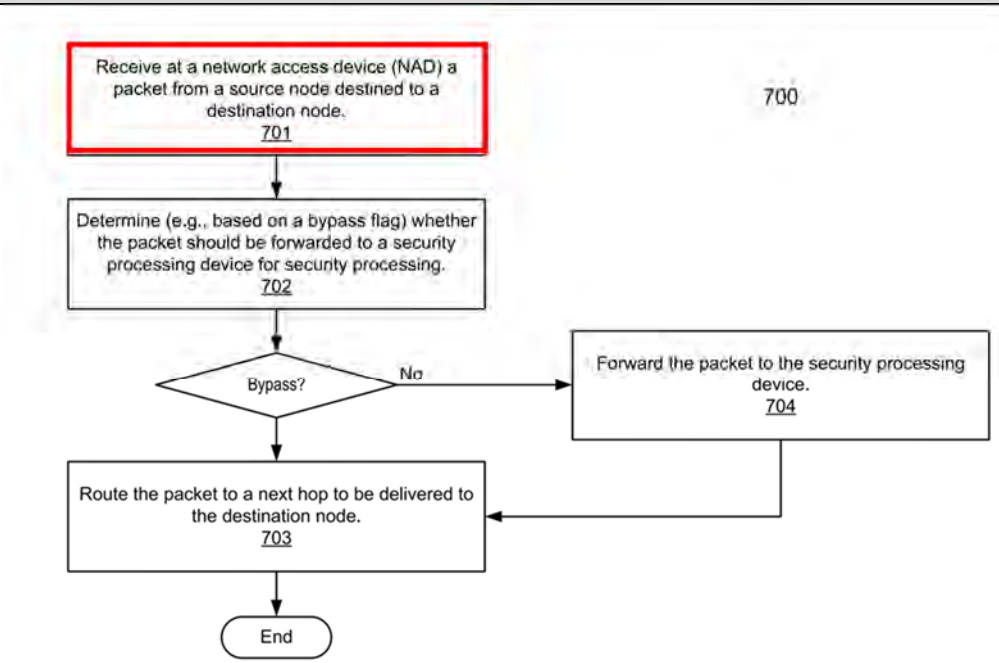


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 712 of 1100

| No. | '111 Patent Claim 1 | Shieh '088 |
|---|---|---|
| | | or flow table (e.g., session table as shown in FIG. 5), whether the packet should be forwarded to security processing function 210 for security inspection. The configuration of IO function 209 to control whether to forward the packets to security processing function 210 can be set based on a command received from an administrator or alternatively, based on a signal received from security processing function 210.").<br><br>Shieh '088 ¶ [0049] ("FIG. 7 is a flow diagram illustrating a method for performing firewall operations using a distributed firewall according to one embodiment of the invention. Method 700 may be performed by processing logic that may include software, hardware, or a combination of both. For example, method 700 may be performed by distributed firewall 212 of FIG. 1. Referring to FIG. 7, at block 701, a network access device receives a packet from a source node destined to a destination node. At block 702, the network access device determines whether the packet should be forwarded to a security device for security inspection. For example, processing logic may check whether there is an entry exists in a session table for the current session. If not, it may forward the packet to the security device for security processing at block 704. Alternatively, the processing logic may check whether there is a bypass flag set to a predetermined value for the current session. If there is, the packet will not be forwarded to the security device; instead, the packet will be directly routed to the destination node at block 703."). |

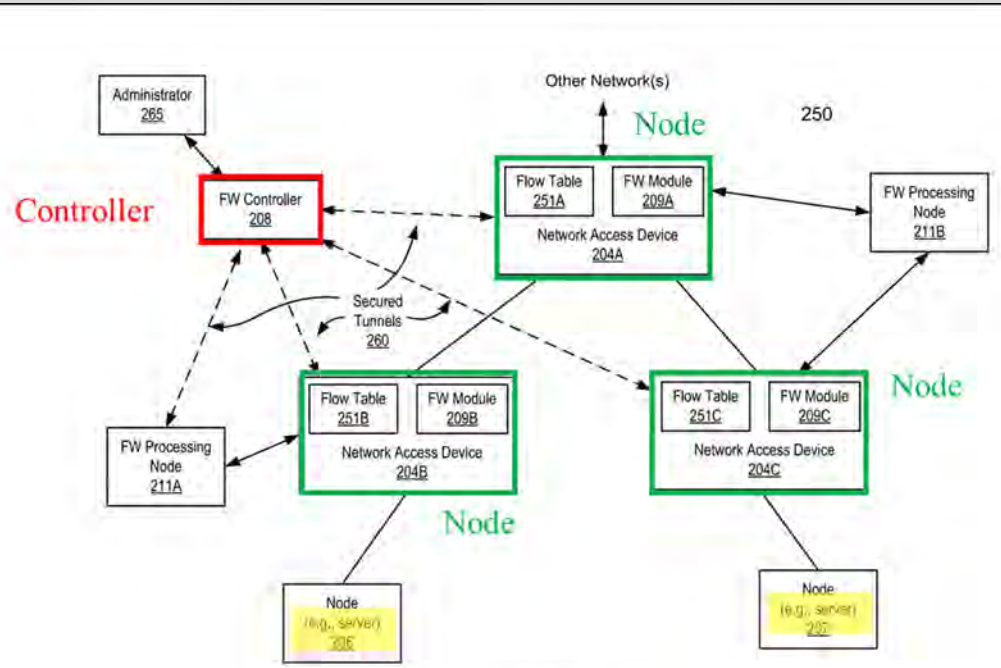


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 713 of 1100

| No. | '111 Patent Claim 1 | Shieh '088 |
|---|---|---|
| | |  <br>FIG. 7<br><br>Fig. 7 (annotation added)<br><br>Shieh '088 ¶ [0021] ("All of the firewall modules 209-210 are managed by firewall controller 208, which may be located within network access device 204, or external to network access device 204, such as, for example, in a public cloud associated with network 203, or in a private cloud associated with network 205."). |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 714 of 1100

| No. | '111 Patent Claim 1 | Shieh '088 |
|---|---|---|
| | |  <br>Fig. 2A (annotation added)<br><br>Shieh '088 ¶ [0042] ("In one embodiment, central controller 208 is the central place to control forwarding of the packets amongst I/O modules 301-304, security processing modules 309-311, and service processing modules 312-313. When a virtual I/O module receives a packet, according to one embodiment, it forwards the packet to central controller 208 if it cannot find an existing connection in its local cache, as shown in FIG. 5. When central controller 208 receives the packet, it decides which of security processing modules 309-311 is able to process the packets, and then forwards the packets to the designated security processing module. It also instructs the virtual I/O module to create the local cache to store connection state information so the subsequent packets of the same connection session do not need to be forwarded to central controller 208; rather, they can be directly forwarded to the proper security processing module identified in the cache."). |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 715 of 1100

| No. | '111 Patent Claim 1 | Shieh '088 |
|---|---|---|
| | | |
| 1[a] | sending, by the controller to the network node over the packet network, an instruction and a packet-applicable criterion; | Shieh '088 discloses sending, by the controller to the network node over the packet network, an instruction and a packet-applicable criterion.<br><br>For example, Shieh '088 discloses an external controller that controls network nodes in a packet network, and identifies a command sent by the controller to the network access devices through a persistent connection to set up a set of filtering rules concerning whether and/or what types of packets should be forwarded to a security device.<br><br>Shieh '088 ¶ [0017] ("According to some embodiments, a mechanism is utilized to dynamically perform security inspection in a network. In one embodiment, the mechanism includes two functions: 1) an input/output (IO) function that performs the distribution of network traffic; and 2) a security-processing function that performs security processing, including security inspection and policy enforcement. The IO function receives the packets and uses a session table to forward the packets to the security-processing function. A session table is a data structure that stores connection states, including the destination of the security-processing function. In one embodiment, the IO function determines, based on an internal data structure such as a session or flow table, whether the packet should be forwarded to the security processing function for security inspection. The configuration of the IO function to control whether to forward the packets to the security processing function can be set based on a command received from an administrator or alternatively, based on a signal received from the security processing function.").<br><br>Shieh '088 ¶ [0018] ("According to one embodiment, an administrator can configure, for example, via a controller or a management entity, a network access device to set up a set of filtering rules specifying whether and/or what types of packets should be forwarded to a security device and which of the security devices for security inspection. In this embodiment, the controller is configured to manage multiple network access devices and/or multiple security devices. Alternatively, a security device may inform a network access device that subsequent packets of a particular session should be forwarded from the network access device for security inspection. In one embodiment, a security device performs the security inspection at the beginning of the flow or session, and at a certain point, the security device decides that it no longer needs to inspect further packets of the same session.") |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 716 of 1100

| No. | '111 Patent Claim 1 | Shieh '088 |
|---|---|---|
| | | Shieh '088 ¶ [0023] ("According to one embodiment, a mechanism is utilized to dynamically perform security inspection in a network. In one embodiment, the mechanism includes two functions: 1) an input/output (IO) function (e.g., firewall module(s) 209) that performs the distribution of network traffic; and 2) a security-processing function (e.g., firewall module(s) 210) that performs security processing, including security inspection and policy enforcement. IO function 209 receives the packets and uses a session table to forward the packets to security-processing function 210. A session table is a data structure that stores connection states, including the destination of security-processing function. In one embodiment, IO function 209 determines, based on an internal data structure such as a session or flow table (e.g., session table as shown in FIG. 5), whether the packet should be forwarded to security processing function 210 for security inspection. The configuration of IO function 209 to control whether to forward the packets to security processing function 210 can be set based on a command received from an administrator or alternatively, based on a signal received from security processing function 210.")<br><br>Shieh '088 ¶ [0025] ("According to one embodiment, each of network access devices 204A-204C maintains a persistent connection such as secure connections or tunnels 260 with a controller or management entity 208 for exchanging management messages and configurations, or distributing routing information to network access devices 204A-204C, etc. In one embodiment, controller 208 communicates with each of the network access devices 204A-204C using a management protocol such as the OpenFlow™ protocol. OpenFlow is a Layer 2 communications protocol (e.g., media access control or MAC layer) that gives access to the forwarding plane of a network switch or router over the network. In simpler terms, OpenFlow allows the path of network packets through the network of switches to be determined by software running on multiple routers (minimum two of them, primary and secondary, having a role of observers). This separation of the control from the forwarding allows for more sophisticated traffic management than is feasible using access control lists (ACLs) and routing protocols.").<br><br>Shieh '088 ¶ [0026] ("The OpenFlow technology consists of three parts: flow tables installed on switches, a controller, and an OpenFlow protocol for the controller to talk securely with switches. Flow tables are set up on switches or routers. Controllers talk to the switches via the OpenFlow Protocol, which is secure, and impose policies on flows. For example, a simple |

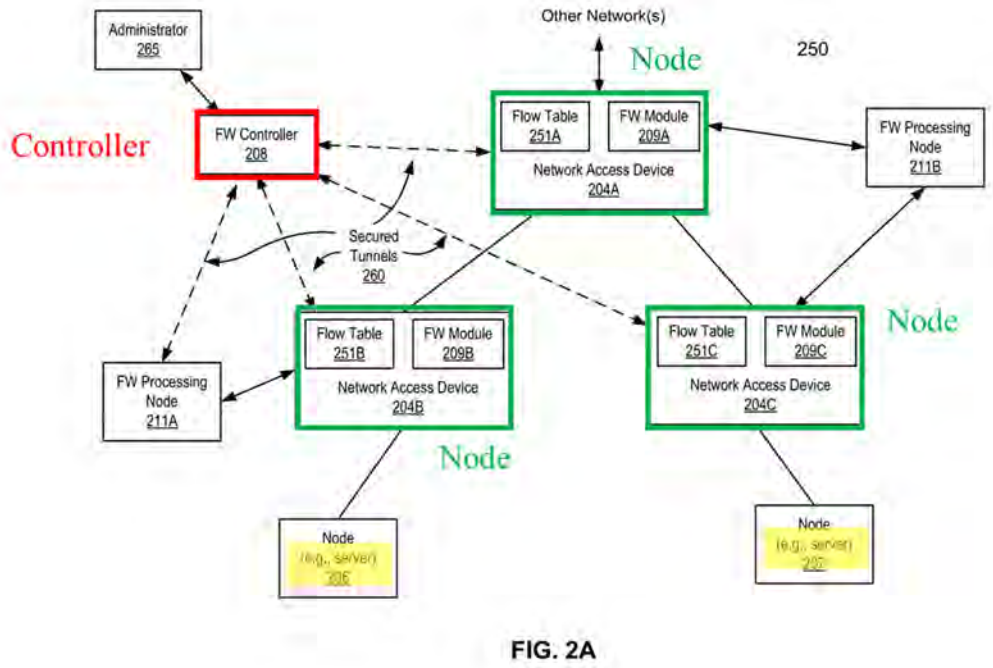


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 717 of 1100

| No. | '111 Patent Claim 1 | Shieh '088 |
|---|---|---|
| | | flow might be defined as any traffic from a given IP address. The rule governing it might be to route the flow through a given switch port. With its knowledge of the network, the controller could set up paths through the network optimized for speed, fewest number of hops or reduced latency, among other characteristics. Using OpenFlow takes control of how traffic flows through the network out of the hands of the infrastructure, the switches and routers, and puts it in the hands of the network owner (such as a corporation), individual users or individual applications.").<br><br>Shieh '088 ¶ [0028] ("Firewall modules 209A-209C may be part of a distributed firewall described above. For example, firewall modules 209A-209C may be the IO functions of a firewall while nodes 211A-211B may be firewall processing nodes. That is, modules 211A-211B may be dedicated firewall processing devices that perform some firewall processing operations such as DPI, content inspection, antivirus, etc., while firewall modules 209A-209C are responsible for routing data packets. For example, when firewall module 209B receives a packet from node 206, it may forward the packet to firewall processing node 211A for content inspection and/or forwards the packet to controller 208 for routing information. In response, firewall processing node 211A analyzes the received packet and/or further communicates with controller 208. Controller 208 may provide further routing information back to network access device 204B regarding how to route the packet. Each of the firewall processing nodes 211A-211B may further maintains a persistent connection or tunnel with controller 208, for example, using the OpenFlow communication protocol."). |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 718 of 1100

| No. | '111 Patent Claim 1 | Shieh '088 |
|---|---|---|
| | |  <br>FIG. 2A<br><br>Fig. 2A (annotation added)<br><br>Shieh '088 ¶ [0029] ("According to one embodiment, an administrator 265 configures, for example, via a controller or a management entity 208, a network access device (e.g., network access devices 204A-204C) to set up a set of filtering rules concerning whether and/or what types of packets should be forwarded to a security device and which of the security devices (e.g., security devices 211A-211B) for security inspection. In this embodiment, controller 208 is configured to manage multiple network access devices 204A-204C and/or multiple security devices 211A-211B. Alternatively, a security device, such as security device 211A, may inform a network access device, such as network access device 204B, whether subsequent packets of a particular session should be forwarded from the network access device for security inspection. A security device may perform the security inspection |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 719 of 1100

| No. | '111 Patent Claim 1 | Shieh '088 |
|---|---|---|
| | | on packets at the beginning of the flow or session, and at a certain point, the security device decides that it no longer needs to inspect further packets of the same session.").<br><br>Shieh '088 ¶ [0035] ("An embodiment of the invention also controls the communication between I/O functions and security-processing functions to enable packets to bypass security-processing function if there is no more need to inspect the packets of the connection. Some of the security functions do not need to inspect all the packets of a connection. For examples, to identify the application of a connection, there may be only need to inspect first four or five packets to make the identification. In this case, the security-processing function can notify I/O functions to bypass the security-processing function for the rest of the packets of the connections. Once the I/O function receives the notification, it will forward the packets out without redirecting the packets to the security-processing functions. This would greatly improve the performance even when security inspection is turned on.").<br><br>Shieh '088 ¶ [0036] ("During the bypass phase, the I/O function may notify the security-processing function if there are special events in the packet stream. These events could be receipt of TCP FIN or TCP RST packets, or not receiving any packets of the connection within a time threshold. The notification from I/O functions to security processing functions could help to clean up the state in the security-processing nodes.").<br><br>Shieh '088 ¶ [0049] ("In one embodiment, firewall modules 300A-300B could be distributed in different networks, even on different locations, as long as the modules can reach the module that is next in terms of processing and the central controller. In one embodiment, virtual I/O modules and corresponding security processing modules are in a public cloud and the central controller is in a private cloud. This configuration may provide the flexibility to secure and control packets coming from the public cloud, and allow central controller having overall view of traffic from Internet as well as from internal network."). |
| 1[b] | receiving, by the network node from the controller, the instruction and the criterion; | Shieh '088 discloses receiving, by the network node from the controller, the instruction and the criterion.<br><br>For example, Shieh '088 discloses that a command is received by the network access devices from the controller to set up a set of filtering rules concerning whether and/or what types of packets should be forwarded to a security device. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 720 of 1100

| No. | '111 Patent Claim 1 | Shieh '088 |
|---|---|---|
| | | *See supra at* 1[a].<br><br>Shieh '088 ¶ [0023] ("According to one embodiment, a mechanism is utilized to dynamically perform security inspection in a network. In one embodiment, the mechanism includes two functions: 1) an input/output (IO) function (e.g., firewall module(s) 209) that performs the distribution of network traffic; and 2) a security-processing function (e.g., firewall module(s) 210) that performs security processing, including security inspection and policy enforcement. IO function 209 receives the packets and uses a session table to forward the packets to security-processing function 210. A session table is a data structure that stores connection states, including the destination of security-processing function. In one embodiment, IO function 209 determines, based on an internal data structure such as a session or flow table (e.g., session table as shown in FIG. 5), whether the packet should be forwarded to security processing function 210 for security inspection. The configuration of IO function 209 to control whether to forward the packets to security processing function 210 can be set based on a command received from an administrator or alternatively, based on a signal received from security processing function 210.")<br><br>Shieh '088 ¶ [0028] ("Firewall modules 209A-209C may be part of a distributed firewall described above. For example, firewall modules 209A-209C may be the IO functions of a firewall while nodes 211A-211B may be firewall processing nodes. That is, modules 211A-211B may be dedicated firewall processing devices that perform some firewall processing operations such as DPI, content inspection, antivirus, etc., while firewall modules 209A-209C are responsible for routing data packets. For example, when firewall module 209B receives a packet from node 206, it may forward the packet to firewall processing node 211A for content inspection and/or forwards the packet to controller 208 for routing information. In response, firewall processing node 211A analyzes the received packet and/or further communicates with controller 208. Controller 208 may provide further routing information back to network access device 204B regarding how to route the packet. Each of the firewall processing nodes 211A-211B may further maintains a persistent connection or tunnel with controller 208, for example, using the OpenFlow communication protocol.").<br><br>Shieh '088 ¶ [0029] ("According to one embodiment, an administrator 265 configures, for example, via a controller or a management entity 208, a network access device (e.g., network |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 721 of 1100

| No. | '111 Patent Claim 1 | Shieh '088 |
|---|---|---|
| | | access devices 204A-204C) to set up a set of filtering rules concerning whether and/or what types of packets should be forwarded to a security device and which of the security devices (e.g., security devices 211A-211B) for security inspection. In this embodiment, controller 208 is configured to manage multiple network access devices 204A-204C and/or multiple security devices 211A-211B. Alternatively, a security device, such as security device 211A, may inform a network access device, such as network access device 204B, whether subsequent packets of a particular session should be forwarded from the network access device for security inspection. A security device may perform the security inspection on packets at the beginning of the flow or session, and at a certain point, the security device decides that it no longer needs to inspect further packets of the same session."). |
| | | Shieh '088 Claim 1 ("A computer-implemented method, comprising:<br>receiving at a network access device a packet from a source node destined to a destination node;<br>examining a data structure maintained by the network access device to determine whether the data structure stores a data member having a predetermined value, the data member indicating whether the packet should undergo security processing;<br>if the data member matches the predetermined value, transmitting the packet to a security device associated with the network access device to allow the security device to perform content inspection, and<br>in response to a response received from the security device, routing the packet to the destination node dependent upon the response; and<br>transmitting the packet to the destination node without forwarding the packet to the security device, if the data member does not match the predetermined value."). |
| | | Shieh '088 Claim 17 ("The system of claim 15, further comprising a controller to manage the network access device and the security device, wherein the network access device is further to<br>receive a message having a data value from the controller, the data value indicating whether the network access device should forward further packets to the security device for security inspection, and<br>store the data value in the data member of the data structure."). |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 722 of 1100

| No. | '111 Patent Claim 1 | Shieh '088 |
|---|---|---|
| 1[c] | receiving, by the network node from the first entity over the packet network, a packet addressed to the second entity; | Shieh '088 discloses receiving, by the network node from the first entity over the packet network, a packet addressed to the second entity.<br><br>For example, Shieh '088 discloses that a network access device receives a packet from a source node that is addressed to a destination node where the method is used to process a network flow.<br><br>Shieh '088 Claim 1 ("A computer-implemented method, comprising:<br>receiving at a network access device a packet from a source node destined to a destination node;<br>examining a data structure maintained by the network access device to determine whether the data structure stores a data member having a predetermined value, the data member indicating whether the packet should undergo security processing;<br>if the data member matches the predetermined value, transmitting the packet to a security device associated with the network access device to allow the security device to perform content inspection, and<br>in response to a response received from the security device, routing the packet to the destination node dependent upon the response; and<br>transmitting the packet to the destination node without forwarding the packet to the security device, if the data member does not match the predetermined value.").<br><br>Shieh '088 ¶ [0027] ("Referring back to FIG. 2A, in one embodiment, each of the network access devices 204A-204C maintains a flow table or session table (e.g., flow tables 251A-251C) and a firewall module (e.g., 209A-209C). A network flow refers to a sequence of packets from a source computer to a destination, which may be another host, a multicast group, or a broadcast domain. For example, a TCP/IP flow can be uniquely identified by the following parameters within a certain time period: 1) Source and Destination IP address; 2) Source and Destination Port; and 3) Layer 4 Protocol (TCP/UDP/ICMP). A session is a semi-permanent interactive information interchange, also known as a dialogue, a conversation or a meeting, between two or more communicating devices. A session is set up or established at a certain point in time and torn down at a later point in time. An established communication session may involve more than one message in each direction. A session is typically, but not always, stateful, meaning that at least one of the communicating entities needs to save information about the session history in order to be able to communicate, as opposed to |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 723 of 1100

| No. | '111 Patent Claim 1 | Shieh '088 |
|---|---|---|
| | | stateless communication, where the communication consists of independent requests with responses. Flow tables 251A-251C may be implemented as a combination of a flow table and a session table.").<br><br>Shieh '088 ¶ [0020] ("FIG. 1 is a block diagram illustrating an example of network configuration according to one embodiment of the invention. Referring to FIG. 1, network access device 204, which may be a router or gateway, a switch or an access point, etc., provides an interface between network 203 and network 205. Network 203 may be an external network such as a wide area network (WAN) (e.g., Internet) while network 205 represents a local area network (LAN). Nodes 206-207 go through gateway device 204 in order to reach nodes 201-202, or vice versa. Any of nodes 201-202 and 206-207 may be a client device (e.g., a desktop, laptop, Smartphone, gaming device) or a server.").<br><br>Shieh '088 ¶ [0037] ("FIG. 2B is a processing flow diagram illustrating a process of security inspection according to one embodiment of the invention. Referring to FIG. 2B, as an example, network switch 272 may represent any of network access devices 204A-204C and security device 273 may represents any of security processing devices 211A-211B as described above with respect to FIG. 2A. When device 272 receives a packet from a source node 271 via transaction 281, device 272 may determine whether the packet should be forwarded to security device 273. For example, device 272 may look up in its session table such as the one as shown in FIG. 5 to determine whether a bypass flag has been set to a predetermined value. If the bypass flag matches the predetermined value, the packet is forwarded to security device 273 via path 282; otherwise, the packet is routed to destination node 274. Alternatively, if there is no entry in the session table corresponding to the current session, the packet will also be transmitted to security device 273. After network device 272 receives a response from security device 273 via path 283, dependent upon the response, the packet may then be routed to destination node 274 via path 284. These processes may continue until a notification is received from security device 273 via path 285 indicating that it no longer wishes to receive further packets of the same session for inspection, such that subsequent packets will be directly routed to destination node 274 via path 286 without routing to security device 273. If there are certain events that have been registered from security device 273, network device 272 may notify security device 274 via path 287 upon detecting the registered events."). |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 724 of 1100

| No. | '111 Patent Claim 1 | Shieh '088 |
|---|---|---|
| | | Shieh '088 ¶ [0049] ("FIG. 7 is a flow diagram illustrating a method for performing firewall operations using a distributed firewall according to one embodiment of the invention. Method 700 may be performed by processing logic that may include software, hardware, or a combination of both. For example, method 700 may be performed by distributed firewall 212 of FIG. 1. Referring to FIG. 7, at block 701, a network access device receives a packet from a source node destined to a destination node. At block 702, the network access device determines whether the packet should be forwarded to a security device for security inspection. For example, processing logic may check whether there is an entry exists in a session table for the current session. If not, it may forward the packet to the security device for security processing at block 704. Alternatively, the processing logic may check whether there is a bypass flag set to a predetermined value for the current session. If there is, the packet will not be forwarded to the security device; instead, the packet will be directly routed to the destination node at block 703."). |
| 1[d] | checking, by the network node, if the packet satisfies the criterion; | Shieh '088 discloses checking, by the network node, if the packet satisfies the criterion.<br><br>For example, Shieh '088 discloses a network access device that uses filtering rules concerning whether and/or what types of packets should be forwarded to a security device.<br><br>Shieh '088 ¶ [0039] "(An I/O module running within a virtual machine is referred to herein as a virtual I/O module. Each of virtual I/O modules 301-304 receives packets from any of servers 321-324 of LAN 320 and sends packets to external network 315 outside of the firewall. In one embodiment, each of I/O modules 301-304 keeps a local cache (e.g., caches 305-308) storing location(s) of a security processing module(s) (e.g., security processing modules 309-311) for each connection session. A cache maintained by each I/O module contains a forwarding table mapping certain connection sessions to any of security modules 309-311. An example of a forwarding table is shown in FIG. 5. Upon receiving a packet, an I/O module performs a packet classification to find out the associated connection and forwards the packet to the corresponding security processing module identified by the forwarding table. If it cannot find the connection in its local cache, the packets are forwarded to central controller 208 for processing. In such a case, controller 208 assigns the connection to one of security processing modules 309-311 based on one or more of a variety of factors such as load balancing. The virtual I/O modules 302-304 can be located at multiple locations of the networks to receive and send out packets."). |

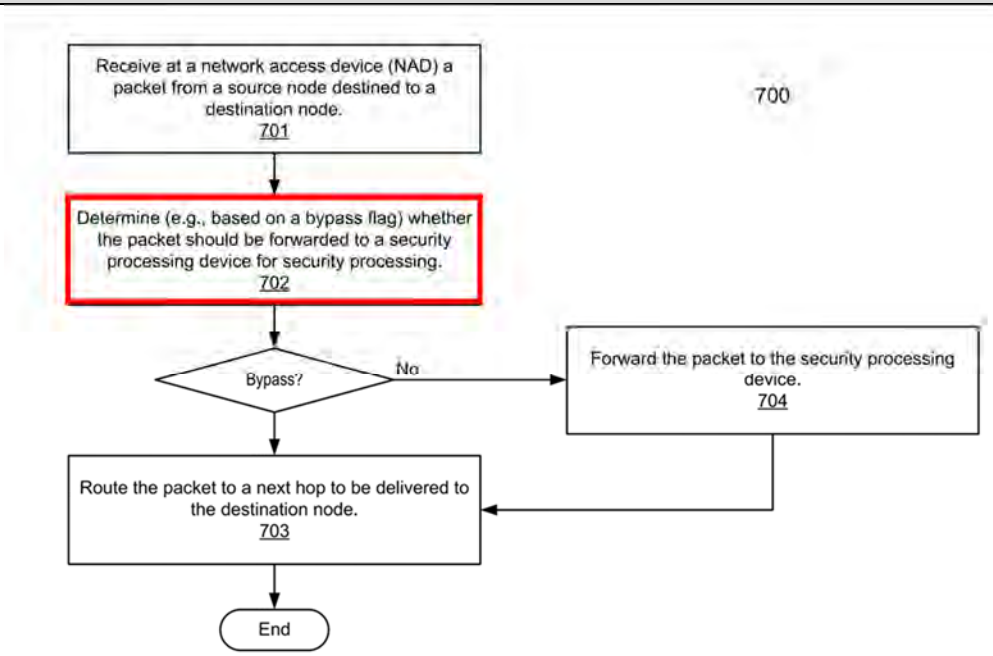


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 725 of 1100

| No. | ’111 Patent Claim 1 | Shieh ’088 |
|---|---|---|
| | | Shieh ’088 Claim 1 (“A computer-implemented method, comprising:<br>receiving at a network access device a packet from a source node destined to a destination node;<br>examining a data structure maintained by the network access device to determine whether the data structure stores a data member having a predetermined value, the data member indicating whether the packet should undergo security processing;<br>if the data member matches the predetermined value, transmitting the packet to a security device associated with the network access device to allow the security device to perform content inspection, and<br>in response to a response received from the security device, routing the packet to the destination node dependent upon the response; and<br>transmitting the packet to the destination node without forwarding the packet to the security device, if the data member does not match the predetermined value.”). |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 726 of 1100

| No. | '111 Patent Claim 1 | Shieh '088 |
|---|---|---|
| | |  <br>FIG. 7<br><br>Fig. 7 (annotation added)<br><br>Shieh '088 ¶ [0018] ("According to one embodiment, an administrator can configure, for example, via a controller or a management entity, a network access device to set up a set of filtering rules specifying whether and/or what types of packets should be forwarded to a security device and which of the security devices for security inspection. In this embodiment, the controller is configured to manage multiple network access devices and/or multiple security devices. Alternatively, a security device may inform a network access device that subsequent packets of a particular session should be forwarded from the network access device for security inspection. In one embodiment, a security device performs the security inspection at the beginning of the flow or session, and at a certain point, the security device decides that it no longer needs to inspect further packets of the same session."). |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 727 of 1100

| No. | '111 Patent Claim 1 | Shieh '088 |
|---|---|---|
| | | Shieh '088 ¶ [0023] ("According to one embodiment, a mechanism is utilized to dynamically perform security inspection in a network. In one embodiment, the mechanism includes two functions: 1) an input/output (IO) function (e.g., firewall module(s) 209) that performs the distribution of network traffic; and 2) a security-processing function (e.g., firewall module(s) 210) that performs security processing, including security inspection and policy enforcement. IO function 209 receives the packets and uses a session table to forward the packets to security-processing function 210. A session table is a data structure that stores connection states, including the destination of security-processing function. In one embodiment, IO function 209 determines, based on an internal data structure such as a session or flow table (e.g., session table as shown in FIG. 5), whether the packet should be forwarded to security processing function 210 for security inspection. The configuration of IO function 209 to control whether to forward the packets to security processing function 210 can be set based on a command received from an administrator or alternatively, based on a signal received from security processing function 210.")<br><br>Shieh '088 ¶ [0028] ("Firewall modules 209A-209C may be part of a distributed firewall described above. For example, firewall modules 209A-209C may be the IO functions of a firewall while nodes 211A-211B may be firewall processing nodes. That is, modules 211A-211B may be dedicated firewall processing devices that perform some firewall processing operations such as DPI, content inspection, antivirus, etc., while firewall modules 209A-209C are responsible for routing data packets. For example, when firewall module 209B receives a packet from node 206, it may forward the packet to firewall processing node 211A for content inspection and/or forwards the packet to controller 208 for routing information. In response, firewall processing node 211A analyzes the received packet and/or further communicates with controller 208. Controller 208 may provide further routing information back to network access device 204B regarding how to route the packet. Each of the firewall processing nodes 211A-211B may further maintains a persistent connection or tunnel with controller 208, for example, using the OpenFlow communication protocol.").<br><br>Shieh '088 ¶ [0029] ("According to one embodiment, an administrator 265 configures, for example, via a controller or a management entity 208, a network access device (e.g., network access devices 204A-204C) to set up a set of filtering rules concerning whether and/or what types of packets should be forwarded to a security device and which of the security devices (e.g., security devices 211A-211B) for security inspection. In this embodiment, |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 728 of 1100

| No. | '111 Patent Claim 1 | Shieh '088 |
|---|---|---|
| | | controller 208 is configured to manage multiple network access devices 204A-204C and/or multiple security devices 211A-211B. Alternatively, a security device, such as security device 211A, may inform a network access device, such as network access device 204B, whether subsequent packets of a particular session should be forwarded from the network access device for security inspection. A security device may perform the security inspection on packets at the beginning of the flow or session, and at a certain point, the security device decides that it no longer needs to inspect further packets of the same session."). |
| 1[e] | responsive to the packet not satisfying the criterion, sending, by the network node over the packet network, the packet to the second entity; and | Shieh '088 discloses responsive to the packet not satisfying the criterion, sending, by the network node over the packet network, the packet to the second entity.<br><br>Shieh '088 discloses transmitting the packet to the destination node without forwarding the packet to the security device, if the data member does not match the predetermined value.<br><br>Shieh '088 ¶ [0042] ("In one embodiment, central controller 208 is the central place to control forwarding of the packets amongst I/O modules 301-304, security processing modules 309-311, and service processing modules 312-313. When a virtual I/O module receives a packet, according to one embodiment, it forwards the packet to central controller 208 if it cannot find an existing connection in its local cache, as shown in FIG. 5. When central controller 208 receives the packet, it decides which of security processing modules 309-311 is able to process the packets, and then forwards the packets to the designated security processing module. It also instructs the virtual I/O module to create the local cache to store connection state information so the subsequent packets of the same connection session do not need to be forwarded to central controller 208; rather, they can be directly forwarded to the proper security processing module identified in the cache.").<br><br>Shieh '088 ¶ [0039] "(An I/O module running within a virtual machine is referred to herein as a virtual I/O module. Each of virtual I/O modules 301-304 receives packets from any of servers 321-324 of LAN 320 and sends packets to external network 315 outside of the firewall. In one embodiment, each of I/O modules 301-304 keeps a local cache (e.g., caches 305-308) storing location(s) of a security processing module(s) (e.g., security processing modules 309-311) for each connection session. A cache maintained by each I/O module contains a forwarding table mapping certain connection sessions to any of security modules 309-311. An example of a forwarding table is shown in FIG. 5. Upon receiving a |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 729 of 1100

| No. | '111 Patent Claim 1 | Shieh '088 |
|---|---|---|
| | | packet, an I/O module performs a packet classification to find out the associated connection and forwards the packet to the corresponding security processing module identified by the forwarding table. If it cannot find the connection in its local cache, the packets are forwarded to central controller 208 for processing. In such a case, controller 208 assigns the connection to one of security processing modules 309-311 based on one or more of a variety of factors such as load balancing. The virtual I/O modules 302-304 can be located at multiple locations of the networks to receive and send out packets.").<br><br>Shieh '088 ¶ [0035] "(An embodiment of the invention also controls the communication between I/O functions and security-processing functions to enable packets to bypass security-processing function if there is no more need to inspect the packets of the connection. Some of the security functions do not need to inspect all the packets of a connection. For examples, to identify the application of a connection, there may be only need to inspect first four or five packets to make the identification. In this case, the security-processing function can notify I/O functions to bypass the security-processing function for the rest of the packets of the connections. Once the I/O function receives the notification, it will forward the packets out without redirecting the packets to the security-processing functions. This would greatly improve the performance even when security inspection is turned on.").<br><br>Shieh '088 ¶ [0037] "(FIG. 2B is a processing flow diagram illustrating a process of security inspection according to one embodiment of the invention. Referring to FIG. 2B, as an example, network switch 272 may represent any of network access devices 204A-204C and security device 273 may represents any of security processing devices 211A-211B as described above with respect to FIG. 2A. When device 272 receives a packet from a source node 271 via transaction 281, device 272 may determine whether the packet should be forwarded to security device 273. For example, device 272 may look up in its session table such as the one as shown in FIG. 5 to determine whether a bypass flag has been set to a predetermined value. If the bypass flag matches the predetermined value, the packet is forwarded to security device 273 via path 282; otherwise, the packet is routed to destination node 274. Alternatively, if there is no entry in the session table corresponding to the current session, the packet will also be transmitted to security device 273. After network device 272 receives a response from security device 273 via path 283, dependent upon the response, the packet may then be routed to destination node 274 via path 284. These processes may continue until a notification is received from security device 273 via path 285 indicating |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 730 of 1100

| No. | '111 Patent Claim 1 | Shieh '088 |
|---|---|---|
| | | that it no longer wishes to receive further packets of the same session for inspection, such that subsequent packets will be directly routed to destination node 274 via path 286 without routing to security device 273. If there are certain events that have been registered from security device 273, network device 272 may notify security device 274 via path 287 upon detecting the registered events."). |

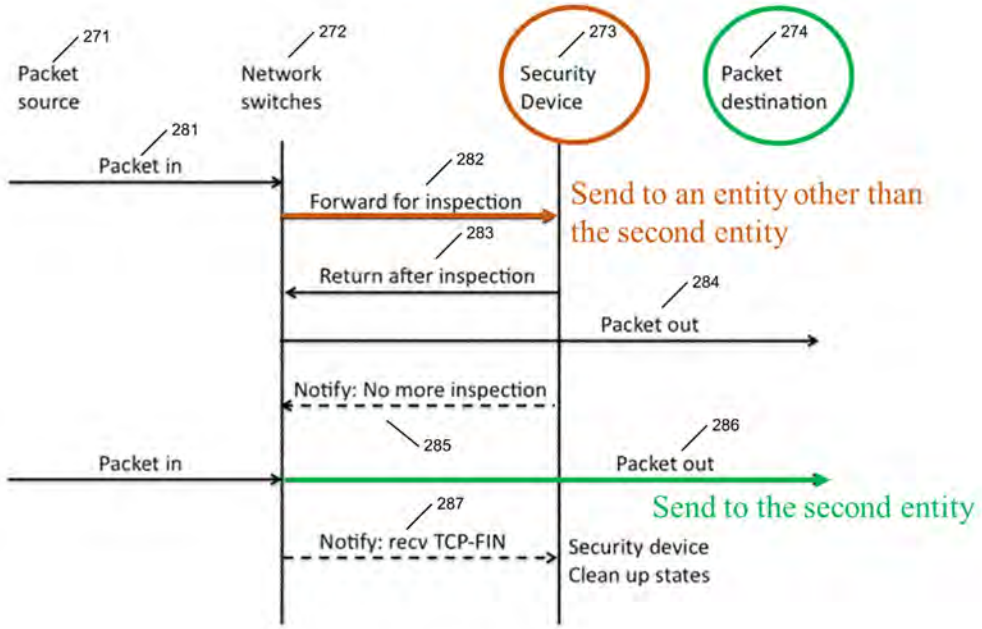


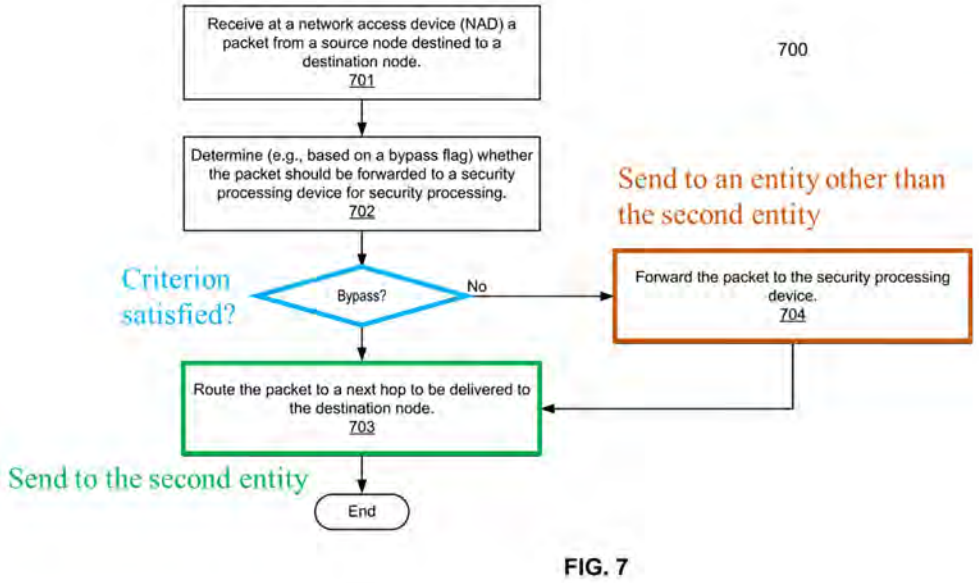

FIG. 2B

Fig. 2B (annotation added)

Shieh '088 Claim 1 ("A computer-implemented method, comprising:
receiving at a network access device a packet from a source node destined to a destination node;



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 731 of 1100

| No. | '111 Patent Claim 1 | Shieh '088 |
|---|---|---|
| | | examining a data structure maintained by the network access device to determine whether the data structure stores a data member having a predetermined value, the data member indicating whether the packet should undergo security processing;<br>if the data member matches the predetermined value, transmitting the packet to a security device associated with the network access device to allow the security device to perform content inspection, and<br>in response to a response received from the security device, routing the packet to the destination node dependent upon the response; and<br>transmitting the packet to the destination node without forwarding the packet to the security device, if the data member does not match the predetermined value.").<br>  <br><br>FIG. 7<br>Fig. 7 (annotation added) |
| 1[f] | responsive to the packet satisfying the criterion, sending the | Shieh '088 discloses responsive to the packet satisfying the criterion, sending the packet, by the network node over the packet network, to an entity that is included in the instruction and is other than the second entity. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 732 of 1100

| No. | '111 Patent Claim 1 | Shieh '088 |
|---|---|---|
| | packet, by the network node over the packet network, to an entity that is included in the instruction and is other than the second entity. | For example, Shieh '088 discloses an embodiment in which a "virtual I/O module" functions as a network nodes that receives packets from servers and sends packets to an external network. Shieh '088 further discloses sending the packet to the controller if it cannot find an existing connection in its local cache. Shieh '088 discloses bypass rules, such as when the bypass flag matches a predetermined value, that when satisfied, instructs that the packet will be sent from the virtual I/O module to a central controller capable of deep packet inspection responsive to the instruction.<br><br>Shieh '088 ¶ [0042] ("In one embodiment, central controller 208 is the central place to control forwarding of the packets amongst I/O modules 301-304, security processing modules 309-311, and service processing modules 312-313. When a virtual I/O module receives a packet, according to one embodiment, it forwards the packet to central controller 208 if it cannot find an existing connection in its local cache, as shown in FIG. 5. When central controller 208 receives the packet, it decides which of security processing modules 309-311 is able to process the packets, and then forwards the packets to the designated security processing module. It also instructs the virtual I/O module to create the local cache to store connection state information so the subsequent packets of the same connection session do not need to be forwarded to central controller 208; rather, they can be directly forwarded to the proper security processing module identified in the cache.").<br><br>Shieh '088 ¶ [0039] "(An I/O module running within a virtual machine is referred to herein as a virtual I/O module. Each of virtual I/O modules 301-304 receives packets from any of servers 321-324 of LAN 320 and sends packets to external network 315 outside of the firewall. In one embodiment, each of I/O modules 301-304 keeps a local cache (e.g., caches 305-308) storing location(s) of a security processing module(s) (e.g., security processing modules 309-311) for each connection session. A cache maintained by each I/O module contains a forwarding table mapping certain connection sessions to any of security modules 309-311. An example of a forwarding table is shown in FIG. 5. Upon receiving a packet, an I/O module performs a packet classification to find out the associated connection and forwards the packet to the corresponding security processing module identified by the forwarding table. If it cannot find the connection in its local cache, the packets are forwarded to central controller 208 for processing. In such a case, controller 208 assigns the connection to one of security processing modules 309-311 based on one or more of a variety of factors, |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 733 of 1100

| No. | '111 Patent Claim 1 | Shieh '088 |
|---|---|---|
| | | such as load balancing. The virtual I/O modules 302-304 can be located at multiple locations of the networks to receive and send out packets.").<br><br>Shieh '088 ¶ [0035] "(An embodiment of the invention also controls the communication between I/O functions and security-processing functions to enable packets to bypass security-processing function if there is no more need to inspect the packets of the connection. Some of the security functions do not need to inspect all the packets of a connection. For examples, to identify the application of a connection, there may be only need to inspect first four or five packets to make the identification. In this case, the security-processing function can notify I/O functions to bypass the security-processing function for the rest of the packets of the connections. Once the I/O function receives the notification, it will forward the packets out without redirecting the packets to the security-processing functions. This would greatly improve the performance even when security inspection is turned on.").<br><br>Shieh '088 ¶ [0036] "(During the bypass phase, the I/O function may notify the security-processing function if there are special events in the packet stream. These events could be receipt of TCP FIN or TCP RST packets, or not receiving any packets of the connection within a time threshold. The notification from I/O functions to security processing functions could help to clean up the state in the security-processing nodes.").<br><br>Shieh '088 ¶ [0037] "(FIG. 2B is a processing flow diagram illustrating a process of security inspection according to one embodiment of the invention. Referring to FIG. 2B, as an example, network switch 272 may represent any of network access devices 204A-204C and security device 273 may represents any of security processing devices 211A-211B as described above with respect to FIG. 2A. When device 272 receives a packet from a source node 271 via transaction 281, device 272 may determine whether the packet should be forwarded to security device 273. For example, device 272 may look up in its session table such as the one as shown in FIG. 5 to determine whether a bypass flag has been set to a predetermined value. If the bypass flag matches the predetermined value, the packet is forwarded to security device 273 via path 282; otherwise, the packet is routed to destination node 274. Alternatively, if there is no entry in the session table corresponding to the current session, the packet will also be transmitted to security device 273. After network device 272 receives a response from security device 273 via path 283, dependent upon the response, the packet may then be routed to destination node 274 via path 284. These processes |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 734 of 1100

| No. | '111 Patent Claim 1 | Shieh '088 |
|---|---|---|
| | | may continue until a notification is received from security device 273 via path 285 indicating that it no longer wishes to receive further packets of the same session for inspection, such that subsequent packets will be directly routed to destination node 274 via path 286 without routing to security device 273. If there are certain events that have been registered from security device 273, network device 272 may notify security device 274 via path 287 upon detecting the registered events."). |

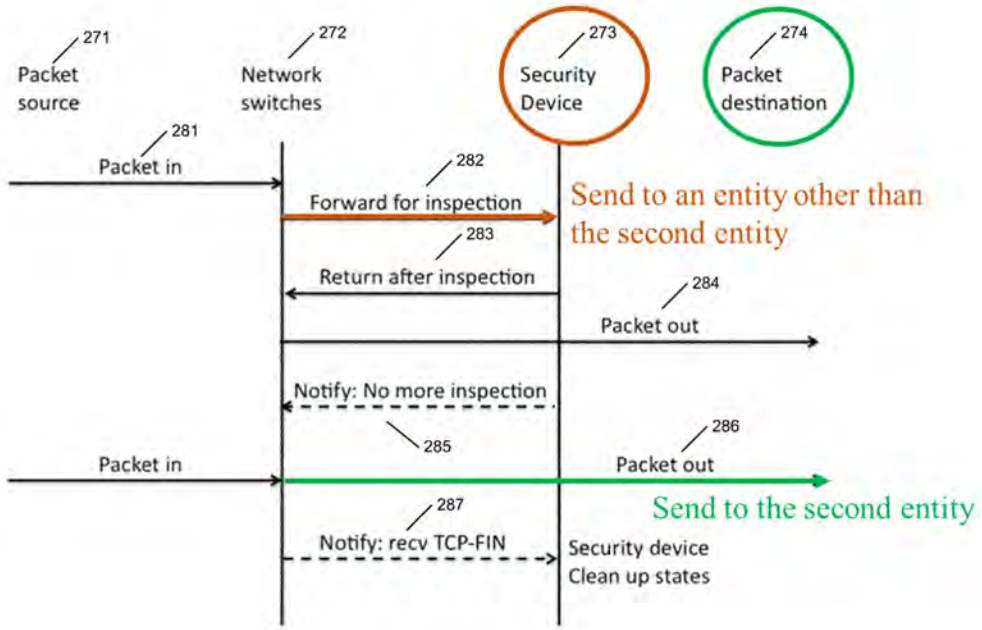



FIG. 2B

Fig. 2B (annotation added)

Shieh '088 ¶ [0021] "(According to one embodiment, network access device 204 is associated with a distributed firewall 212 that includes various firewall processing modules, for example, each being executed within a virtual machine (VM). In one embodiment, each firewall module is responsible for performing one or more firewall functions, but it does not include all of the firewall functions of a firewall. Examples of the firewall functions include



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 735 of 1100

| No. | '111 Patent Claim 1 | Shieh '088 |
|---|---|---|
| | | but are not limited to, network address translation (NAT), virtual private network (VPN), deep packet inspection (DPI), and/or anti-virus, etc. In one embodiment, some of the firewall processing modules are located within network access device 204 (e.g., firewall modules 209) and some are located external to network access device 204 (e.g., firewall modules 210 maintained by firewall processing node(s) 211, which may be a dedicated firewall processing machine. All of the firewall modules 209-210 are managed by firewall controller 208, which may be located within network access device 204, or external to network access device 204, such as, for example, in a public cloud associated with network 203, or in a private cloud associated with network 205. Controller 208 and firewall processing modules 209-210 collectively are referred to herein as distributed firewall 212.").<br><br>Shieh '088 ¶ [0028] "(Firewall modules 209A-209C may be part of a distributed firewall described above. For example, firewall modules 209A-209C may be the IO functions of a firewall while nodes 211A-211B may be firewall processing nodes. That is, modules 211A-211B may be dedicated firewall processing devices that perform some firewall processing operations such as DPI, content inspection, antivirus, etc., while firewall modules 209A-209C are responsible for routing data packets. For example, when firewall module 209B receives a packet from node 206, it may forward the packet to firewall processing node 211A for content inspection and/or forwards the packet to controller 208 for routing information. In response, firewall processing node 211A analyzes the received packet and/or further communicates with controller 208. Controller 208 may provide further routing information back to network access device 204B regarding how to route the packet. Each of the firewall processing nodes 211A-211B may further maintains a persistent connection or tunnel with controller 208, for example, using the OpenFlow communication protocol.").<br><br>Shieh '088 ¶ [0040] "(In one embodiment, each of security processing modules 309-311 performs major security processing functions, such as, for example, NAT, VPN, DPI, and/or anti-virus, etc. A security processing module receives packets and runs the packets through one or more various security functions in the module for security processing. There could be several security modules and each handles the same or different security functions. If the packets need to go through another security or service processing, the module sends the packets to the other modules. Optionally, it can run the packets through a load balancing mechanism to distribute the load to multiple modules. If a module is the last processing module in the chain to process the packets, it can forward the packets back to the virtual I/O |

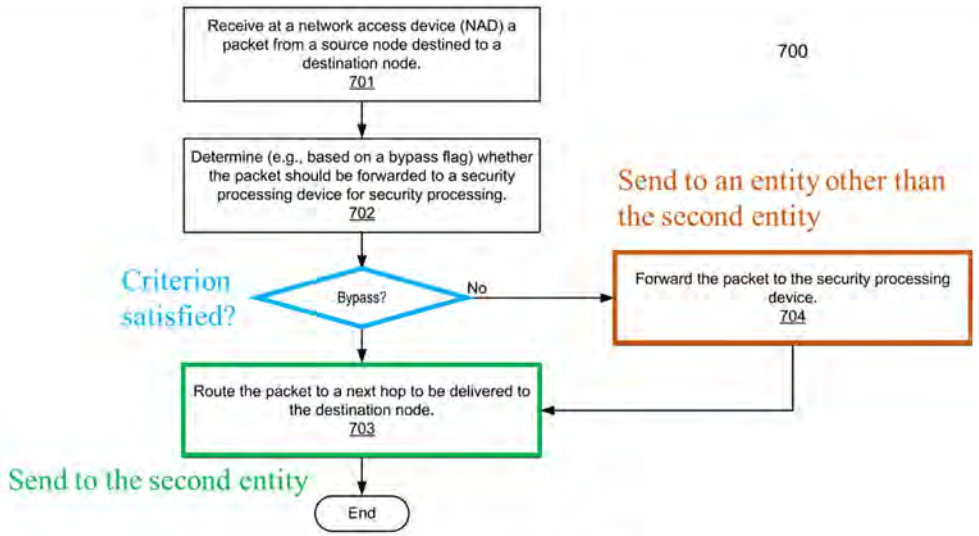


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 736 of 1100

| No. | '111 Patent Claim 1 | Shieh '088 |
|---|---|---|
| | | module to send out, or send the packet out directly to its destination if it's configured to do so.").<br><br>Shieh '088 ¶ [0041] ("In one embodiment, each of service processing modules 312-313 performs one or more of the functions of security processing module, such as, for example, NAT, VPN, DPI, and/or anti-virus, etc. However, it is different from the security processing module in that it only receives and sends packets to the same security processing module. If the tasks cannot be done in a security processing module, for example, due to a resource limitation, system load, or the requirement of a different operation system, the packets can be forwarded to one or more of service processing modules 312-313 for further processing. The packets then are sent back to the same security processing module for the next security function processing. To further share the system load, any of security processing modules 309-311 can load balance the computational-intensive services using multiple service processing modules.").<br><br>Shieh '088 Claim 1 ("A computer-implemented method, comprising:<br>receiving at a network access device a packet from a source node destined to a destination node;<br>examining a data structure maintained by the network access device to determine whether the data structure stores a data member having a predetermined value, the data member indicating whether the packet should undergo security processing;<br>if the data member matches the predetermined value, transmitting the packet to a security device associated with the network access device to allow the security device to perform content inspection, and<br>in response to a response received from the security device, routing the packet to the destination node dependent upon the response; and<br>transmitting the packet to the destination node without forwarding the packet to the security device, if the data member does not match the predetermined value."). |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 737 of 1100

| No. | '111 Patent Claim 1 | Shieh '088 |
|---|---|---|
| | |  <br>Fig. 7 (annotation added) |

| No. | '111 Patent Claim 2 | Shieh '088 |
|---|---|---|
| 2[a] | The method according to claim 1, wherein the instruction is 'probe', 'mirror', or 'terminate' instruction, and | Shieh '088 discloses wherein the instruction is 'probe', 'mirror', or 'terminate' instruction.<br><br>For example, Shieh '088 discloses an external controller that controls network nodes in a packet network, and identifies a command sent by the controller to the network access devices through a persistent connection to set up a set of filtering rules concerning whether and/or what types of packets should be forwarded to a security device. A person of ordinary skill would understand that such demands could include a probe, mirror, or terminate command. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Shieh '088 is found to not meet this limitation, wherein the instruction is 'probe', 'mirror', or 'terminate' instruction would have been obvious to a person having ordinary skill in the art, as explained below. |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 738 of 1100

| No. | '111 Patent Claim 2 | Shieh '088 |
|---|---|---|
| | | Shieh '088 ¶ [0017] ("According to some embodiments, a mechanism is utilized to dynamically perform security inspection in a network. In one embodiment, the mechanism includes two functions: 1) an input/output (IO) function that performs the distribution of network traffic; and 2) a security-processing function that performs security processing, including security inspection and policy enforcement. The IO function receives the packets and uses a session table to forward the packets to the security-processing function. A session table is a data structure that stores connection states, including the destination of the security-processing function. In one embodiment, the IO function determines, based on an internal data structure such as a session or flow table, whether the packet should be forwarded to the security processing function for security inspection. The configuration of the IO function to control whether to forward the packets to the security processing function can be set based on a command received from an administrator or alternatively, based on a signal received from the security processing function.").<br><br>Shieh '088 ¶ [0018] ("According to one embodiment, an administrator can configure, for example, via a controller or a management entity, a network access device to set up a set of filtering rules specifying whether and/or what types of packets should be forwarded to a security device and which of the security devices for security inspection. In this embodiment, the controller is configured to manage multiple network access devices and/or multiple security devices. Alternatively, a security device may inform a network access device that subsequent packets of a particular session should be forwarded from the network access device for security inspection. In one embodiment, a security device performs the security inspection at the beginning of the flow or session, and at a certain point, the security device decides that it no longer needs to inspect further packets of the same session.")<br><br>Shieh '088 ¶ [0023] ("According to one embodiment, a mechanism is utilized to dynamically perform security inspection in a network. In one embodiment, the mechanism includes two functions: 1) an input/output (IO) function (e.g., firewall module(s) 209) that performs the distribution of network traffic; and 2) a security-processing function (e.g., firewall module(s) 210) that performs security processing, including security inspection and policy enforcement. IO function 209 receives the packets and uses a session table to forward the packets to security-processing function 210. A session table is a data structure that stores connection states, including the destination of security-processing function. In one |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 739 of 1100

| No. | '111 Patent Claim 2 | Shieh '088 |
|---|---|---|
| | | embodiment, IO function 209 determines, based on an internal data structure such as a session or flow table (e.g., session table as shown in FIG. 5), whether the packet should be forwarded to security processing function 210 for security inspection. The configuration of IO function 209 to control whether to forward the packets to security processing function 210 can be set based on a command received from an administrator or alternatively, based on a signal received from security processing function 210.")<br><br>Shieh '088 ¶ [0025] ("According to one embodiment, each of network access devices 204A-204C maintains a persistent connection such as secure connections or tunnels 260 with a controller or management entity 208 for exchanging management messages and configurations, or distributing routing information to network access devices 204A-204C, etc. In one embodiment, controller 208 communicates with each of the network access devices 204A-204C using a management protocol such as the OpenFlow™ protocol. OpenFlow is a Layer 2 communications protocol (e.g., media access control or MAC layer) that gives access to the forwarding plane of a network switch or router over the network. In simpler terms, OpenFlow allows the path of network packets through the network of switches to be determined by software running on multiple routers (minimum two of them, primary and secondary, having a role of observers). This separation of the control from the forwarding allows for more sophisticated traffic management than is feasible using access control lists (ACLs) and routing protocols.").<br><br>Shieh '088 ¶ [0026] ("The OpenFlow technology consists of three parts: flow tables installed on switches, a controller, and an OpenFlow protocol for the controller to talk securely with switches. Flow tables are set up on switches or routers. Controllers talk to the switches via the OpenFlow Protocol, which is secure, and impose policies on flows. For example, a simple flow might be defined as any traffic from a given IP address. The rule governing it might be to route the flow through a given switch port. With its knowledge of the network, the controller could set up paths through the network optimized for speed, fewest number of hops or reduced latency, among other characteristics. Using OpenFlow takes control of how traffic flows through the network out of the hands of the infrastructure, the switches and routers, and puts it in the hands of the network owner (such as a corporation), individual users or individual applications.").<br><br> |

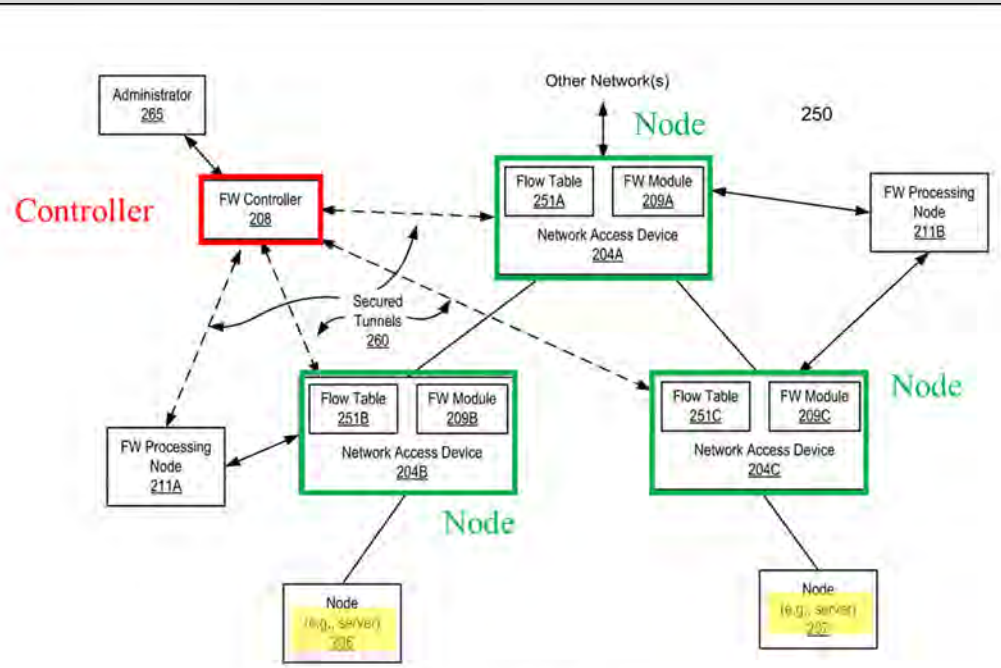


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 740 of 1100

| No. | '111 Patent Claim 2 | Shieh '088 |
|---|---|---|
| | | Shieh '088 ¶ [0028] ("Firewall modules 209A-209C may be part of a distributed firewall described above. For example, firewall modules 209A-209C may be the IO functions of a firewall while nodes 211A-211B may be firewall processing nodes. That is, modules 211A-211B may be dedicated firewall processing devices that perform some firewall processing operations such as DPI, content inspection, antivirus, etc., while firewall modules 209A-209C are responsible for routing data packets. For example, when firewall module 209B receives a packet from node 206, it may forward the packet to firewall processing node 211A for content inspection and/or forwards the packet to controller 208 for routing information. In response, firewall processing node 211A analyzes the received packet and/or further communicates with controller 208. Controller 208 may provide further routing information back to network access device 204B regarding how to route the packet. Each of the firewall processing nodes 211A-211B may further maintains a persistent connection or tunnel with controller 208, for example, using the OpenFlow communication protocol."). |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 741 of 1100

| No. | '111 Patent Claim 2 | Shieh '088 |
|---|---|---|
| | |  <br>Fig. 2A (annotation added)<br><br>Shieh '088 ¶ [0029] ("According to one embodiment, an administrator 265 configures, for example, via a controller or a management entity 208, a network access device (e.g., network access devices 204A-204C) to set up a set of filtering rules concerning whether and/or what types of packets should be forwarded to a security device and which of the security devices (e.g., security devices 211A-211B) for security inspection. In this embodiment, controller 208 is configured to manage multiple network access devices 204A-204C and/or multiple security devices 211A-211B. Alternatively, a security device, such as security device 211A, may inform a network access device, such as network access device 204B, whether subsequent packets of a particular session should be forwarded from the network access device for security inspection. A security device may perform the security inspection |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 742 of 1100

| No. | '111 Patent Claim 2 | Shieh '088 |
|---|---|---|
| | | on packets at the beginning of the flow or session, and at a certain point, the security device decides that it no longer needs to inspect further packets of the same session.").<br><br>Shieh '088 ¶ [0035] ("An embodiment of the invention also controls the communication between I/O functions and security-processing functions to enable packets to bypass security-processing function if there is no more need to inspect the packets of the connection. Some of the security functions do not need to inspect all the packets of a connection. For examples, to identify the application of a connection, there may be only need to inspect first four or five packets to make the identification. In this case, the security-processing function can notify I/O functions to bypass the security-processing function for the rest of the packets of the connections. Once the I/O function receives the notification, it will forward the packets out without redirecting the packets to the security-processing functions. This would greatly improve the performance even when security inspection is turned on.").<br><br>Shieh '088 ¶ [0036] ("During the bypass phase, the I/O function may notify the security-processing function if there are special events in the packet stream. These events could be receipt of TCP FIN or TCP RST packets, or not receiving any packets of the connection within a time threshold. The notification from I/O functions to security processing functions could help to clean up the state in the security-processing nodes.").<br><br>Shieh '088 ¶ [0049] ("In one embodiment, firewall modules 300A-300B could be distributed in different networks, even on different locations, as long as the modules can reach the module that is next in terms of processing and the central controller. In one embodiment, virtual I/O modules and corresponding security processing modules are in a public cloud and the central controller is in a private cloud. This configuration may provide the flexibility to secure and control packets coming from the public cloud, and allow central controller having overall view of traffic from Internet as well as from internal network.").<br><br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Shieh '088 in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 2[a] of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references. |


~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 743 of 1100

| No. | '111 Patent Claim 2 | Shieh '088 |
|---|---|---|
| | | For example, Chua discloses programming network nodes with redirecting, mirroring, and blocking programmed actions.<br><br>Chua at 7:28-54 ("SDN controller 112 may receive data as input from service devices 116. For example, SDN controller 112 may be con-figured to receive data from an intrusion detection system (IDS) device, a Denial of Service (DoS) device, a Distributed Denial of Service (DDoS) device, an intrusion prevention system (IPS) device, or the like. Based on this information, SDN controller 112 may make network enforcement decisions for specific traffic flows. That is, SDN controller 112 may program network devices of SDN 106 to perform pro-grammed actions on packets of a packet flow based on this data. Such programmed actions may include:<br><br>Allow-explicitly allow a certain network flow to proceed to its destination<br>Block-explicitly block a certain flow from traversing SDN 106<br>Mirror-allow the traffic, but send a copy of the traffic for deeper inspection or recording to, e.g., one of service devices 116<br>Redirect-redirect the traffic to another network (such as a honeypot device or other device of service devices 116) for either inspection or to keep a potential hacker 'busy' to determine if there is a real security threat.<br>Transform-modify or translate values of headers of packets in the network flow<br>Encapsulate-encapsulate packets in the network flow with a particular header")<br><br>Chua at 28:7-32 ("In addition, SDN controller 112 may configure the service device to send service-related data to one or more network devices (334). The service-related data may cause the net-work devices to change a path along which the packet is forwarded. For example, when the service device is a security device (e.g., a firewall or an IDS), if the security device determines that one or more packets of a packet flow are malicious, the security device may send service data indicat-ing that the packet flow includes malicious data. SDN con-troller 112 may program the network devices of the SDN to perform a programmed action based on the service-related data (336). For example, SDN controller 112 may program network devices to, in response to an indication that packets of a packet flow include malicious data, forward packets of the packet flow to a destination of the packet flow, forward packets of malicious packet flows to a collection device for further analysis, cause network devices to drop packets of the malicious packet flows, send a close session message to devices from |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 744 of 1100

| No. | '111 Patent Claim 2 | Shieh '088 |
|---|---|---|
| | | which packets of the malicious packet flows were received, block the packets of the packet flow, mirror copies of the packets of the packet flow to a second service device while forwarding the packets of the packet flow to the destination of the packet flow, redirect the packets of the packet flow to a third service device, transform one or more values of headers of the packets, and/or encapsulate the pack-ets with a particular header, or other such actions.")<br><br>As another example, Copeland discloses probing, copying, and terminating rules configured on the network device.<br><br>Copeland at [0057] ("In accordance with an aspect of the invention, a flow is considered terminated after a predetermined period of time has elapsed on a particular connection or port. For example, if HTTP Web traffic on port 80 ceases for a predetermined period of time, but other traffic begins to occur on port 80 after the expiration of that predetermined time period, it is considered that a new flow has begun, and the system responds accordingly to assign a new flow number and track the statistics and characteristics thereof. In the disclosed embodiment, the predetermined time period is 330 seconds, but those skilled in the art will understand that this time is arbitrary and may be heuristically adjusted.")<br><br>Copeland at [0082] ("Following the reserved field, the next 6 bits are a series of one-bit flags, shown in FIG. 2 as flags U, A, P, R, S, F. The first flag is the urgent flag (U). If the U flag is set, it indicates that the urgent pointer is valid and points to urgent data that should be acted upon as soon as possible. The next flag is the A ( or ACK or "acknowledgment") flag. The ACK flag indicates that an acknowledgment number is valid, and acknowledges that data has been received. The next flag, the push (P) flag, tells the receiving end to push all buffered data to the receiving application. The reset (R) flag is the following flag, which terminates both ends of the TCP connection. Next, the S (or SYN for "synchronize") flag is set in the initial packet of a TCP connection where both ends have to synchronize their TCP buffers. Following the SYN flag is the F (for FIN or "finish") flag. This flag signifies that the sending end of the communication and the host will not send any more data but still may acknowledge data that is received.")<br><br>Copeland at [0093] ("As illustrated, when Hostl terminates its end of the session, it sends a packet with the FIN and ACK flags set. The FIN flag informs Host2 that Hostl will send no |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 745 of 1100

| No. | '111 Patent Claim 2 | Shieh '088 |
|---|---|---|
| | | more data. The ACK flag acknowledges the last data received by Hostl by informing Host2 of the next sequence number it expects to receive.")<br><br>Copeland at [0095] ("When Host 2 is ready to terminate the session, it sends its own packet with the FIN and ACK flags set. Hostl responds that it has received the final packet with an ACK packet providing to Host2 an acknowledgment number one greater than the sequence number provided in the FIN-ACK packet of Host2.")<br><br>Copeland at [0099] ("As another example, if a particular host sends a large number of SYN packets to a target host and in response receives numerous R packets from the targeted host, a potential TCP probe is indicated. Likewise, numerous UDP packets sent from one host to a targeted host and numerous ICMP "port unavailable" packets received from the targeted host indicate a potential UDP probe. A stealth probe is indicated by multiple packets from the same source port number sent to different port numbers on a targeted host.")<br><br>Copeland at [0107] ("A flow is terminated if no communications occur between the two IP addresses and the one low port ( e.g. port 80) for 330 seconds. Most Web browsers or a TCP connec-tion send a reset packet (i.e. a packet with the R flag set) if no communications are sent or received for 5 minutes. An analysis can determine if the flow is abnormal or not for HTTP communications.")<br><br>Copeland at [0123] ("Flow processing is done for TCP and UDP packets, and the port numbers in the transport layer header are used to identify the flow record to be updated. For ICMP packets that constitute rejections of a packet, the copy of the rejected packet in the ICMP data field is used to identify the IP addresses and port numbers of the corresponding flow.")<br><br>Copeland at [0145] ("A list IP of addresses contacted or probed by each host can be maintained. When this list indicates that more than a threshold number of other hosts (e.g., 8) have been contacted in the same subnet, CI is added to the to the host and a bit in the host record is set to indicate that the host has received CI for "address scanning." Note that the number of hosts to designate a scan is not required to be a fixed value, but could be adjusted based on the sample rate or other means to enhance the accuracy making the number of hosts |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 746 of 1100

| No. | '111 Patent Claim 2 | Shieh '088 |
|---|---|---|
| | | scanned "statistically significant". These and other values of concern index are shown for non-flow based events in FIG. 7.")<br><br>Copeland at [0158] ("Flow processing is done for TCP and UDP packets, and the port numbers in the transport layer header are used to identify the flow record to be updated. For ICMP packets that constitute rejections of a packet, the copy of the rejected packet in the ICMP data field is used to identify the IP addresses and port numbers of the corresponding flow.") |
| 2[b] | upon receiving by the network node the 'terminate'' instruction, the method further comprising blocking, by the network node, the packet from being sent to the second entity and to the controller. | Shieh '088 discloses upon receiving by the network node the 'terminate' instruction, the method further comprising blocking, by the network node, the packet from being sent to the second entity and to the controller.<br><br>For example, Shieh '088 discloses an external controller that controls network nodes in a packet network, and identifies a command sent by the controller to the network access devices through a persistent connection to set up a set of filtering rules concerning whether and/or what types of packets should be forwarded to a security device. A person of ordinary skill would understand that such demands could include a terminate command. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Shieh '088 is found to not meet this limitation, upon receiving by the network node the 'terminate ' instruction, the method further comprising blocking, by the network node, the packet from being sent to the second entity and to the controller would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>Shieh '088 ¶ [0017] ("According to some embodiments, a mechanism is utilized to dynamically perform security inspection in a network. In one embodiment, the mechanism includes two functions: 1) an input/output (IO) function that performs the distribution of network traffic; and 2) a security-processing function that performs security processing, including security inspection and policy enforcement. The IO function receives the packets and uses a session table to forward the packets to the security-processing function. A session table is a data structure that stores connection states, including the destination of the security-processing function. In one embodiment, the IO function determines, based on an internal data structure such as a session or flow table, whether the packet should be forwarded to the security processing function for security inspection. The configuration of the IO function to |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 747 of 1100

| No. | '111 Patent Claim 2 | Shieh '088 |
|---|---|---|
| | | control whether to forward the packets to the security processing function can be set based on a command received from an administrator or alternatively, based on a signal received from the security processing function.").<br><br>Shieh '088 ¶ [0018] ("According to one embodiment, an administrator can configure, for example, via a controller or a management entity, a network access device to set up a set of filtering rules specifying whether and/or what types of packets should be forwarded to a security device and which of the security devices for security inspection. In this embodiment, the controller is configured to manage multiple network access devices and/or multiple security devices. Alternatively, a security device may inform a network access device that subsequent packets of a particular session should be forwarded from the network access device for security inspection. In one embodiment, a security device performs the security inspection at the beginning of the flow or session, and at a certain point, the security device decides that it no longer needs to inspect further packets of the same session.")<br><br>Shieh '088 ¶ [0023] ("According to one embodiment, a mechanism is utilized to dynamically perform security inspection in a network. In one embodiment, the mechanism includes two functions: 1) an input/output (IO) function (e.g., firewall module(s) 209) that performs the distribution of network traffic; and 2) a security-processing function (e.g., firewall module(s) 210) that performs security processing, including security inspection and policy enforcement. IO function 209 receives the packets and uses a session table to forward the packets to security-processing function 210. A session table is a data structure that stores connection states, including the destination of security-processing function. In one embodiment, IO function 209 determines, based on an internal data structure such as a session or flow table (e.g., session table as shown in FIG. 5), whether the packet should be forwarded to security processing function 210 for security inspection. The configuration of IO function 209 to control whether to forward the packets to security processing function 210 can be set based on a command received from an administrator or alternatively, based on a signal received from security processing function 210.")<br><br>Shieh '088 ¶ [0025] ("According to one embodiment, each of network access devices 204A-204C maintains a persistent connection such as secure connections or tunnels 260 with a controller or management entity 208 for exchanging management messages and configurations, or distributing routing information to network access devices 204A-204C, etc. |

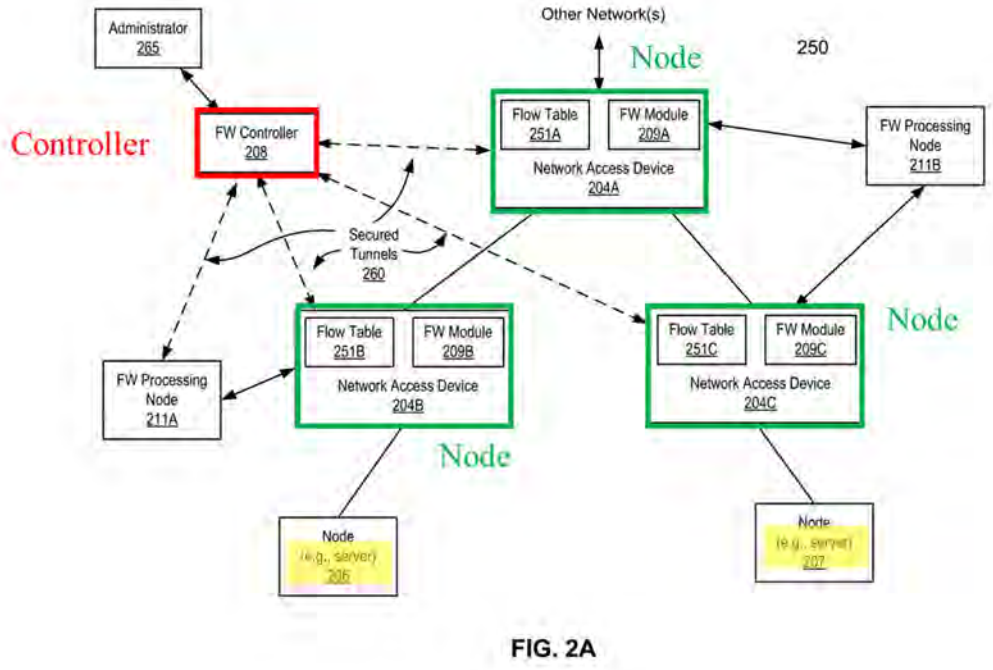


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 748 of 1100

| No. | '111 Patent Claim 2 | Shieh '088 |
|---|---|---|
| | | In one embodiment, controller 208 communicates with each of the network access devices 204A-204C using a management protocol such as the OpenFlow™ protocol. OpenFlow is a Layer 2 communications protocol (e.g., media access control or MAC layer) that gives access to the forwarding plane of a network switch or router over the network. In simpler terms, OpenFlow allows the path of network packets through the network of switches to be determined by software running on multiple routers (minimum two of them, primary and secondary, having a role of observers). This separation of the control from the forwarding allows for more sophisticated traffic management than is feasible using access control lists (ACLs) and routing protocols.").<br><br>Shieh '088 ¶ [0026] ("The OpenFlow technology consists of three parts: flow tables installed on switches, a controller, and an OpenFlow protocol for the controller to talk securely with switches. Flow tables are set up on switches or routers. Controllers talk to the switches via the OpenFlow Protocol, which is secure, and impose policies on flows. For example, a simple flow might be defined as any traffic from a given IP address. The rule governing it might be to route the flow through a given switch port. With its knowledge of the network, the controller could set up paths through the network optimized for speed, fewest number of hops or reduced latency, among other characteristics. Using OpenFlow takes control of how traffic flows through the network out of the hands of the infrastructure, the switches and routers, and puts it in the hands of the network owner (such as a corporation), individual users or individual applications.").<br><br>Shieh '088 ¶ [0028] ("Firewall modules 209A-209C may be part of a distributed firewall described above. For example, firewall modules 209A-209C may be the IO functions of a firewall while nodes 211A-211B may be firewall processing nodes. That is, modules 211A-211B may be dedicated firewall processing devices that perform some firewall processing operations such as DPI, content inspection, antivirus, etc., while firewall modules 209A-209C are responsible for routing data packets. For example, when firewall module 209B receives a packet from node 206, it may forward the packet to firewall processing node 211A for content inspection and/or forwards the packet to controller 208 for routing information. In response, firewall processing node 211A analyzes the received packet and/or further communicates with controller 208. Controller 208 may provide further routing information back to network access device 204B regarding how to route the packet. Each of the firewall processing |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 749 of 1100

| No. | '111 Patent Claim 2 | Shieh '088 |
|---|---|---|
| | | nodes 211A-211B may further maintains a persistent connection or tunnel with controller 208, for example, using the OpenFlow communication protocol.").<br>  <br><br>FIG. 2A<br>Fig. 2A (annotation added)<br>Shieh '088 ¶ [0029] ("According to one embodiment, an administrator 265 configures, for example, via a controller or a management entity 208, a network access device (e.g., network access devices 204A-204C) to set up a set of filtering rules concerning whether and/or what types of packets should be forwarded to a security device and which of the security devices (e.g., security devices 211A-211B) for security inspection. In this embodiment, controller 208 is configured to manage multiple network access devices 204A-204C and/or multiple security devices 211A-211B. Alternatively, a security device, such as security device 211A, may inform a network access device, such as network access device 204B, |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 750 of 1100

| No. | '111 Patent Claim 2 | Shieh '088 |
|---|---|---|
| | | whether subsequent packets of a particular session should be forwarded from the network access device for security inspection. A security device may perform the security inspection on packets at the beginning of the flow or session, and at a certain point, the security device decides that it no longer needs to inspect further packets of the same session.").<br><br>Shieh '088 ¶ [0035] ("An embodiment of the invention also controls the communication between I/O functions and security-processing functions to enable packets to bypass security-processing function if there is no more need to inspect the packets of the connection. Some of the security functions do not need to inspect all the packets of a connection. For examples, to identify the application of a connection, there may be only need to inspect first four or five packets to make the identification. In this case, the security-processing function can notify I/O functions to bypass the security-processing function for the rest of the packets of the connections. Once the I/O function receives the notification, it will forward the packets out without redirecting the packets to the security-processing functions. This would greatly improve the performance even when security inspection is turned on.").<br><br>Shieh '088 ¶ [0036] ("During the bypass phase, the I/O function may notify the security-processing function if there are special events in the packet stream. These events could be receipt of TCP FIN or TCP RST packets, or not receiving any packets of the connection within a time threshold. The notification from I/O functions to security processing functions could help to clean up the state in the security-processing nodes.").<br><br>Shieh '088 ¶ [0049] ("In one embodiment, firewall modules 300A-300B could be distributed in different networks, even on different locations, as long as the modules can reach the module that is next in terms of processing and the central controller. In one embodiment, virtual I/O modules and corresponding security processing modules are in a public cloud and the central controller is in a private cloud. This configuration may provide the flexibility to secure and control packets coming from the public cloud, and allow central controller having overall view of traffic from Internet as well as from internal network.").<br><br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Shieh '088 in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 751 of 1100

| No. | '111 Patent Claim 2 | Shieh '088 |
|---|---|---|
| | | references identified in element 2[b] of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references.<br><br>For example, Chua discloses programming network nodes with blocking programmed actions.<br><br>Chua at 7:28-54 ("SDN controller 112 may receive data as input from service devices 116. For example, SDN controller 112 may be con-figured to receive data from an intrusion detection system (IDS) device, a Denial of Service (DoS) device, a Distributed Denial of Service (DDoS) device, an intrusion prevention system (IPS) device, or the like. Based on this information, SDN controller 112 may make network enforcement decisions for specific traffic flows. That is, SDN controller 112 may program network devices of SDN 106 to perform pro-grammed actions on packets of a packet flow based on this data. Such programmed actions may include:<br><br>Allow-explicitly allow a certain network flow to proceed to its destination<br>Block-explicitly block a certain flow from traversing SDN 106<br>Mirror-allow the traffic, but send a copy of the traffic for deeper inspection or recording to, e.g., one of service devices 116<br>Redirect-redirect the traffic to another network (such as a honeypot device or other device of service devices 116) for either inspection or to keep a potential hacker 'busy' to determine if there is a real security threat.<br>Transform-modify or translate values of headers of packets in the network flow<br>Encapsulate-encapsulate packets in the network flow with a particular header")<br><br>Chua at 28:7-32 ("In addition, SDN controller 112 may configure the service device to send service-related data to one or more network devices (334). The service-related data may cause the net-work devices to change a path along which the packet is forwarded. For example, when the service device is a security device (e.g., a firewall or an IDS), if the security device determines that one or more packets of a packet flow are malicious, the security device may send service data indicat-ing that the packet flow includes malicious data. SDN con-troller 112 may program the network devices of the SDN to perform a programmed action based on the service-related data (336). For example, SDN controller 112 may program network devices to, in response to an indication that packets of a packet flow include malicious data, |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 752 of 1100

| No. | '111 Patent Claim 2 | Shieh '088 |
|---|---|---|
| | | forward packets of the packet flow to a destination of the packet flow, forward packets of malicious packet flows to a collection device for further analysis, cause network devices to drop packets of the malicious packet flows, send a close session message to devices from which packets of the malicious packet flows were received, block the packets of the packet flow, mirror copies of the packets of the packet flow to a second service device while forwarding the packets of the packet flow to the destination of the packet flow, redirect the packets of the packet flow to a third service device, transform one or more values of headers of the packets, and/or encapsulate the pack-ets with a particular header, or other such actions.")<br><br>As another example, Copeland discloses terminating rules configured on the network device.<br><br>Copeland at [0057] ("In accordance with an aspect of the invention, a flow is considered terminated after a predetermined period of time has elapsed on a particular connection or port. For example, if HTTP Web traffic on port 80 ceases for a predetermined period of time, but other traffic begins to occur on port 80 after the expiration of that predetermined time period, it is considered that a new flow has begun, and the system responds accordingly to assign a new flow number and track the statistics and characteristics thereof. In the disclosed embodiment, the predetermined time period is 330 seconds, but those skilled in the art will understand that this time is arbitrary and may be heuristically adjusted.")<br><br>Copeland at [0082] ("Following the reserved field, the next 6 bits are a series of one-bit flags, shown in FIG. 2 as flags U, A, P, R, S, F. The first flag is the urgent flag (U). If the U flag is set, it indicates that the urgent pointer is valid and points to urgent data that should be acted upon as soon as possible. The next flag is the A ( or ACK or "acknowledgment") flag. The ACK flag indicates that an acknowledgment number is valid, and acknowledges that data has been received. The next flag, the push (P) flag, tells the receiving end to push all buffered data to the receiving application. The reset (R) flag is the following flag, which terminates both ends of the TCP connection. Next, the S (or SYN for "synchronize") flag is set in the initial packet of a TCP connection where both ends have to synchronize their TCP buffers. Following the SYN flag is the F (for FIN or "finish") flag. This flag signifies that the sending end of the communication and the host will not send any more data but still may acknowledge data that is received.") |

| No. | '111 Patent Claim 2 | Shieh '088 |
|---|---|---|
| | | Copeland at [0093] ("As illustrated, when Hostl terminates its end of the session, it sends a packet with the FIN and ACK flags set. The FIN flag informs Host2 that Hostl will send no more data. The ACK flag acknowledges the last data received by Hostl by informing Host2 of the next sequence number it expects to receive.")<br><br>Copeland at [0095] ("When Host 2 is ready to terminate the session, it sends its own packet with the FIN and ACK flags set. Hostl responds that it has received the final packet with an ACK packet providing to Host2 an acknowledgment number one greater than the sequence number provided in the FIN-ACK packet of Host2.")<br><br>Copeland at [0099] ("As another example, if a particular host sends a large number of SYN packets to a target host and in response receives numerous R packets from the targeted host, a potential TCP probe is indicated. Likewise, numerous UDP packets sent from one host to a targeted host and numerous ICMP "port unavailable" packets received from the targeted host indicate a potential UDP probe. A stealth probe is indicated by multiple packets from the same source port number sent to different port numbers on a targeted host.")<br><br>Copeland at [0107] ("A flow is terminated if no communications occur between the two IP addresses and the one low port ( e.g. port 80) for 330 seconds. Most Web browsers or a TCP connec-tion send a reset packet (i.e. a packet with the R flag set) if no communications are sent or received for 5 minutes. An analysis can determine if the flow is abnormal or not for HTTP communications.")<br><br>Copeland at [0123] ("Flow processing is done for TCP and UDP packets, and the port numbers in the transport layer header are used to identify the flow record to be updated. For ICMP packets that constitute rejections of a packet, the copy of the rejected packet in the ICMP data field is used to identify the IP addresses and port numbers of the corresponding flow.")<br><br>Copeland at [0145] ("A list IP of addresses contacted or probed by each host can be maintained. When this list indicates that more than a threshold number of other hosts (e.g., 8) have been contacted in the same subnet, CI is added to the to the host and a bit in the host record is set to indicate that the host has received CI for "address scanning." Note that the number of hosts to designate a scan is not required to be a fixed value, but could be adjusted |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 754 of 1100

| No. | '111 Patent Claim 2 | Shieh '088 |
|---|---|---|
| | | based on the sample rate or other means to enhance the accuracy making the number of hosts scanned "statistically significant". These and other values of concern index are shown for non-flow based events in FIG. 7.")<br><br>Copeland at [0158] ("Flow processing is done for TCP and UDP packets, and the port numbers in the transport layer header are used to identify the flow record to be updated. For ICMP packets that constitute rejections of a packet, the copy of the rejected packet in the ICMP data field is used to identify the IP addresses and port numbers of the corresponding flow.") |

| No. | '111 Patent Claim 3 | Shieh '088 |
|---|---|---|
| 3[a] | The method according to claim 1, wherein the instruction is a 'probe', a 'mirror', or a 'terminate' instruction, and | Shieh '088 discloses wherein the instruction is a 'probe', a 'mirror', or a 'terminate' instruction.<br><br>*See supra at* 1[a]. |
| 3[b] | upon receiving by the network node the 'mirror' instruction and responsive to the packet satisfying the criterion, the method further comprising sending the packet, by the network node, to the second entity and to the controller. | Shieh '088 discloses upon receiving by the network node the 'mirror' instruction and responsive to the packet satisfying the criterion, the method further comprising sending the packet, by the network node, to the second entity and to the controller.<br><br>For example, Shieh '088 discloses an external controller that controls network nodes in a packet network, and identifies a command sent by the controller to the network access devices through a persistent connection to set up a set of filtering rules concerning whether and/or what types of packets should be forwarded to a security device. A person of ordinary skill would understand that such demands could include a mirror command. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Shieh '088 is found to not meet this limitation, upon receiving by the network node the 'mirror' instruction and responsive to the packet satisfying the criterion, method further comprising sending the packet, by the network node, to the second entity and to the controller would have been obvious to a person having ordinary skill in the art, as explained below. |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 755 of 1100

| No. | '111 Patent Claim 3 | Shieh '088 |
|---|---|---|
| | | Shieh '088 ¶ [0017] ("According to some embodiments, a mechanism is utilized to dynamically perform security inspection in a network. In one embodiment, the mechanism includes two functions: 1) an input/output (IO) function that performs the distribution of network traffic; and 2) a security-processing function that performs security processing, including security inspection and policy enforcement. The IO function receives the packets and uses a session table to forward the packets to the security-processing function. A session table is a data structure that stores connection states, including the destination of the security-processing function. In one embodiment, the IO function determines, based on an internal data structure such as a session or flow table, whether the packet should be forwarded to the security processing function for security inspection. The configuration of the IO function to control whether to forward the packets to the security processing function can be set based on a command received from an administrator or alternatively, based on a signal received from the security processing function.").<br><br>Shieh '088 ¶ [0018] ("According to one embodiment, an administrator can configure, for example, via a controller or a management entity, a network access device to set up a set of filtering rules specifying whether and/or what types of packets should be forwarded to a security device and which of the security devices for security inspection. In this embodiment, the controller is configured to manage multiple network access devices and/or multiple security devices. Alternatively, a security device may inform a network access device that subsequent packets of a particular session should be forwarded from the network access device for security inspection. In one embodiment, a security device performs the security inspection at the beginning of the flow or session, and at a certain point, the security device decides that it no longer needs to inspect further packets of the same session.")<br><br>Shieh '088 ¶ [0023] ("According to one embodiment, a mechanism is utilized to dynamically perform security inspection in a network. In one embodiment, the mechanism includes two functions: 1) an input/output (IO) function (e.g., firewall module(s) 209) that performs the distribution of network traffic; and 2) a security-processing function (e.g., firewall module(s) 210) that performs security processing, including security inspection and policy enforcement. IO function 209 receives the packets and uses a session table to forward the packets to security-processing function 210. A session table is a data structure that stores connection states, including the destination of security-processing function. In one embodiment, IO function 209 determines, based on an internal data structure such as a session |

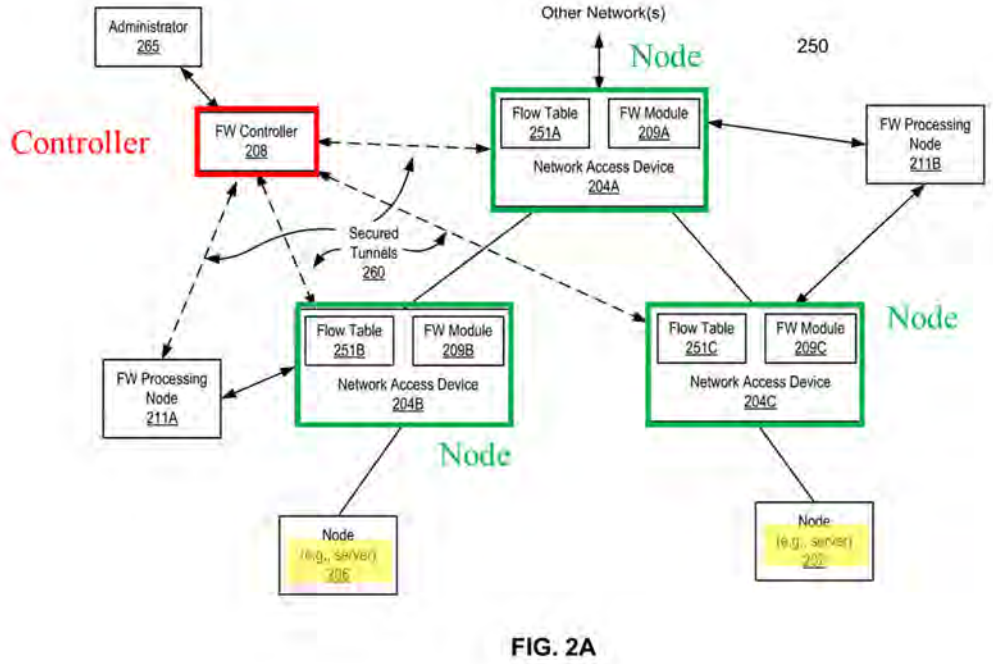


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 756 of 1100

| No. | '111 Patent Claim 3 | Shieh '088 |
|---|---|---|
| | | or flow table (e.g., session table as shown in FIG. 5), whether the packet should be forwarded to security processing function 210 for security inspection. The configuration of IO function 209 to control whether to forward the packets to security processing function 210 can be set based on a command received from an administrator or alternatively, based on a signal received from security processing function 210.")<br><br>Shieh '088 ¶ [0025] ("According to one embodiment, each of network access devices 204A-204C maintains a persistent connection such as secure connections or tunnels 260 with a controller or management entity 208 for exchanging management messages and configurations, or distributing routing information to network access devices 204A-204C, etc. In one embodiment, controller 208 communicates with each of the network access devices 204A-204C using a management protocol such as the OpenFlow™ protocol. OpenFlow is a Layer 2 communications protocol (e.g., media access control or MAC layer) that gives access to the forwarding plane of a network switch or router over the network. In simpler terms, OpenFlow allows the path of network packets through the network of switches to be determined by software running on multiple routers (minimum two of them, primary and secondary, having a role of observers). This separation of the control from the forwarding allows for more sophisticated traffic management than is feasible using access control lists (ACLs) and routing protocols.").<br><br>Shieh '088 ¶ [0026] ("The OpenFlow technology consists of three parts: flow tables installed on switches, a controller, and an OpenFlow protocol for the controller to talk securely with switches. Flow tables are set up on switches or routers. Controllers talk to the switches via the OpenFlow Protocol, which is secure, and impose policies on flows. For example, a simple flow might be defined as any traffic from a given IP address. The rule governing it might be to route the flow through a given switch port. With its knowledge of the network, the controller could set up paths through the network optimized for speed, fewest number of hops or reduced latency, among other characteristics. Using OpenFlow takes control of how traffic flows through the network out of the hands of the infrastructure, the switches and routers, and puts it in the hands of the network owner (such as a corporation), individual users or individual applications.").<br><br>Shieh '088 ¶ [0028] ("Firewall modules 209A-209C may be part of a distributed firewall described above. For example, firewall modules 209A-209C may be the IO functions of a |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 757 of 1100

| No. | '111 Patent Claim 3 | Shieh '088 |
|---|---|---|
| | | firewall while nodes 211A-211B may be firewall processing nodes. That is, modules 211A-211B may be dedicated firewall processing devices that perform some firewall processing operations such as DPI, content inspection, antivirus, etc., while firewall modules 209A-209C are responsible for routing data packets. For example, when firewall module 209B receives a packet from node 206, it may forward the packet to firewall processing node 211A for content inspection and/or forwards the packet to controller 208 for routing information. In response, firewall processing node 211A analyzes the received packet and/or further communicates with controller 208. Controller 208 may provide further routing information back to network access device 204B regarding how to route the packet. Each of the firewall processing nodes 211A-211B may further maintains a persistent connection or tunnel with controller 208, for example, using the OpenFlow communication protocol.").<br>  <br><br>Fig. 2A (annotation added) |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 758 of 1100

| No. | '111 Patent Claim 3 | Shieh '088 |
|---|---|---|
| | | Shieh '088 ¶ [0029] ("According to one embodiment, an administrator 265 configures, for example, via a controller or a management entity 208, a network access device (e.g., network access devices 204A-204C) to set up a set of filtering rules concerning whether and/or what types of packets should be forwarded to a security device and which of the security devices (e.g., security devices 211A-211B) for security inspection. In this embodiment, controller 208 is configured to manage multiple network access devices 204A-204C and/or multiple security devices 211A-211B. Alternatively, a security device, such as security device 211A, may inform a network access device, such as network access device 204B, whether subsequent packets of a particular session should be forwarded from the network access device for security inspection. A security device may perform the security inspection on packets at the beginning of the flow or session, and at a certain point, the security device decides that it no longer needs to inspect further packets of the same session.").<br><br>Shieh '088 ¶ [0035] ("An embodiment of the invention also controls the communication between I/O functions and security-processing functions to enable packets to bypass security-processing function if there is no more need to inspect the packets of the connection. Some of the security functions do not need to inspect all the packets of a connection. For examples, to identify the application of a connection, there may be only need to inspect first four or five packets to make the identification. In this case, the security-processing function can notify I/O functions to bypass the security-processing function for the rest of the packets of the connections. Once the I/O function receives the notification, it will forward the packets out without redirecting the packets to the security-processing functions. This would greatly improve the performance even when security inspection is turned on.").<br><br>Shieh '088 ¶ [0036] ("During the bypass phase, the I/O function may notify the security-processing function if there are special events in the packet stream. These events could be receipt of TCP FIN or TCP RST packets, or not receiving any packets of the connection within a time threshold. The notification from I/O functions to security processing functions could help to clean up the state in the security-processing nodes.").<br><br>Shieh '088 ¶ [0049] ("In one embodiment, firewall modules 300A-300B could be distributed in different networks, even on different locations, as long as the modules can reach the module that is next in terms of processing and the central controller. In one embodiment, virtual I/O |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 759 of 1100

| No. | '111 Patent Claim 3 | Shieh '088 |
|---|---|---|
| | | modules and corresponding security processing modules are in a public cloud and the central controller is in a private cloud. This configuration may provide the flexibility to secure and control packets coming from the public cloud, and allow central controller having overall view of traffic from Internet as well as from internal network.").<br><br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Shieh '088 in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 3[b] of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references.<br><br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Shieh '088 in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 3(b) of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references.<br><br>For example, Chua discloses a mirror program in response to an indication based on the packet header in which the network devices mirror copies of the packets of the packet flow to a second service device while forwarding the packets of the packet flow to the destination of the packet flow.<br><br>Chua at 7:28-54 ("SDN controller 112 may receive data as input from service devices 116. For example, SDN controller 112 may be con-figured to receive data from an intrusion detection system (IDS) device, a Denial of Service (DoS) device, a Distributed Denial of Service (DDoS) device, an intrusion prevention system (IPS) device, or the like. Based on this information, SDN controller 112 may make network enforcement decisions for specific traffic flows. That is, SDN controller 112 may program network devices of SDN 106 to perform pro-grammed actions on packets of a packet flow based on this data. Such programmed actions may include:<br><br>Allow-explicitly allow a certain network flow to proceed to its destination<br>Block-explicitly block a certain flow from traversing SDN 106 |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 760 of 1100

| No. | '111 Patent Claim 3 | Shieh '088 |
|---|---|---|
| | | Mirror-allow the traffic, but send a copy of the traffic for deeper inspection or recording to, e.g., one of service devices 116<br>Redirect-redirect the traffic to another network (such as a honeypot device or other device of service devices 116) for either inspection or to keep a potential hacker 'busy' to determine if there is a real security threat.<br>Transform-modify or translate values of headers of packets in the network flow<br>Encapsulate-encapsulate packets in the network flow with a particular header")<br><br>Chua at 16:23-44 ("More particularly, control unit 130 may configure any of service devices 116 to send data representative of a particular event to SDN controller 112, and control unit 130 may auto-matically reprogram one or more network devices of SDN 106 in response to such data. For example, security monitor-ing applications of service devices 116 may determine that a specific source port, destination port, source IP address, des-tination IP address, or the like should be acted upon. Alter-natively, security monitoring applications may determine that, due to content or deep packet inspection, a specific type of traffic is malicious and should be blocked. In either case, the corresponding one of service devices 116 may send a message to SDN controller 112 representative of these deter-minations. As yet another example, a network performance device may monitor various performance metrics, such as latency, jitter, packet loss, or the like, and provide feedback data to SDN controller 112 based on these metrics. SDN controller 112 may respond by programming network devices of SDN 106 to perform a programmed action, such as allowing corresponding traffic, blocking corresponding traf-fic, mirroring corresponding traffic, redirecting correspond-ing traffic.")<br><br>Chua at 28:7-32 ("In addition, SDN controller 112 may configure the service device to send service-related data to one or more network devices (334). The service-related data may cause the net-work devices to change a path along which the packet is forwarded. For example, when the service device is a security device (e.g., a firewall or an IDS), if the security device determines that one or more packets of a packet flow are malicious, the security device may send service data indicat-ing that the packet flow includes malicious data. SDN con-troller 112 may program the network devices of the SDN to perform a programmed action based on the service-related data (336). For example, SDN controller 112 may program network devices to, in response to an indication that packets of a packet flow include malicious data, forward packets of the packet flow to a destination of the packet flow, forward packets of |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 761 of 1100

| No. | ’111 Patent Claim 3 | Shieh ’088 |
|---|---|---|
| | | malicious packet flows to a collection device for further analysis, cause network devices to drop packets of the malicious packet flows, send a close session message to devices from which packets of the malicious packet flows were received, block the packets of the packet flow, mirror copies of the packets of the packet flow to a second service device while forwarding the packets of the packet flow to the destination of the packet flow, redirect the packets of the packet flow to a third service device, transform one or more values of headers of the packets, and/or encapsulate the pack-ets with a particular header, or other such actions.”)<br><br>As another example, Swenson discloses a counting mode instructed by the network controller to the steering device for monitoring and optimizing, in which the steering device forwards the packet flow to the user device/origin server and at the same time, sending the packet flow to the network controller.<br><br>Swenson at [0026] (“The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.”)<br><br>Swenson at [0028] (“In contrast to conventional inline TCP throughput monitoring devices that monitor every single data packets transmitted and received, the network controller 140 is an "out-of-band" computer server that interfaces with the steer-ing device 130 to selectively inspect user flows of interest. The network controller 140 may further identify user flows (e.g., among the flows of interest) for optimization. In one embodiment, the network controller 140 may be imple-mented at the steering device 130 to monitor traffic. In other embodiments, the network controller 140 is coupled to and communicates with the steering device 130 for traffic moni-toring and optimization. When queried by the steering device 130, the network controller 140 determines if a given network flow should be ignored, monitored |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 762 of 1100

| No. | '111 Patent Claim 3 | Shieh '088 |
|---|---|---|
| | | further or optimized. Opti-mization of a flow is often decided at the beginning of the flow because it is rarely possible to switch to optimized content mid-stream once non-optimized content delivery has begun. However, the network controller 140 may determine that existing flows associated with a particular subscriber or other entity should be optimized. In turn, new flows ( e.g., resulting from seek requests in media, new media requests, resume after pause, etc.) determined to be associated with the entity may be optimized. The network controller 140 uses the net-work state as well as historical traffic data in its decision for monitoring and optimization. Knowledge on the current net-work state, such as congestion, deems critical when it comes to data optimization.")<br><br>Swenson at [0059] ("In one embodiment, as the steering device 130 monitors network responses, it is looking for flows that match one or more signatures for video and images. When a match-ing flow is detected, the steering device 130 forwards the HTTP request and a portion of the HTTP response to the network controller 140 over the ICAP client interface 404. After receiving the request and the portion of response at the ICAP server interface 406, the flow analyzer 312 of the net-work controller 140 performs a deep flow inspection to deter-mine if the flow is worth bandwidth monitoring and/or user detection. For example, the flow inspection performed by the flow analyzer 312 may determine if the flow indeed contains large or medium object ( e.g., larger than 50 kB), and/or if the source IP address of the flow is from a user or a group of users that are required to be monitored by policies. The flow ana-lyzer 312 may also determine if the flow needs to be opti-mized based on historical flow statistical data.")<br><br>Swenson at [0064] ("Similar to the "continue" mode, after receiving the initial HTTP messages of a flow and determining to monitor the flow, the network controller 140 notify the steering device 130 to work in a "counting" mode for bandwidth monitoring. In contrast to the "continue" mode, when a matching flow is detected for "counting" mode, the steering device 130 for-wards the HTTP response directly to the user device 110. While at the same time, the steering device 130 send a cus-tomized ICAP message to the network controller 140 over the network link 425. In one embodiment, the customized ICAP message contains the HTTP request and response headers, as well as a count of payload size of the current flow. After updating the flow statistics, the network controller 140 may acknowledge the gateway over the network line 426. In the "counting" mode, the network controller 140 does not join the network response path as an inline network element, but simply listens to the counting of |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 763 of 1100

| No. | '111 Patent Claim 3 | Shieh '088 |
|---|---|---|
| | | flow size. The benefit of the "counting" mode is to off-load the network controller 140 from ingesting and forwarding the network flow on the net-work response path, while still enabling the detection of con-gestions and estimation of bandwidth associated with the flows of interest.")<br><br>Swenson at [0071] ("After receiving the request, the video optimizer 150 forwards the video HTTP GET requests 622 to the origin server 160 and in return, receives a video file 624 from the origin server 160. The video optimizer 150 transcodes the video file to a format usable by the client device 110 based on network bandwidth available to the user device 110. The optimized video 626 is then transmitted from the video opti-mizer 150 to the steering device 130. In one embodiment, the steering device 130 intercepts the optimized video 626. The steering device 130 will then send an ICAP request to the network controller 140 for inspection. The network controller 140 deems this flow to be monitored and sends ICAP response 630. The steering device 130 then allows the flow to go through to the user device 110. The steering device 130 next sends periodic ICAP "counting" updates 632 to the network controller 140 until the flow completes. As such, the client receives the optimized video 626 for substantially real-time playback on an application executing on the user device 110.")<br>Swenson at [0072] ("In one embodiment, if the video optimizer 150 failed to retrieve user requested video file from the origin server 160, the video optimizer 150 appends a "do not transcode" flag to the HTTP redirect request and returned to the user device 110, which re-sends the request out over the network to the origin server 160. The origin server 160 responds appropriately to the request by sending back video 624, which is intercepted by the steering device 130 only. The steering device 130 forwards the video to the user device 110 and at the same time reports the flow size to the network controller 140 for monitoring purpose.") |

| No. | '111 Patent Claim 4 | Shieh '088 |
|---|---|---|
| 4[a] | The method according to claim 1, wherein the instruction is 'probe', 'mirror', or 'terminate' instruction, and | Shieh '088 discloses wherein the instruction is 'probe', 'mirror', or 'terminate' instruction.<br><br>*See supra at* 1[a]. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 764 of 1100

| No. | '111 Patent Claim 4 | Shieh '088 |
|---|---|---|
| 4[b] | upon receiving by the network node the 'probe' instruction and responsive to the packet satisfying the criterion, the method further comprising: sending the packet, by the network node, to the controller; | Shieh '088 discloses upon receiving by the network node the 'probe' instruction and responsive to the packet satisfying the criterion, the method further comprising: sending the packet, by the network node, to the controller.<br><br>For example, Shieh '088 discloses an external controller that controls network nodes in a packet network, and identifies a command sent by the controller to the network access devices through a persistent connection to set up a set of filtering rules concerning whether and/or what types of packets should be forwarded to a security device. A person of ordinary skill would understand that such demands are probe commands. Shieh '088 further discloses that when the central controller receives the packet, it decides which of the security processing modules is able to process the packets, and then forwards the packets to the designated security processing module. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Shieh '088 is found to not meet this limitation, upon receiving by the network node the 'probe' instruction and responsive to the packet satisfying the criterion, the method further comprising: sending the packet, by the network node, to the controller would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>Shieh '088 ¶ [0017] ("According to some embodiments, a mechanism is utilized to dynamically perform security inspection in a network. In one embodiment, the mechanism includes two functions: 1) an input/output (IO) function that performs the distribution of network traffic; and 2) a security-processing function that performs security processing, including security inspection and policy enforcement. The IO function receives the packets and uses a session table to forward the packets to the security-processing function. A session table is a data structure that stores connection states, including the destination of the security-processing function. In one embodiment, the IO function determines, based on an internal data structure such as a session or flow table, whether the packet should be forwarded to the security processing function for security inspection. The configuration of the IO function to control whether to forward the packets to the security processing function can be set based on a command received from an administrator or alternatively, based on a signal received from the security processing function.").<br><br>Shieh '088 ¶ [0018] ("According to one embodiment, an administrator can configure, for example, via a controller or a management entity, a network access device to set up a set of |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 765 of 1100

| No. | '111 Patent Claim 4 | Shieh '088 |
|---|---|---|
| | | filtering rules specifying whether and/or what types of packets should be forwarded to a security device and which of the security devices for security inspection. In this embodiment, the controller is configured to manage multiple network access devices and/or multiple security devices. Alternatively, a security device may inform a network access device that subsequent packets of a particular session should be forwarded from the network access device for security inspection. In one embodiment, a security device performs the security inspection at the beginning of the flow or session, and at a certain point, the security device decides that it no longer needs to inspect further packets of the same session.")<br><br>Shieh '088 ¶ [0023] ("According to one embodiment, a mechanism is utilized to dynamically perform security inspection in a network. In one embodiment, the mechanism includes two functions: 1) an input/output (IO) function (e.g., firewall module(s) 209) that performs the distribution of network traffic; and 2) a security-processing function (e.g., firewall module(s) 210) that performs security processing, including security inspection and policy enforcement. IO function 209 receives the packets and uses a session table to forward the packets to security-processing function 210. A session table is a data structure that stores connection states, including the destination of security-processing function. In one embodiment, IO function 209 determines, based on an internal data structure such as a session or flow table (e.g., session table as shown in FIG. 5), whether the packet should be forwarded to security processing function 210 for security inspection. The configuration of IO function 209 to control whether to forward the packets to security processing function 210 can be set based on a command received from an administrator or alternatively, based on a signal received from security processing function 210.")<br><br>Shieh '088 ¶ [0025] ("According to one embodiment, each of network access devices 204A-204C maintains a persistent connection such as secure connections or tunnels 260 with a controller or management entity 208 for exchanging management messages and configurations, or distributing routing information to network access devices 204A-204C, etc. In one embodiment, controller 208 communicates with each of the network access devices 204A-204C using a management protocol such as the OpenFlow™ protocol. OpenFlow is a Layer 2 communications protocol (e.g., media access control or MAC layer) that gives access to the forwarding plane of a network switch or router over the network. In simpler terms, OpenFlow allows the path of network packets through the network of switches to be determined by software running on multiple routers (minimum two of them, primary |

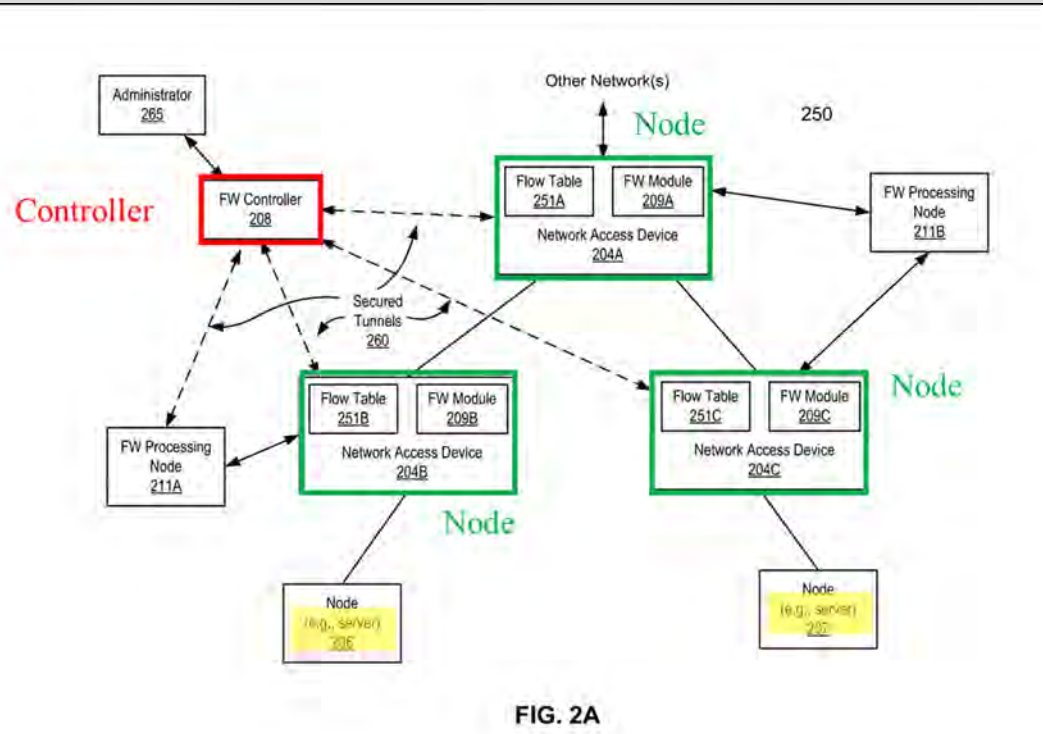


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 766 of 1100

| No. | '111 Patent Claim 4 | Shieh '088 |
|---|---|---|
| | | and secondary, having a role of observers). This separation of the control from the forwarding allows for more sophisticated traffic management than is feasible using access control lists (ACLs) and routing protocols.").<br><br>Shieh '088 ¶ [0026] ("The OpenFlow technology consists of three parts: flow tables installed on switches, a controller, and an OpenFlow protocol for the controller to talk securely with switches. Flow tables are set up on switches or routers. Controllers talk to the switches via the OpenFlow Protocol, which is secure, and impose policies on flows. For example, a simple flow might be defined as any traffic from a given IP address. The rule governing it might be to route the flow through a given switch port. With its knowledge of the network, the controller could set up paths through the network optimized for speed, fewest number of hops or reduced latency, among other characteristics. Using OpenFlow takes control of how traffic flows through the network out of the hands of the infrastructure, the switches and routers, and puts it in the hands of the network owner (such as a corporation), individual users or individual applications.").<br><br>Shieh '088 ¶ [0028] ("Firewall modules 209A-209C may be part of a distributed firewall described above. For example, firewall modules 209A-209C may be the IO functions of a firewall while nodes 211A-211B may be firewall processing nodes. That is, modules 211A-211B may be dedicated firewall processing devices that perform some firewall processing operations such as DPI, content inspection, antivirus, etc., while firewall modules 209A-209C are responsible for routing data packets. For example, when firewall module 209B receives a packet from node 206, it may forward the packet to firewall processing node 211A for content inspection and/or forwards the packet to controller 208 for routing information. In response, firewall processing node 211A analyzes the received packet and/or further communicates with controller 208. Controller 208 may provide further routing information back to network access device 204B regarding how to route the packet. Each of the firewall processing nodes 211A-211B may further maintains a persistent connection or tunnel with controller 208, for example, using the OpenFlow communication protocol."). |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 767 of 1100

| No. | '111 Patent Claim 4 | Shieh '088 |
|---|---|---|
| | |  <br>Fig. 2A (annotation added)<br><br>Shieh '088 ¶ [0029] ("According to one embodiment, an administrator 265 configures, for example, via a controller or a management entity 208, a network access device (e.g., network access devices 204A-204C) to set up a set of filtering rules concerning whether and/or what types of packets should be forwarded to a security device and which of the security devices (e.g., security devices 211A-211B) for security inspection. In this embodiment, controller 208 is configured to manage multiple network access devices 204A-204C and/or multiple security devices 211A-211B. Alternatively, a security device, such as security device 211A, may inform a network access device, such as network access device 204B, whether subsequent packets of a particular session should be forwarded from the network access device for security inspection. A security device may perform the security inspection |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 768 of 1100

| No. | '111 Patent Claim 4 | Shieh '088 |
|---|---|---|
| | | on packets at the beginning of the flow or session, and at a certain point, the security device decides that it no longer needs to inspect further packets of the same session.").<br><br>Shieh '088 ¶ [0035] ("An embodiment of the invention also controls the communication between I/O functions and security-processing functions to enable packets to bypass security-processing function if there is no more need to inspect the packets of the connection. Some of the security functions do not need to inspect all the packets of a connection. For examples, to identify the application of a connection, there may be only need to inspect first four or five packets to make the identification. In this case, the security-processing function can notify I/O functions to bypass the security-processing function for the rest of the packets of the connections. Once the I/O function receives the notification, it will forward the packets out without redirecting the packets to the security-processing functions. This would greatly improve the performance even when security inspection is turned on.").<br><br>Shieh '088 ¶ [0036] ("During the bypass phase, the I/O function may notify the security-processing function if there are special events in the packet stream. These events could be receipt of TCP FIN or TCP RST packets, or not receiving any packets of the connection within a time threshold. The notification from I/O functions to security processing functions could help to clean up the state in the security-processing nodes.").<br><br>Shieh '088 ¶ [0042] ("In one embodiment, central controller 208 is the central place to control forwarding of the packets amongst I/O modules 301-304, security processing modules 309-311, and service processing modules 312-313. When a virtual I/O module receives a packet, according to one embodiment, it forwards the packet to central controller 208 if it cannot find an existing connection in its local cache, as shown in FIG. 5. When central controller 208 receives the packet, it decides which of security processing modules 309-311 is able to process the packets, and then forwards the packets to the designated security processing module. It also instructs the virtual I/O module to create the local cache to store connection state information so the subsequent packets of the same connection session do not need to be forwarded to central controller 208; rather, they can be directly forwarded to the proper security processing module identified in the cache.").<br><br>Shieh '088 ¶ [0049] ("In one embodiment, firewall modules 300A-300B could be distributed in different networks, even on different locations, as long as the modules can reach the module |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 769 of 1100

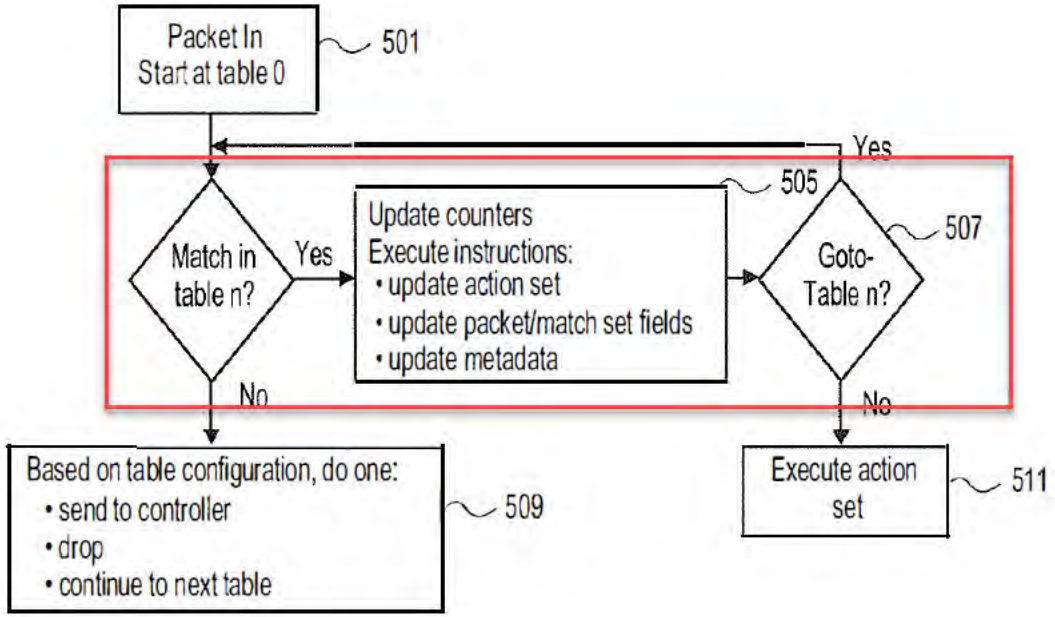
| No. | '111 Patent Claim 4 | Shieh '088 |
|---|---|---|
| | | that is next in terms of processing and the central controller. In one embodiment, virtual I/O modules and corresponding security processing modules are in a public cloud and the central controller is in a private cloud. This configuration may provide the flexibility to secure and control packets coming from the public cloud, and allow central controller having overall view of traffic from Internet as well as from internal network.").<br><br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Shieh '088 in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 4[b] of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references.<br><br>For example, Kempf discloses sending the packet from the network element to the controller or another table, in response to the packet matching the action in the flow table.<br><br>Kempf at [0044] ("FIG. 1 is a diagram of one embodiment of an example network with an OpenFlow switch, conforming to the OpenFlow 1.0 specification. The OpenFlow 1.0 protocol enables a controller 101 to connect to an OpenFlow 1.0 enabled switch 109 using a secure channel 103 and control a single forwarding table 107 in the switch 109. The controller 101 is an external software component executed by a remote computing device that enables a user to configure the Open-Flow 1.0 switch 109. The secure channel 103 can be provided by any type of network including a local area network (LAN) or a wide area network (WAN), such as the Internet.")<br><br>Kempf at [0045] ("FIG. 2 is a diagram illustrating one embodiment of the contents of a flow table entry. The forwarding table 107 is populated with entries consisting of a rule 201 defining matches for fields in packet headers; an action 203 associated to the flow match; and a collection of statistics 205 on the flow. When an incoming packet is received a lookup for a matching rule is made in the flow table 107. If the incoming packet matches a particular rule, the associated action defined in that flow table entry is performed on the packet.")<br><br>Kempf at [0046] ("A rule 201 contains key fields from several headers in the protocol stack, for example source and destination Ethernet MAC addresses, source and destination IP addresses, IP protocol type number, incoming and outgoing TCP or UDP port numbers. To |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 770 of 1100

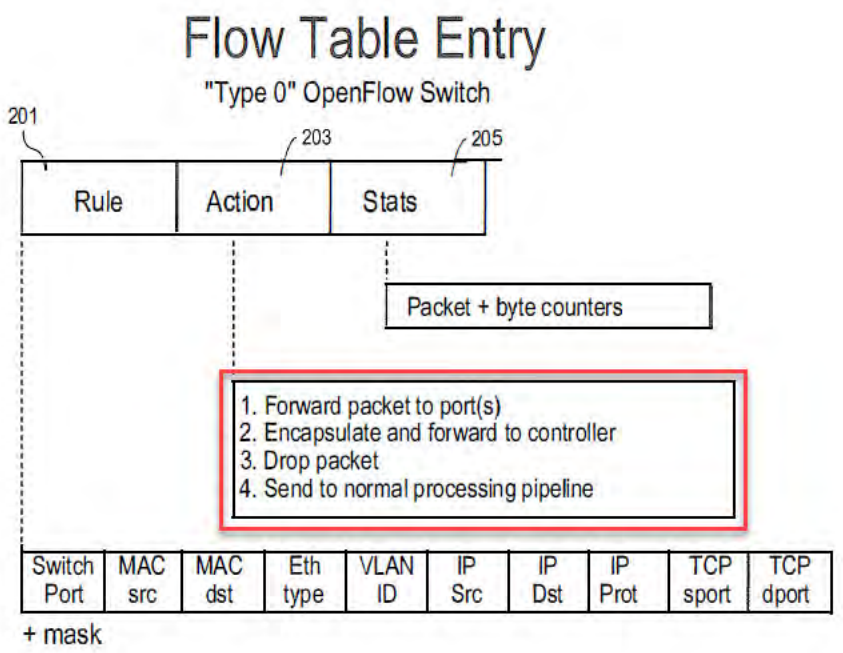
| No. | '111 Patent Claim 4 | Shieh '088 |
|---|---|---|
| | | define a flow, all the available matching fields may be used. But it is also possible to restrict the matching rule to a subset of the available fields by using wildcards for the unwanted fields.")<br><br>Kempf at [0047] ("The actions that are defined by the specification of OpenFlow 1.0 are Drop, which drops the matching packets; Forward, which forwards the packet to one or all outgoing ports, the incoming physical port itself, the controller via the secure channel, or the local networking stack (if it exists). OpenFlow 1.0 protocol data units (PDU s) are defined with a set of structures specified using the C programming language. Some of the more commonly used messages are: report switch configuration message; modify state messages (in-cluding a modify flow entry message and port modification message); read state messages, where while the system is running, the datapath may be queried about its current state using this message; and send packet message, which is used when the controller wishes to send a packet out through the datapath.")<br><br>Kempf at [0050] ("FIG. 4 illustrates one embodiment of the processing of packets through an OpenFlow 1.1 switched packet pro-cessing pipeline. A received packet is compared against each of the flow tables 401. After each flow table match, the actions are accumulated into an action set. If processing requires matching against another flow table, the actions in the matched rule include an action directing processing to the next table in the pipeline. Absent the inclusion of an action in the set to execute all accumulated actions immediately, the actions are executed at the end 403 of the packet processing pipeline. An action allows the writing of data to a metadata register, which is carried along in the packet processing pipe-line like the packet header.")<br><br>Kempf at [0091] ("When a packet header matches a rule associated with the virtual port, the GTP TEID is written into the lower 32 bits of the metadata and the packet is directed to the virtual port. The virtual port calculates the hash of the TEID and looks up the tunnel header information in the tunnel header table. If no such tunnel information is present, the packet is forwarded to the controller with an error indication. Other-wise, the virtual port constructs a GTP tunnel header and encapsulates the packet. Any DSCP bits or VLAN priority bits are additionally set in the IP or MAC tunnel headers, and any VLAN tags or MPLS labels are pushed onto the packet. The encapsulated packet is forwarded out the physical port to which the virtual port is bound.") |


~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 771 of 1100

| No. | '111 Patent Claim 4 | Shieh '088 |
|---|---|---|
| | | Kempf at [0106] ("This encapsulates the packet and sends it to the OpenFlow controller.")<br><br>Kempf at Figure 5 (annotation added)<br><br>  <br><br>FIG. 5<br><br>Kempf at Figure 2 (annotation added) |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 772 of 1100

| No. | '111 Patent Claim 4 | Shieh '088 |
|---|---|---|
| | |  <br>**FIG. 2**<br><br>For example, Swenson discloses determining by the steering device monitors flows that match one or more signatures or criteria of the packet. Swenson further discloses that when a matching flow is detected the steering device forwards the packet to the network controller.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 773 of 1100

| No. | '111 Patent Claim 4 | Shieh '088 |
|---|---|---|
| | | steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0028] ("In contrast to conventional inline TCP throughput monitoring devices that monitor every single data packets transmitted and received, the network controller 140 is an "out-of-band" computer server that interfaces with the steer-ing device 130 to selectively inspect user flows of interest. The network controller 140 may further identify user flows (e.g., among the flows of interest) for optimization. In one embodiment, the network controller 140 may be imple-mented at the steering device 130 to monitor traffic. In other embodiments, the network controller 140 is coupled to and communicates with the steering device 130 for traffic moni-toring and optimization. When queried by the steering device 130, the network controller 140 determines if a given network flow should be ignored, monitored further or optimized. Opti-mization of a flow is often decided at the beginning of the flow because it is rarely possible to switch to optimized content mid-stream once non-optimized content delivery has begun. However, the network controller 140 may determine that existing flows associated with a particular subscriber or other entity should be optimized. In turn, new flows ( e.g., resulting from seek requests in media, new media requests, resume after pause, etc.) determined to be associated with the entity may be optimized. The network controller 140 uses the net-work state as well as historical traffic data in its decision for monitoring and optimization. Knowledge on the current net-work state, such as congestion, deems critical when it comes to data optimization.")<br><br>Swenson at [0029] ("As a flow is sent to the network controller 140 for inspection, historical network traffic data stored at the net-work controller 140 may be searched. The historical network traffic data includes information such as subscriber informa-tion, the cell towers to which the user devices attached, rout-ers through which the traffic is passing, geography regions, the backhaul segments, and time-of-day of the flows. For example, in a mobile network, the cell tower to which a user device is attached can be most useful, since it is the location where most congestion occurs due to limited bandwidth and high cost of the radio access network infrastructure. The network controller 140 looks into the historical traffic data for the average of the bandwidth per user at the particular cell tower. The network controller |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 774 of 1100

| No. | ’111 Patent Claim 4 | Shieh ’088 |
|---|---|---|
| | | 140 can then estimate the amount ofbandwidth or degree of congestion for the new flow based on the historical record.”)<br><br>Swenson at [0038] (“Turning back to FIG. 1, the network controller 140 allows network operators to apply fine granular optimization policies to ensure high quality of experience (QoE) based on cell tower congestion, device types, subscriber profiles and service plans with lower hardware and software costs. The architecture of the network controller 140 provides an excel-lent fit for the net neutrality guideline of"reasonable network management", and better compliance to the copyright law (DMCA) than solutions that rely on long-term caching. Hav-ing the ability of monitoring network traffic on a per sub-scriber, per flow, or per video file basis, the network controller 140 also selectively monitors and optimizes only a subset of traffic that benefits from optimization the most, thus achiev-ing both scalability and efficiency for optimization at a com-petitive price-point. The core element of the network control-ler 140 lies in its mechanisms for congestion detection and mitigation, which allows optimization resources to be utilized in the most efficient and surgical manner.”)<br><br>Swenson at [0039] (“Referring now to FIG. 3, it illustrates one embodi-ment of an example architecture of the network controller 140 for providing selective real-time network monitoring and subscriber identification. The network controller 140 com-prises a flow analyzer 312, a policy engine 314, a steering device interface 316, a video optimizer redirector 318, a flow cache 322, and a subscriber log 324. In other embodiments, the network controller 140 may include additional, fewer, or different components for various applications. Conventional components such as network interfaces, security functions, failover servers, management and network operations con-soles, and the like are not shown so as to not obscure the details of the system architecture.”)<br><br>Swenson at [0045] (“The steering device interface 316 interacts with an external routing appliance, such as the steering device 130 to divert portions of the network traffic ( e.g., large object net-work flows). Existing routing appliances in most carrier net-works are designed to handle large amounts of network traf-fic. They are not, however, ideal devices to operate for monitoring and analysis individual flows. Through the steer-ing device interface 316, the network controller 140 may communicate with the external routing appliances, such as the steering device 130, to steer a portion of network traffic to the network controller 140 when certain conditions are met. Generally, network flows of interest to the network controller 140 |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 775 of 1100

| No. | '111 Patent Claim 4 | Shieh '088 |
|---|---|---|
| | | contain larger media objects, such as videos and images. In one embodiment, the smaller flows, such as web page and text information, are not exchanged over the steering device interface 316.")<br><br>Swenson at [0059] ("In one embodiment, as the steering device 130 monitors network responses, it is looking for flows that match one or more signatures for video and images. When a match-ing flow is detected, the steering device 130 forwards the HTTP request and a portion of the HTTP response to the network controller 140 over the ICAP client interface 404. After receiving the request and the portion of response at the ICAP server interface 406, the flow analyzer 312 of the net-work controller 140 performs a deep flow inspection to deter-mine if the flow is worth bandwidth monitoring and/or user detection. For example, the flow inspection performed by the flow analyzer 312 may determine if the flow indeed contains large or medium object ( e.g., larger than 50 kB), and/or if the source IP address of the flow is from a user or a group of users that are required to be monitored by policies. The flow ana-lyzer 312 may also determine if the flow needs to be opti-mized based on historical flow statistical data.")<br><br>Swenson at [0060] ("If the flow is deemed of interest, the steering device 130 is notified to steer the flow through the network controller 140. This is known as the "continue" working mode for bandwidth monitoring. In the "continue" mode, the network controller 140 interfaces with the steering device 130 to func-tion, on-demand, as a traditional inline network element for flows deemed of interest. Thus, the network controller 140 ingests the network flow for inspection and subsequently forwards the network flow on the network response path. For example, for this particular flow, the origin server 160 responds to the user request by sending video or images over the network link 413 to the steering device 130, which for-wards the video or images to the network controller 140 over a network link 414. After the network controller 140 updates the flow statistics, the video or images are returned to the steering device 130 over a network link 415, which transmits the video or images to the user device 110 over the network link 416.")<br><br>Swenson at [0065] ("FIG. 5 is a block diagram illustrating an example event trace of "continue" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 512 to retrieve content from the origin server 160. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 776 of 1100

| No. | '111 Patent Claim 4 | Shieh '088 |
|---|---|---|
| | | The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 512 to the intended origin server 160 and receives a response 514 back from the origin server 160. The steering device 130 then sends an ICAP request message 516 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 518. Upon receiving the instruc-tion, the steering device 130 re-writes the response 514 to an HTTP redirect response 520, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 520 directly to the user device 110. In case the flow dose not contain video or image objects, or the network controller 140 determines not to monitor the flow, the steering device 13 0 would forward the response to the user device 110.")<br><br>Swenson at [0069] ("FIG. 6 is a block diagram illustrating an example event trace of "counting" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 612 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 612 to the intended origin server 160 and receives a response 614 back from the origin server 160. The steering device 130 then sends an ICAP request message 616 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 618. Upon receiving the instruc-tion, the steering device 130 re-writes the response 614 to an HTTP redirect response 620, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 620 directly to the user device 110. In case the flow dose not contain video or image objects that need to be redirected, the steering device 130 would forward the response to the user device 110.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 777 of 1100

| No. | '111 Patent Claim 4 | Shieh '088 |
|---|---|---|
| 4[c] | responsive to receiving the packet, analyzing the packet, by the controller; | Shieh '088 discloses responsive to receiving the packet, analyzing the packet, by the controller.<br><br>For example, Shieh '088 discloses that when the central controller receives the packet, it decides which of the security processing modules is able to process the packets, and then forwards the packets to the designated security processing module.<br><br>Shieh '088 ¶ [0028] ("Firewall modules 209A-209C may be part of a distributed firewall described above. For example, firewall modules 209A-209C may be the IO functions of a firewall while nodes 211A-211B may be firewall processing nodes. That is, modules 211A-211B may be dedicated firewall processing devices that perform some firewall processing operations such as DPI, content inspection, antivirus, etc., while firewall modules 209A-209C are responsible for routing data packets. For example, when firewall module 209B receives a packet from node 206, it may forward the packet to firewall processing node 211A for content inspection and/or forwards the packet to controller 208 for routing information. In response, firewall processing node 211A analyzes the received packet and/or further communicates with controller 208. Controller 208 may provide further routing information back to network access device 204B regarding how to route the packet. Each of the firewall processing nodes 211A-211B may further maintains a persistent connection or tunnel with controller 208, for example, using the OpenFlow communication protocol.").<br><br>Shieh '088 ¶ [0042] ("In one embodiment, central controller 208 is the central place to control forwarding of the packets amongst I/O modules 301-304, security processing modules 309-311, and service processing modules 312-313. When a virtual I/O module receives a packet, according to one embodiment, it forwards the packet to central controller 208 if it cannot find an existing connection in its local cache, as shown in FIG. 5. When central controller 208 receives the packet, it decides which of security processing modules 309-311 is able to process the packets, and then forwards the packets to the designated security processing module. It also instructs the virtual I/O module to create the local cache to store connection state information so the subsequent packets of the same connection session do not need to be forwarded to central controller 208; rather, they can be directly forwarded to the proper security processing module identified in the cache."). |

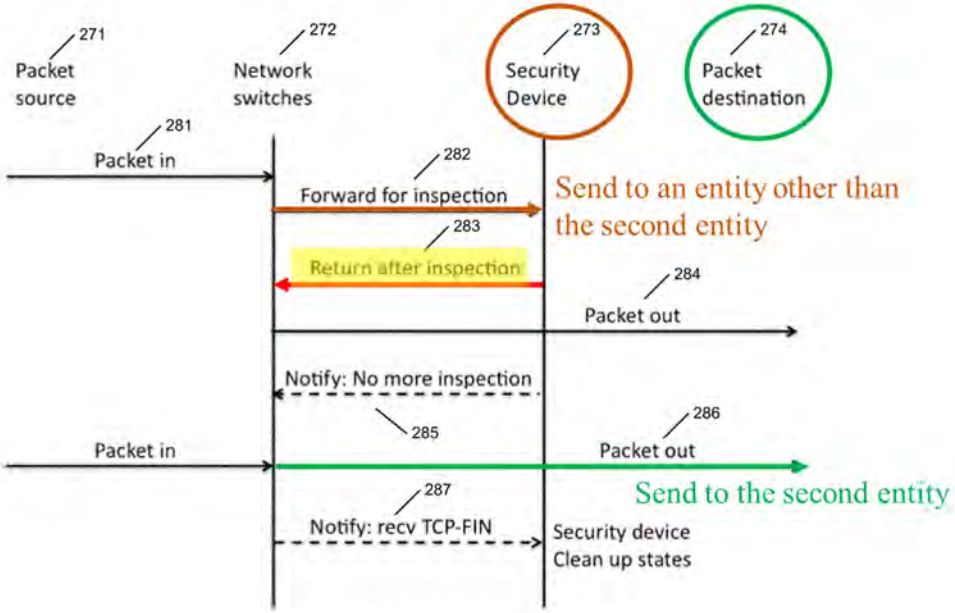


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 778 of 1100

| No. | '111 Patent Claim 4 | Shieh '088 |
|---|---|---|
| 4[d] | sending the packet, by the controller, to the network node; and | Shieh '088 discloses sending the packet, by the controller, to the network node.<br><br>For example, Shieh '088 discloses, after being analyzed by the security device, returning the packet to the security device.<br><br>Shieh '088 ¶ [0037] "(FIG. 2B is a processing flow diagram illustrating a process of security inspection according to one embodiment of the invention. Referring to FIG. 2B, as an example, network switch 272 may represent any of network access devices 204A-204C and security device 273 may represents any of security processing devices 211A-211B as described above with respect to FIG. 2A. When device 272 receives a packet from a source node 271 via transaction 281, device 272 may determine whether the packet should be forwarded to security device 273. For example, device 272 may look up in its session table such as the one as shown in FIG. 5 to determine whether a bypass flag has been set to a predetermined value. If the bypass flag matches the predetermined value, the packet is forwarded to security device 273 via path 282; otherwise, the packet is routed to destination node 274. Alternatively, if there is no entry in the session table corresponding to the current session, the packet will also be transmitted to security device 273. After network device 272 receives a response from security device 273 via path 283, dependent upon the response, the packet may then be routed to destination node 274 via path 284. These processes may continue until a notification is received from security device 273 via path 285 indicating that it no longer wishes to receive further packets of the same session for inspection, such that subsequent packets will be directly routed to destination node 274 via path 286 without routing to security device 273. If there are certain events that have been registered from security device 273, network device 272 may notify security device 274 via path 287 upon detecting the registered events."). |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 779 of 1100

| No. | '111 Patent Claim 4 | Shieh '088 |
|---|---|---|
| | |  <br>FIG. 2B<br>Fig. 2B (annotation added) |
| 4[e] | responsive to receiving the packet, sending the packet, by the network node, to the second entity. | Shieh '088 discloses responsive to receiving the packet, sending the packet, by the network node, to the second entity.<br><br>For example, Shieh '088 discloses routing the security device-approved packet to the packet destination.<br><br>Shieh '088 ¶ [0037] "(FIG. 2B is a processing flow diagram illustrating a process of security inspection according to one embodiment of the invention. Referring to FIG. 2B, as an example, network switch 272 may represent any of network access devices 204A-204C and security device 273 may represents any of security processing devices 211A-211B as described above with respect to FIG. 2A. When device 272 receives a packet from a source |

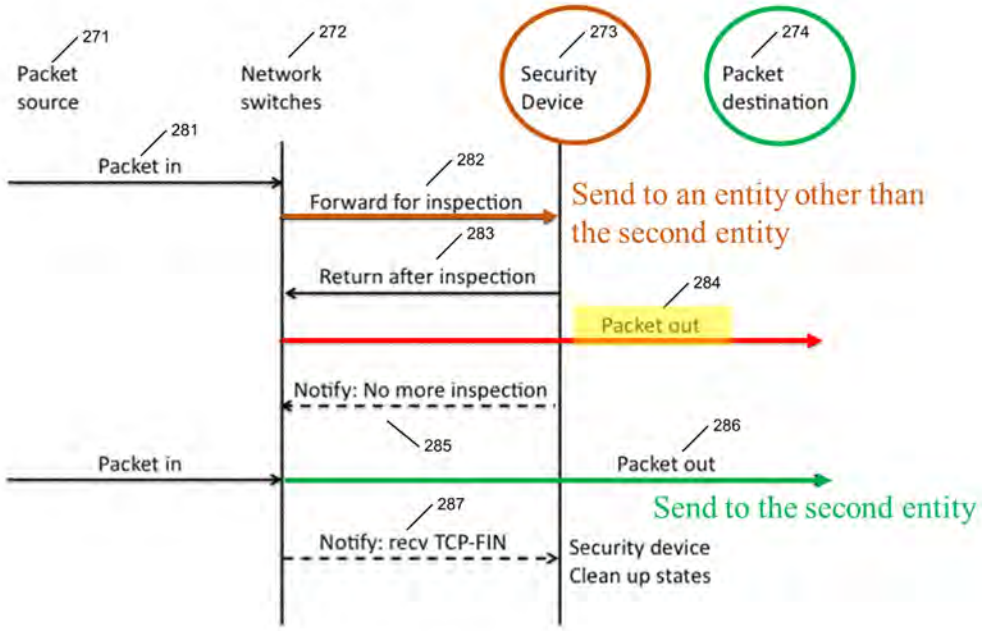


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 780 of 1100

| No. | '111 Patent Claim 4 | Shieh '088 |
|---|---|---|
| | | node 271 via transaction 281, device 272 may determine whether the packet should be forwarded to security device 273. For example, device 272 may look up in its session table such as the one as shown in FIG. 5 to determine whether a bypass flag has been set to a predetermined value. If the bypass flag matches the predetermined value, the packet is forwarded to security device 273 via path 282; otherwise, the packet is routed to destination node 274. Alternatively, if there is no entry in the session table corresponding to the current session, the packet will also be transmitted to security device 273. After network device 272 receives a response from security device 273 via path 283, dependent upon the response, the packet may then be routed to destination node 274 via path 284. These processes may continue until a notification is received from security device 273 via path 285 indicating that it no longer wishes to receive further packets of the same session for inspection, such that subsequent packets will be directly routed to destination node 274 via path 286 without routing to security device 273. If there are certain events that have been registered from security device 273, network device 272 may notify security device 274 via path 287 upon detecting the registered events."). |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 781 of 1100

| No. | '111 Patent Claim 4 | Shieh '088 |
|---|---|---|
| | |  <br>FIG. 2B<br>Fig. 2B (annotation added) |

| No. | '111 Patent Claim 5 | Shieh '088 |
|---|---|---|
| 5 | The method according to claim 1, further comprising responsive to the packet satisfying the criterion and to the instruction, sending the packet or a portion thereof, by the | Shieh '088 discloses the method according to claim 1, responsive to the packet satisfying the criterion and to the instruction, sending the packet or a portion thereof, by the network node, to the controller.<br><br>For example, Shieh '088 discloses bypass rules, such as when the bypass flag matches a predetermined value, that when satisfied, instructs that the packet will be sent from the virtual I/O module to a central controller capable of deep packet inspection responsive to the instruction. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Shieh '088 is found to not meet this |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 782 of 1100

| No. | '111 Patent Claim 5 | Shieh '088 |
|---|---|---|
| | network node, to the controller. | limitation, further comprising responsive to the packet satisfying the criterion and to the instruction, sending the packet or a portion thereof, by the network node, to the controller would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>*See supra* at 1.<br><br>Shieh '088 ¶ [0042] ("In one embodiment, central controller 208 is the central place to control forwarding of the packets amongst I/O modules 301-304, security processing modules 309-311, and service processing modules 312-313. When a virtual I/O module receives a packet, according to one embodiment, it forwards the packet to central controller 208 if it cannot find an existing connection in its local cache, as shown in FIG. 5. When central controller 208 receives the packet, it decides which of security processing modules 309-311 is able to process the packets, and then forwards the packets to the designated security processing module. It also instructs the virtual I/O module to create the local cache to store connection state information so the subsequent packets of the same connection session do not need to be forwarded to central controller 208; rather, they can be directly forwarded to the proper security processing module identified in the cache.").<br><br>Shieh '088 ¶ [0039] "(An I/O module running within a virtual machine is referred to herein as a virtual I/O module. Each of virtual I/O modules 301-304 receives packets from any of servers 321-324 of LAN 320 and sends packets to external network 315 outside of the firewall. In one embodiment, each of I/O modules 301-304 keeps a local cache (e.g., caches 305-308) storing location(s) of a security processing module(s) (e.g., security processing modules 309-311) for each connection session. A cache maintained by each I/O module contains a forwarding table mapping certain connection sessions to any of security modules 309-311. An example of a forwarding table is shown in FIG. 5. Upon receiving a packet, an I/O module performs a packet classification to find out the associated connection and forwards the packet to the corresponding security processing module identified by the forwarding table. If it cannot find the connection in its local cache, the packets are forwarded to central controller 208 for processing. In such a case, controller 208 assigns the connection to one of security processing modules 309-311 based on one or more of a variety of factors such as load balancing. The virtual I/O modules 302-304 can be located at multiple locations of the networks to receive and send out packets.").<br><br>Orckit Exhibit 2018<br>Cisco Systems v. Orckit Corp.<br>IPR2023-00714, Page 783 of 1100 |

| No. | '111 Patent Claim 5 | Shieh '088 |
|---|---|---|
| | | Shieh '088 ¶ [0036] "(During the bypass phase, the I/O function may notify the security-processing function if there are special events in the packet stream. These events could be receipt of TCP FIN or TCP RST packets, or not receiving any packets of the connection within a time threshold. The notification from I/O functions to security processing functions could help to clean up the state in the security-processing nodes.").<br><br>Shieh '088 ¶ [0037] "(FIG. 2B is a processing flow diagram illustrating a process of security inspection according to one embodiment of the invention. Referring to FIG. 2B, as an example, network switch 272 may represent any of network access devices 204A-204C and security device 273 may represents any of security processing devices 211A-211B as described above with respect to FIG. 2A. When device 272 receives a packet from a source node 271 via transaction 281, device 272 may determine whether the packet should be forwarded to security device 273. For example, device 272 may look up in its session table such as the one as shown in FIG. 5 to determine whether a bypass flag has been set to a predetermined value. If the bypass flag matches the predetermined value, the packet is forwarded to security device 273 via path 282; otherwise, the packet is routed to destination node 274. Alternatively, if there is no entry in the session table corresponding to the current session, the packet will also be transmitted to security device 273. After network device 272 receives a response from security device 273 via path 283, dependent upon the response, the packet may then be routed to destination node 274 via path 284. These processes may continue until a notification is received from security device 273 via path 285 indicating that it no longer wishes to receive further packets of the same session for inspection, such that subsequent packets will be directly routed to destination node 274 via path 286 without routing to security device 273. If there are certain events that have been registered from security device 273, network device 272 may notify security device 274 via path 287 upon detecting the registered events."). |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 784 of 1100

| No. | '111 Patent Claim 5 | Shieh '088 |
|---|---|---|

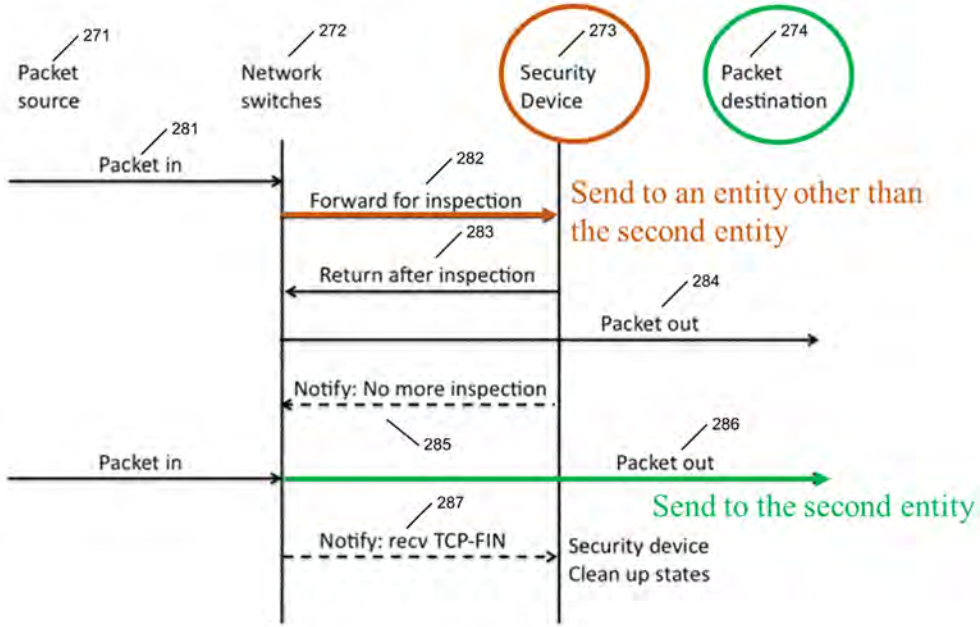



FIG. 2B

Fig. 2B (annotation added)

Shieh '088 Claim 1 ("A computer-implemented method, comprising:
receiving at a network access device a packet from a source node destined to a destination node;
examining a data structure maintained by the network access device to determine whether the data structure stores a data member having a predetermined value, the data member indicating whether the packet should undergo security processing;
if the data member matches the predetermined value, transmitting the packet to a security device associated with the network access device to allow the security device to perform content inspection, and
in response to a response received from the security device, routing the packet to the destination node dependent upon the response; and



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 785 of 1100

| No. | '111 Patent Claim 5 | Shieh '088 |
|---|---|---|
| | | transmitting the packet to the destination node without forwarding the packet to the security device, if the data member does not match the predetermined value."). |

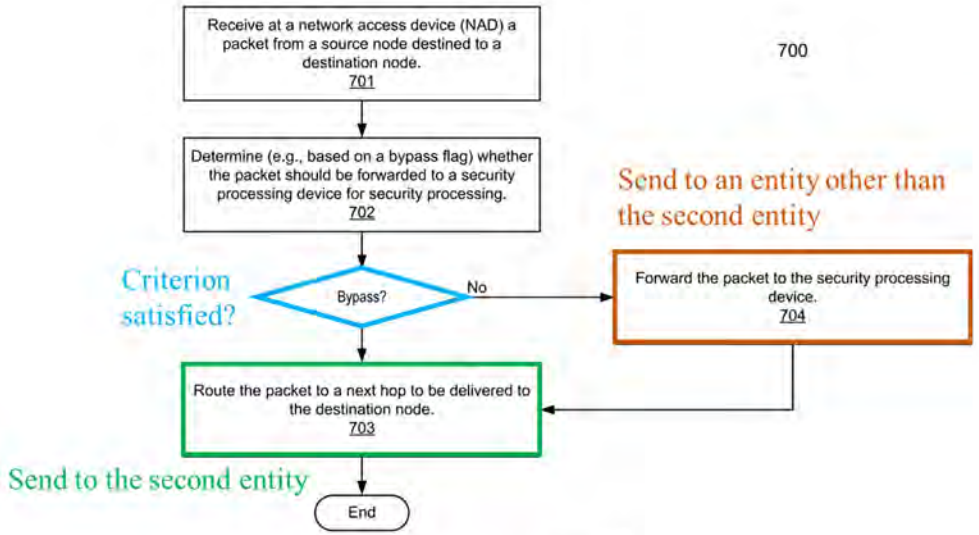



FIG. 7

Fig. 7 (annotation added)

Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Shieh '088 in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 5 of Exhibit E-4 renders the claim, including the present limitation, obvious. Below is an example.

For example, Copeland discloses sending packets and sampled packet headers to the intrusion detection engine on the monitoring appliance based on matching predetermined values associated with a concern index.

Copeland at [0067] ("The host servers 130 are directly or indirectly coupled to one or more network devices 135 such as routers or switches that support providing a sampled data stream and



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 786 of 1100

| No. | '111 Patent Claim 5 | Shieh '088 |
|---|---|---|
| | | such as that provided by sFlow. In a typical preferred configuration for the present invention, a monitoring appli-ance 150 operating a flow-based intrusion detection engine 155 is receiving sampled packet headers from one or more network devices 135. The monitoring appliance 150 moni-tors the communications between the host server 130 and other hosts 120, 110 in the attempt to detect intrusion activity.")<br><br>Copeland [0079] ("Large packets tend to be fragmented by networks that cannot handle a large packet size. A 16-bit packet identification is used to reassemble fragmented packets. Three one-bit set of fragmentation flags control whether a packet is or may be fragmented. The 13-bit fragment offset is a sequence number for the 4-byte words in the packet when reassembled. In a series of fragments, the first offset will be zero.")<br><br>Copeland at [0097] ("The described TCP session 300 of FIG. 3 is a generic TCP session in which a network might engage. In accordance with the invention, flow data is collected about the session to help determine if the communication is abnormal. In the preferred embodiment, information such as the total number of packets sent, the total amount of data sent, the session start time and duration, and the TCP flags set in all of the packets, are collected, stored in the database 160, and analyzed to determine if the communication was suspicious. If a communication is deemed suspicious, i.e. it meets predetermined criteria, a predetermined concern index value associated with a determined category of suspicious activity is added to the cumulated CI value associated with the host that made the communication.")<br><br>Copeland at [0120] ("The sampled packet headers sent from the sFlow agent are captured and processed by the sample packet collector 505 in order to create a "Packet Data" data struc-ture that includes the sFlow agent source of the packets, the header of the sampled packets, and other information avail-able from the sFlow data stream that may be important. For<br>example, one data field that is optionally available pr vides the username of the user using the computer at the time of the communications. This information is extremely useful in some environments subject to regulatory requirements and monitoring of the communications on the network. In this case the username will be stored as "supplementary infor-mation" for auditing purposes in the flow data. Other infor-mation, including the sampling device and the physical port on which the communications was detected may also be retained for other uses such as mitigation, where a host may be removed from the network.") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 787 of 1100

| No. | '111 Patent Claim 5 | Shieh '088 |
|---|---|---|
| | | Copeland at [0126]-[0129] ("If a particular packet 101 being processed by the packet classifier 510 matches a particular entry or record in the flow data structure 162, data from that particular packet 101 is used to update the statistics in the corresponding flow data structure record. A packet 101 is considered to match to a flow data structure record if both IP numbers match and the source of the sampled packet matches and:<br><br>(1) both port numbers match and no port is marked as the "server" port, or<br>(2) the port number previously marked as the "server" port matches, or<br>(3) one of the port numbers matches, but the other does not, and the neither port number has been marked as the server port (in this case the matching port number is marked as the "server" port).")<br><br>Copeland at [0144] ("Concern index (CI) values calculated from packet anomalies also add to a host's accumulated concern index value. Table II of FIG. 7 shows one scheme for assigning concern index values due to other events revealed by the flow analysis. For example, there are many combinations of TCP flag bits that are rarely or never seen in valid TCP connections. When the packet classifier thread 510 recog-nizes one of these combinations, it directly adds a predeter-mined value to the sending host's accumulated concern index value. When the packet classifier thread 510 searches along the flow linked-list (i.e. flow data 162) for a match to the current packet 101, it keeps count of the number of flows active with matching IP addresses but no matching port number. If this number exceeds a predetermined threshold value (e.g., 4) and is greater than the previous number noticed, CI is added for an amount corresponding to a "port scan." A bit in the host record is set to indicate that the host has received CI for "port scanning."")<br><br>Copeland at [0150] ("A preferred hardware configuration 800 of an embodiment that executes the functions of the above-described flow-based engine is described in reference to FIG. 8. FIG. 8 illustrates a typically hardware configuration 800 for a network intrusion detection system. A monitoring appliance 150 serves as a pass-by filter of network traffic. A network device 135, such as a router or switch supporting sFlow provides the location for connecting the monitoring appliance 150 to the network 899 for monitoring the network traffic.")<br><br>Copeland at [0159]-[0162] ("A packet 101 is considered to match to a flow data structure record if both IP numbers match and the source of the sampled data matches and: |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 788 of 1100

| No. | '111 Patent Claim 5 | Shieh '088 |
|---|---|---|
| | | (a). both port numbers match and no port is marked as the "server" port, or<br>(b). the port number previously marked as the "server" port matches, or<br>(c). one of the port numbers matches, but the other does not, and the neither port number has been marked as the server port (in this case the matching port number is marked as the "server" port).") |

| No. | '111 Patent Claim 6 | Shieh '088 |
|---|---|---|
| 6 | The method according to claim 5, further comprising storing the received packet or a portion thereof, by the controller, in a memory. | Shieh '088 discloses the method according to claim 5, further comprising storing the received packet or a portion thereof, by the controller, in a memory.<br><br>For example, Shieh '088 instructs the virtual I/O module to create a local cache to store connection state information. A person of ordinary skill in the art would understand that the controller also stores packet information in memory. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Shieh '088 is found to not meet this limitation, further comprising storing the received packet or a portion thereof, by the controller, in a memory would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>*See supra* at 5.<br><br>Shieh '088 ¶ [0039] ("An I/O module running within a virtual machine is referred to herein as a virtual I/O module. Each of virtual I/O modules 301-304 receives packets from any of servers 321-324 of LAN 320 and sends packets to external network 315 outside of the firewall. In one embodiment, each of I/O modules 301-304 keeps a local cache (e.g., caches 305-308) storing location(s) of a security processing module(s) (e.g., security processing modules 309-311) for each connection session. A cache maintained by each I/O module contains a forwarding table mapping certain connection sessions to any of security modules 309-311. An example of a forwarding table is shown in FIG. 5. Upon receiving a packet, an I/O module performs a packet classification to find out the associated connection and forwards the packet to the corresponding security processing module identified by the |

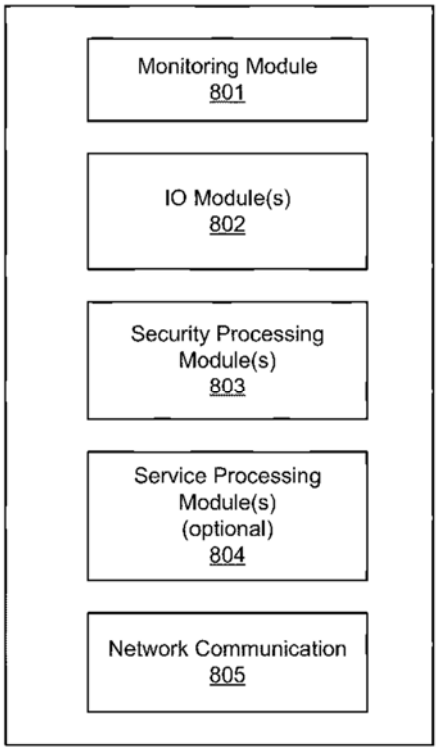


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 789 of 1100

| No. | '111 Patent Claim 6 | Shieh '088 |
|---|---|---|
| | | forwarding table. If it cannot find the connection in its local cache, the packets are forwarded to central controller 208 for processing. In such a case, controller 208 assigns the connection to one of security processing modules 309-311 based on one or more of a variety of factors such as load balancing. The virtual I/O modules 302-304 can be located at multiple locations of the networks to receive and send out packets.").<br><br>Shieh '088 ¶ [0042] ("In one embodiment, central controller 208 is the central place to control forwarding of the packets amongst I/O modules 301-304, security processing modules 309-311, and service processing modules 312-313. When a virtual I/O module receives a packet, according to one embodiment, it forwards the packet to central controller 208 if it cannot find an existing connection in its local cache, as shown in FIG. 5. When central controller 208 receives the packet, it decides which of security processing modules 309-311 is able to process the packets, and then forwards the packets to the designated security processing module. It also instructs the virtual I/O module to create the local cache to store connection state information so the subsequent packets of the same connection session do not need to be forwarded to central controller 208; rather, they can be directly forwarded to the proper security processing module identified in the cache.").<br><br>Shieh '088 ¶ [0060] ("Some portions of the preceding detailed descriptions have been presented in terms of algorithms and symbolic representations of operations on data bits within a computer memory. These algorithmic descriptions and representations are the ways used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. An algorithm is here, and generally, conceived to be a self-consistent sequence of operations leading to a desired result. The operations are those requiring physical manipulations of physical quantities.").<br><br>Shieh '088 ¶ [0057] ("Referring to FIG. 8, the memory 460 includes a monitoring module 801 which when executed by a processor is responsible for performing traffic monitoring of traffic from the VMs as described above. Memory 460 also stores one or more IO modules 802 which, when executed by a processor, is responsible for performing forwarding inbound and outbound packets. Memory 460 further stores one or more security processing modules 803 which, when executed by a processor, is responsible for security processes on the packets provided by IO modules 802. Memory 460 also stores one or more optional service processing modules 804, which when executed by a processor performs a |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 790 of 1100

| No. | '111 Patent Claim 6 | Shieh '088 |
|---|---|---|
| | | particular security process on behalf of security processing modules 803. The memory also includes a network communication module 805 used for performing network communication and communication with the other devices (e.g., servers, clients, etc.).").<br>  <br><br>FIG. 8<br>Fig. 8 |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 791 of 1100

| No. | '111 Patent Claim 6 | Shieh '088 |
|---|---|---|
| | | Shieh '088 Claim 17 ("The system of claim 15, further comprising a controller to manage the network access device and the security device, wherein the network access device is further to<br>receive a message having a data value from the controller, the data value indicating whether the network access device should forward further packets to the security device for security inspection, and<br>store the data value in the data member of the data structure.").<br><br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Shieh '088 in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 6 of Exhibit E-4 renders the claim, including the present limitation, obvious. Below is an example.<br><br>For example, Swenson discloses the network controller storing historical network traffic data based on received packet flows.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0028] ("In contrast to conventional inline TCP throughput monitoring devices that monitor every single data packets transmitted and received, the network controller 140 is an "out-of-band" computer server that interfaces with the steer-ing device 130 to selectively inspect user flows of interest. The network controller 140 may further identify user flows (e.g., among the flows of interest) for optimization. In one embodiment, the network controller 140 may be imple-mented at the steering device 130 to monitor traffic. In other |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 792 of 1100

| No. | '111 Patent Claim 6 | Shieh '088 |
|---|---|---|
| | | embodiments, the network controller 140 is coupled to and communicates with the steering device 130 for traffic moni-toring and optimization. When queried by the steering device 130, the network controller 140 determines if a given network flow should be ignored, monitored further or optimized. Opti-mization of a flow is often decided at the beginning of the flow because it is rarely possible to switch to optimized content mid-stream once non-optimized content delivery has begun. However, the network controller 140 may determine that existing flows associated with a particular subscriber or other entity should be optimized. In turn, new flows ( e.g., resulting from seek requests in media, new media requests, resume after pause, etc.) determined to be associated with the entity may be optimized. The network controller 140 uses the net-work state as well as historical traffic data in its decision for monitoring and optimization. Knowledge on the current net-work state, such as congestion, deems critical when it comes to data optimization.")<br><br>Swenson at [0029] ("As a flow is sent to the network controller 140 for inspection, historical network traffic data stored at the net-work controller 140 may be searched. The historical network traffic data includes information such as subscriber informa-tion, the cell towers to which the user devices attached, rout-ers through which the traffic is passing, geography regions, the backhaul segments, and time-of-day of the flows. For example, in a mobile network, the cell tower to which a user device is attached can be most useful, since it is the location where most congestion occurs due to limited bandwidth and high cost of the radio access network infrastructure. The network controller 140 looks into the historical traffic data for the average of the bandwidth per user at the particular cell tower. The network controller 140 can then estimate the amount ofbandwidth or degree of congestion for the new flow based on the historical record.")<br><br>Swenson at [0044] ("Additionally, historical flow data over a longer term helps the flow analyzer 312 to determine repeating patterns and heat-maps of certain network sections and to predict when they are under congestion. In this case, the flow statis-tics stored in the flow cache 322 can be mapped against traffic categories for analysis, for example, long-term running aver-ages of video flow bandwidth help determine suitability for optimization. Furthermore, estimated bandwidth per user ( or per cell-ID, per tower, or per router) over time may be metrics calculated by the flow analyzer 312 in order to determine short term needs for optimization. For example, the flow analyzer 312 may determine to being optimizing flows asso-ciated with a particular cell-ID (or those flows for identified high-bandwidth users on |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 793 of 1100

| No. | '111 Patent Claim 6 | Shieh '088 |
|---|---|---|
| | | the cell-ID) in response to a thresh-old number of high-bandwidth users connecting to a same cell tower corresponding to the cell-ID. The reason why flow analyzer 312 selectively monitors large flows lies in the real-ization that TCP statistics for small objects, which make up most web flows, can be misleading and cause huge errors in throughput estimations.")<br><br>Swenson at [0046] ("The flow cache 322 stores monitored flow informa-tion, which is updated for a flow with each associated trans-action from the steering device 13 0. In one embodiment, data in the flow cache is stored in a map indexed by a hash, which can be up to 64-bit or longer. An entry in the flow cache map may be organized as a linked list to allow hash collisions. Alternatively, fewer bits in the hash index can also be used to speed up binary search in the flow cache map. For example, instead of using 64-bit hash index, which requires at worst 64 steps to find a node, the hash index can be reduced to 16-24 bits. There will be more hash collisions, hence the longer linked list. Other embodiments may use other type of maps or binary trees instead of the linked list to further optimize the hash collision searches.")<br><br>Swenson at [0059] ("In one embodiment, as the steering device 130 monitors network responses, it is looking for flows that match one or more signatures for video and images. When a match-ing flow is detected, the steering device 130 forwards the HTTP request and a portion of the HTTP response to the network controller 140 over the ICAP client interface 404. After receiving the request and the portion of response at the ICAP server interface 406, the flow analyzer 312 of the net-work controller 140 performs a deep flow inspection to deter-mine if the flow is worth bandwidth monitoring and/or user detection. For example, the flow inspection performed by the flow analyzer 312 may determine if the flow indeed contains large or medium object ( e.g., larger than 50 kB), and/or if the source IP address of the flow is from a user or a group of users that are required to be monitored by policies. The flow ana-lyzer 312 may also determine if the flow needs to be opti-mized based on historical flow statistical data.")<br><br>Swenson at [0061] ("Once a flow is reported to the network controller 140, a flow cache entry is created for the flow in the flow cache 322. The flow cache entry keeps track of the flow and its associated bandwidth. For a flow that is marked in "continue" mode, each time the steering device 130 forwards a next portion of the flow payload to the network controller 140, the flow cache 3 22 updates the number of bytes for transmitted in the flow. By monitoring the number of bytes per flow over time, the flow analyzer 312 is capable of determining an |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 794 of 1100

| No. | ’111 Patent Claim 6 | Shieh ’088 |
|---|---|---|
| | | estimate value of bandwidth associated with flow. Further-more, since the steering device 130 does not have infinite packet buffers, if congestion happens on the network link 416 from the steering device 130 to the user device 110, the TCP congestion control mechanism kicks in at the steering device 130, which may slows down and/or eventually stop receiving data over the network link 413 from origin server 160. During the congestion, the steering device 130 would not forward any data to the network controller 140, since the link 416 is congested and the network controller 140 would not be able to transmit data to the user device 110. Therefore, as an inline element, the network controller 140 can detect network con-gestions and estimate bandwidth associated with any flows of interest selected by the network controller 140. However, in the "continue" mode, the network controller 140 does not modify and transform the HTTP messaged it receives over the ICAP interface. The network controller 140 simply updates the flow statistics and returns the video or images to the steering device 130 for transmission to the user device 110.”)<br><br>For example, Chua ‘877 discloses logging and storing the packets, representative data of the packets, paths, and programs.<br><br>Chua ’877 at 7:28-54 (“SDN controller 112 may receive data as input from service devices 116. For example, SDN controller 112 may be con-figured to receive data from an intrusion detection system (IDS) device, a Denial of Service (DoS) device, a Distributed Denial of Service (DDoS) device, an intrusion prevention system (IPS) device, or the like. Based on this information, SDN controller 112 may make network enforcement decisions for specific traffic flows. That is, SDN controller 112 may program network devices of SDN 106 to perform pro-grammed actions on packets of a packet flow based on this data. Such programmed actions may include:<br><br>Allow-explicitly allow a certain network flow to proceed to its destination<br>Block-explicitly block a certain flow from traversing SDN 106<br>Mirror-allow the traffic, but send a copy of the traffic for deeper inspection or recording to, e.g., one of service devices 116<br>Redirect-redirect the traffic to another network (such as a honeypot device or other device of service devices 116) for either inspection or to keep a potential hacker 'busy' to determine if there is a real security threat.<br>Transform-modify or translate values of headers of packets in the network flow |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 795 of 1100

| No. | '111 Patent Claim 6 | Shieh '088 |
|---|---|---|
| | | Encapsulate-encapsulate packets in the network flow with a particular header")<br><br>Chua '877 at 7:55-61 ("SDN controller 112 may also log the programmed actions and information used to make the enforcement decisions, present the information to administrator 114 (or other user), and/or export data representative of the logs or results to a third party application, which may or may not include the application that sent an event that triggered SDN controller 112 to enforce a new policy.")<br><br>Chua '877 at 21:32-52 ("As another example, switch 210 is communicatively coupled to IDS 214 via connections 224, 226. Connection 224 may represent port 9, while connection 226 may represent port 10. After SDN controller 218 determines that a particular packet flow should be inspected for intrusion detection, SDN controller 218 may program switch 210 to direct packets of the packet flow to IDS 214. SDN controller 218 may further configure IDS 214 to send data back to switch 210, such as data indicating whether packets of the packet flow represent a network attack and/or data of the packet flow after the data has been inspected. In some examples, data of the packet flow representing an attack may be dropped or forwarded to a containment device, rather than being forwarded along the original packet flow. The data indicating whether the packet flow represents an attack may be forwarded back to SDN controller 218, to admin workstation 220, to one of web clients 202, or to another device, e.g., for report generation. The device that receives the data, e.g., SDN controller 218, may generate and/or present a report indicative of malicious traffic to a user, e.g., via admin workstation 220.")<br><br>Chua '877 at 25:32-52 ("In the e\xample of FIG. 5, SDN controller 112 determines zones for packet flows through the network devices forming the SDN (304). The zones generally correspond to packet flows, that is, paths through the SDN followed by particular packets. SDN controller 112 may store data defining the zones in the data model discussed above. The data defining the zones may specify entities (e.g., users, devices, or the like) that have access to each zone. Thus, SDN controller 112 may program network devices of the SDN such that entities that are not authorized to access a particular zone are prevented from accessing the zone. SDN controller 112 may specify a zone using packet header field values, such as a source port, a destination port, a source IP address, a destination IP address, a virtual local area network (VLAN) tag, multiprotocol label switching (MPLS) labels, a packet protocol, and/or an IP subnet. In some cases, SDN controller 112 may specify whether a corresponding |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 796 of 1100

| No. | '111 Patent Claim 6 | Shieh '088 |
|---|---|---|
| | | packet flow for a zone is suspect or malicious and construct the zone such that packets of the packet flow are prevented from reaching an intended destination. As noted above, zones may be ordered based on priority values when overlap occurs.") |

| No. | '111 Patent Claim 7 | Shieh '088 |
|---|---|---|
| 7 | The method according to claim 5, further comprising responsive to the packet satisfying the criterion and to the instruction, sending a portion of the packet, by the network node, to the controller. | Shieh '088 discloses responsive to the packet satisfying the criterion and to the instruction, sending a portion of the packet, by the network node, to the controller.<br><br>For example, Shieh '088 discloses analyzing only certain bits in a packet to determine whether to forward packets for security processing or to allow the packets to proceed to the destination node, and thus permits only sending the portion to be analyzed to the controller. Shieh '088 discloses that its system may look for TCP FIN or TCP RST packets when applying the bypass rule, and these packets would have been identified by examining whether a FIN flag bit was set or a RST flag was set. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Shieh '088 is found to not meet this limitation, further comprising responsive to the packet satisfying the criterion and to instruction, sending a portion of the packet, by the network node, to the controller would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>*See supra* at 5.<br><br>Shieh '088 ¶ [0035] ("An embodiment of the invention also controls the communication between I/O functions and security-processing functions to enable packets to bypass security-processing function if there is no more need to inspect the packets of the connection. Some of the security functions do not need to inspect all the packets of a connection. For examples, to identify the application of a connection, there may be only need to inspect first four or five packets to make the identification. In this case, the security-processing function can notify I/O functions to bypass the security-processing function for the rest of the packets of the connections. Once the I/O function receives the notification, it will forward the packets out without redirecting the packets to the security-processing functions. This would greatly improve the performance even when security inspection is turned on."). |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 797 of 1100

| No. | '111 Patent Claim 7 | Shieh '088 |
|---|---|---|
| | | Shieh '088 ¶ [0036] "(During the bypass phase, the I/O function may notify the security-processing function if there are special events in the packet stream. These events could be receipt of TCP FIN or TCP RST packets, or not receiving any packets of the connection within a time threshold. The notification from I/O functions to security processing functions could help to clean up the state in the security-processing nodes.").<br><br>Shieh '088 ¶ [0037] "(FIG. 2B is a processing flow diagram illustrating a process of security inspection according to one embodiment of the invention. Referring to FIG. 2B, as an example, network switch 272 may represent any of network access devices 204A-204C and security device 273 may represents any of security processing devices 211A-211B as described above with respect to FIG. 2A. When device 272 receives a packet from a source node 271 via transaction 281, device 272 may determine whether the packet should be forwarded to security device 273. For example, device 272 may look up in its session table such as the one as shown in FIG. 5 to determine whether a bypass flag has been set to a predetermined value. If the bypass flag matches the predetermined value, the packet is forwarded to security device 273 via path 282; otherwise, the packet is routed to destination node 274. Alternatively, if there is no entry in the session table corresponding to the current session, the packet will also be transmitted to security device 273. After network device 272 receives a response from security device 273 via path 283, dependent upon the response, the packet may then be routed to destination node 274 via path 284. These processes may continue until a notification is received from security device 273 via path 285 indicating that it no longer wishes to receive further packets of the same session for inspection, such that subsequent packets will be directly routed to destination node 274 via path 286 without routing to security device 273. If there are certain events that have been registered from security device 273, network device 272 may notify security device 274 via path 287 upon detecting the registered events.").<br><br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Shieh '088 in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 7 of Exhibit E-4 renders the claim, including the present limitation, obvious. Below is an example. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 798 of 1100

| No. | '111 Patent Claim 7 | Shieh '088 |
|---|---|---|
| | | For example, Copeland discloses sending packets and sampled packet headers to the intrusion detection engine on the monitoring appliance based on matching predetermined values associated with a concern index.<br><br>Copeland at [0067] ("The host servers 130 are directly or indirectly coupled to one or more network devices 135 such as routers or switches that support providing a sampled data stream such as that provided by sFlow. In a typical preferred configuration for the present invention, a monitoring appli-ance 150 operating a flow-based intrusion detection engine 155 is receiving sampled packet headers from one or more network devices 135. The monitoring appliance 150 moni-tors the communications between the host server 130 and other hosts 120, 110 in the attempt to detect intrusion activity.")<br><br>Copeland [0079] ("Large packets tend to be fragmented by networks that cannot handle a large packet size. A 16-bit packet identification is used to reassemble fragmented packets. Three one-bit set of fragmentation flags control whether a packet is or may be fragmented. The 13-bit fragment offset is a sequence number for the 4-byte words in the packet when reassembled. In a series of fragments, the first offset will be zero.")<br><br>Copeland at [0097] ("The described TCP session 300 of FIG. 3 is a generic TCP session in which a network might engage. In accordance with the invention, flow data is collected about the session to help determine if the communication is abnormal. In the preferred embodiment, information such as the total number of packets sent, the total amount of data sent, the session start time and duration, and the TCP flags set in all of the packets, are collected, stored in the database 160, and analyzed to determine if the communication was suspicious. If a communication is deemed suspicious, i.e. it meets predetermined criteria, a predetermined concern index value associated with a determined category of suspicious activity is added to the cumulated CI value associated with the host that made the communication.")<br><br>Copeland at [0120] ("The sampled packet headers sent from the sFlow agent are captured and processed by the sample packet collector 505 in order to create a "Packet Data" data struc-ture that includes the sFlow agent source of the packets, the header of the sampled packets, and other information avail-able from the sFlow data stream that may be important. For example, one data field that is optionally available pr vides the username of the user using the computer at the time of the communications. This information is extremely useful in some |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 799 of 1100

| No. | '111 Patent Claim 7 | Shieh '088 |
|---|---|---|
| | | environments subject to regulatory requirements and monitoring of the communications on the network. In this case the username will be stored as "supplementary infor-mation" for auditing purposes in the flow data. Other infor-mation, including the sampling device and the physical port on which the communications was detected may also be retained for other uses such as mitigation, where a host may be removed from the network.")<br><br>Copeland at [0126]-[0129] ("If a particular packet 101 being processed by the packet classifier 510 matches a particular entry or record in the flow data structure 162, data from that particular packet 101 is used to update the statistics in the corresponding flow data structure record. A packet 101 is considered to match to a flow data structure record if both IP numbers match and the source of the sampled packet matches and:<br><br>(1) both port numbers match and no port is marked as the "server" port, or<br>(2) the port number previously marked as the "server" port matches, or<br>(3) one of the port numbers matches, but the other does not, and the neither port number has been marked as the server port (in this case the matching port number is marked as the "server" port).")<br><br>Copeland at [0144] ("Concern index (CI) values calculated from packet anomalies also add to a host's accumulated concern index value. Table II of FIG. 7 shows one scheme for assigning concern index values due to other events revealed by the flow analysis. For example, there are many combinations of TCP flag bits that are rarely or never seen in valid TCP connections. When the packet classifier thread 510 recog-nizes one of these combinations, it directly adds a predeter-mined value to the sending host's accumulated concern index value. When the packet classifier thread 510 searches along the flow linked-list (i.e. flow data 162) for a match to the current packet 101, it keeps count of the number of flows active with matching IP addresses but no matching port number. If this number exceeds a predetermined threshold value (e.g., 4) and is greater than the previous number noticed, CI is added for an amount corresponding to a "port scan." A bit in the host record is set to indicate that the host has received CI for "port scanning."")<br><br>Copeland at [0150] ("A preferred hardware configuration 800 of an embodiment that executes the functions of the above-described flow-based engine is described in reference to FIG. 8. FIG. 8 illustrates a typically hardware configuration 800 for a network intrusion detection |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 800 of 1100

| No. | '111 Patent Claim 7 | Shieh '088 |
|---|---|---|
| | | system. A monitoring appliance 150 serves as a pass-by filter of network traffic. A network device 135, such as a router or switch supporting sFlow provides the location for connecting the monitoring appliance 150 to the network 899 for monitoring the network traffic.")<br><br>Copeland at [0159]-[0162] ("A packet 101 is considered to match to a flow data structure record if both IP numbers match and the source of the sampled data matches and:<br><br>(a). both port numbers match and no port is marked as the "server" port, or<br>(b). the port number previously marked as the "server" port matches, or<br>(c). one of the port numbers matches, but the other does not, and the neither port number has been marked as the server port (in this case the matching port number is marked as the "server" port).") |

| No. | '111 Patent Claim 8 | Shieh '088 |
|---|---|---|
| 8[a] | The method according to claim 7, wherein the portion of the packet consists of multiple consecutive bytes, and | Shieh '088 discloses wherein the portion of the packet consists of multiple consecutive bytes.<br><br>For example, Shieh '088 discloses analyzing only certain bits in a packet to determine whether to forward packets for security processing or to allow the packets to proceed to the destination node, and thus permits only sending the portion to be analyzed to the controller. Shieh '088 discloses that its system may look for TCP FIN or TCP RST packets when applying the bypass rule, and these packets would have been identified by examining whether a FIN flag bit was set or a RST flag was set. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Shieh '088 is found to not meet this limitation, further comprising responsive to the packet satisfying the criterion and to instruction, sending a portion of the packet, by the network node, to the controller would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>*See supra* at 7.<br><br>Shieh '088 ¶ [0035] ("An embodiment of the invention also controls the communication between I/O functions and security-processing functions to enable packets to bypass security |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 801 of 1100

| No. | '111 Patent Claim 8 | Shieh '088 |
|---|---|---|
| | | processing function if there is no more need to inspect the packets of the connection. Some of the security functions do not need to inspect all the packets of a connection. For examples, to identify the application of a connection, there may be only need to inspect first four or five packets to make the identification. In this case, the security-processing function can notify I/O functions to bypass the security-processing function for the rest of the packets of the connections. Once the I/O function receives the notification, it will forward the packets out without redirecting the packets to the security-processing functions. This would greatly improve the performance even when security inspection is turned on.").<br><br>Shieh '088 ¶ [0036] "(During the bypass phase, the I/O function may notify the security-processing function if there are special events in the packet stream. These events could be receipt of TCP FIN or TCP RST packets, or not receiving any packets of the connection within a time threshold. The notification from I/O functions to security processing functions could help to clean up the state in the security-processing nodes.").<br><br>Shieh '088 ¶ [0037] "(FIG. 2B is a processing flow diagram illustrating a process of security inspection according to one embodiment of the invention. Referring to FIG. 2B, as an example, network switch 272 may represent any of network access devices 204A-204C and security device 273 may represents any of security processing devices 211A-211B as described above with respect to FIG. 2A. When device 272 receives a packet from a source node 271 via transaction 281, device 272 may determine whether the packet should be forwarded to security device 273. For example, device 272 may look up in its session table such as the one as shown in FIG. 5 to determine whether a bypass flag has been set to a predetermined value. If the bypass flag matches the predetermined value, the packet is forwarded to security device 273 via path 282; otherwise, the packet is routed to destination node 274. Alternatively, if there is no entry in the session table corresponding to the current session, the packet will also be transmitted to security device 273. After network device 272 receives a response from security device 273 via path 283, dependent upon the response, the packet may then be routed to destination node 274 via path 284. These processes may continue until a notification is received from security device 273 via path 285 indicating that it no longer wishes to receive further packets of the same session for inspection, such that subsequent packets will be directly routed to destination node 274 via path 286 without routing to security device 273. If there are certain events that have been registered from |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 802 of 1100

| No. | '111 Patent Claim 8 | Shieh '088 |
|---|---|---|
| | | security device 273, network device 272 may notify security device 274 via path 287 upon detecting the registered events.").<br><br>Shieh '088 at [0060] ("Some portions of the preceding detailed descriptions have been presented in terms of algorithms and symbolic representations of operations on data bits within a computer memory. These algorithmic descriptions and representations are the ways used by those skilled in the data processing arts to most effectively convey the Substance of their work to others skilled in the art. An algorithm is here, and generally, conceived to be a self-consistent sequence of operations leading to a desired result. The operations are those requiring physical manipulations of physical quantities.")<br><br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Shieh '088 in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 8(a) of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references.<br><br>For example, Kempf discloses consecutive bytes of a packet header field.<br><br>Kempf at [0081] ("In one embodiment, OpenFlow is modified to pro-vide rules for GTP TEID Routing. FIG. 17 is a diagram of one embodiment of the OpenFlow flow table modification for GTP TEID routing. An OpenFlow switch that supports TEID routing matches on the 2 byte (16 bit) collection of header fields and the 4 byte (32 bit) GTP TEID, in addition to other OpenFlow header fields, in at least one flow table ( e.g., the first flow table). The GTP TEID flag can be wildcarded (i.e. matches are "don't care"). In one embodiment, the EPC pro-tocols do not assign any meaning to TEIDs other than as an endpoint identifier for tunnels, like ports in standard UDP/ TCP transport protocols. In other embodiments, the TEIDs can have a correlated meaning or semantics. The GTP header flags field can also be wildcarded, this can be partially matched by combining the following bitmasks: 0xFF00- Match the Message Type field; 0xe0-Match the Version field; 0xl0-Match the PT field; 0x04-Match the E field; 0x02-Match the S field; and 0x0l-Match the PN field.")<br><br>Kempf at [0083] ("If a packet either needs encapsulation or arrives encapsulated with nonzero header flags, header extensions, and/or the GTP-U packet is not a G-PDU packet (i.e. it is a |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 803 of 1100

| No. | ’111 Patent Claim 8 | Shieh ’088 |
|---|---|---|
| | | GTP-U control packet), the processing must proceed via the gateway's slow path (software) control plane. GTP-C and GTP' packets directed to the gateway's IP address are a result of mis-configuration and are in error. They must be sent to the OpenFlow controller, since these packets are handled by the S-GW-C and P-GW-C control plane entities in the cloud computing system or to the billing entity handling GTP' and not the S-GW-D and P-GW-D data plane switches.”)<br><br>Kempf at [0087] (“In one embodiment, slow path support for GTP is implemented with an OpenFlow gateway switch. An Open-Flow mobile gateway switch also contains support on the software control plane for slow path packet processing. This path is taken by G-PDU (message type 255) packets with nonzero header fields or extension headers, and user data plane packets requiring encapsulation with such fields or addition of extension headers, and by G TP-U control packets. For this purpose, the switch supports three local ports in the software control plane: LOCAL_GTP _CONTROL-the switch fast path forwards GTP encapsulated packets directed to the gateway IP address that contain GTP-U control mes-sages and the local switch software control plane initiates local control plane actions depending on the GTP-U control message; LOCAL_GTP _U_DECAP-the switch fast path forwards G-PDU packets to this port that have nonzero header fields or extension headers (i.e. E!=0, S!=0, or PN!=0). These packets require specialized handling. The local switch software slow path processes the packets and performs the specialized handling; and LOCAL_GTP _U_ENCAP-the switch fast path forwards user data plane packets to this port that require encapsulation in a GTP tunnel with nonzero header fields or extension headers (i.e. E!=0, S!=0, or PN!=0). These packets require specialized handling. The local switch software slow path encapsulates the packets and performs the specialized handling. In addition to forwarding the packet, the switch fast path makes the OpenFlow metadata field avail-able to the slow path software.”)<br><br>Kempf at [0091] (“When a packet header matches a rule associated with the virtual port, the GTP TEID is written into the lower 32 bits of the metadata and the packet is directed to the virtual port. The virtual port calculates the hash of the TEID and looks up the tunnel header information in the tunnel header table. If no such tunnel information is present, the packet is forwarded to the controller with an error indication. Other-wise, the virtual port constructs a GTP tunnel header and encapsulates the packet. Any DSCP bits or VLAN priority bits are additionally set in the IP or MAC tunnel headers, and any VLAN tags or MPLS labels are |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 804 of 1100

| No. | '111 Patent Claim 8 | Shieh '088 |
|---|---|---|
| | | pushed onto the packet. The encapsulated packet is forwarded out the physical port to which the virtual port is bound.")<br><br>Kempf at [0092] ("In one embodiment, the system implements a GTP fast path decapsulation virtual port. When requested by the S-GW and P-GW control plane software running in the cloud computing system, the gateway switch installs rules and actions for routing GTP encapsulated packets out of GTP tunnels. The rules match the GTP header flags and the GTP TEID for the packet, in the modified OpenFlow flow table shown in FIG. 17 as follows: the IP destination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); the UDP destination port is the GTP-U destination port (2152); and the header fields and message type field is wildcarded with the flag 0XFFF0 and the upper two bytes of the field match the G-PDU message type (255) while the lower two bytes match 0x30, i.e. the packet is a GTP packet not a GTP' packet and the version number is 1.")<br><br>Kempf at [0098] ("The header flags and message type fields for the three rules are wildcarded with the following bitmasks and match as follows: bitmask 0xFFF4 and the upper two bytes match the G-PDU message type (255) while the lower two bytes are Ox34, indicating that the version number is 1, the packet is a GTP packet, and there is an extension header present; bitmask 0xFFFF2 and the upper two bytes match the G-PDU message type (255) while the lower two bytes are 0x32, indicating that the version number is 1, the packet is a GTP packet, and there is a sequence number bitmask 0xFF0l and the upper two bytes match the G-PDU message type (255) while the lower two bytes are 0x31, indicating that the version number is 1, the packet is a GTP packet, and a N-PDU is present.")<br><br>Kempf at [0101] ("In one embodiment, the system implements han-dling of user data plane packets requiring GTP-U encapsula-tion with extension headers, sequence numbers, and N-PDU numbers. User data plane packets that require extension head-ers, sequence numbers, or N-PDU numbers during GTP encapsulation require special handling by the software slow path. For these packets, the OpenFlow controller programs a rule matching the 4 tuple: IP source address; IP destination address; UDP/TCP/SCTP source port; and UDP/TCP/SCTP destination port. The instructions for matching packets are:<br><br>Write-Metadata ( GTP-TEID, 0x FFFFFFFF)<br>Apply-Actions (Set-Output-Port LOCAL_GTP_U_ENCAP )") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 805 of 1100

| No. | '111 Patent Claim 8 | Shieh '088 |
|---|---|---|
| | | For example, Copeland discloses fragmenting packets into smaller byte sizes, including headers and flags. Copeland further discloses sending sampled packet headers, consisting of fragmented packets of consecutive bytes to the monitoring device.<br><br>Copeland [0079] ("Large packets tend to be fragmented by networks that cannot handle a large packet size. A 16-bit packet identification is used to reassemble fragmented packets. Three one-bit set of fragmentation flags control whether a packet is or may be fragmented. The 13-bit fragment offset is a sequence number for the 4-byte words in the packet when reassembled. In a series of fragments, the first offset will be zero.") |
| 8[b] | wherein the instruction comprises identification of the consecutive bytes in the packet. | For example, Shieh '088 discloses analyzing only certain bits in a packet to determine whether to forward packets for security processing or to allow the packets to proceed to the destination node, and thus permits only sending the portion to be analyzed to the controller. Shieh '088 discloses that its system may look for TCP FIN or TCP RST packets when applying the bypass rule, and these packets would have been identified by examining whether a FIN flag bit was set or a RST flag was set. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Shieh '088 is found to not meet this limitation, further comprising responsive to the packet satisfying the criterion and to instruction, sending a portion of the packet, by the network node, to the controller would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>*See supra* at 7.<br><br>Shieh '088 ¶ [0035] ("An embodiment of the invention also controls the communication between I/O functions and security-processing functions to enable packets to bypass security-processing function if there is no more need to inspect the packets of the connection. Some of the security functions do not need to inspect all the packets of a connection. For examples, to identify the application of a connection, there may be only need to inspect first four or five packets to make the identification. In this case, the security-processing function can notify I/O functions to bypass the security-processing function for the rest of the packets of the connections. Once the I/O function receives the notification, it will forward the packets out without redirecting the packets to the security-processing functions. This would greatly improve the performance even when security inspection is turned on."). |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 806 of 1100

| No. | '111 Patent Claim 8 | Shieh '088 |
|---|---|---|
| | | Shieh '088 ¶ [0036] "(During the bypass phase, the I/O function may notify the security-processing function if there are special events in the packet stream. These events could be receipt of TCP FIN or TCP RST packets, or not receiving any packets of the connection within a time threshold. The notification from I/O functions to security processing functions could help to clean up the state in the security-processing nodes.").<br><br>Shieh '088 ¶ [0037] "(FIG. 2B is a processing flow diagram illustrating a process of security inspection according to one embodiment of the invention. Referring to FIG. 2B, as an example, network switch 272 may represent any of network access devices 204A-204C and security device 273 may represents any of security processing devices 211A-211B as described above with respect to FIG. 2A. When device 272 receives a packet from a source node 271 via transaction 281, device 272 may determine whether the packet should be forwarded to security device 273. For example, device 272 may look up in its session table such as the one as shown in FIG. 5 to determine whether a bypass flag has been set to a predetermined value. If the bypass flag matches the predetermined value, the packet is forwarded to security device 273 via path 282; otherwise, the packet is routed to destination node 274. Alternatively, if there is no entry in the session table corresponding to the current session, the packet will also be transmitted to security device 273. After network device 272 receives a response from security device 273 via path 283, dependent upon the response, the packet may then be routed to destination node 274 via path 284. These processes may continue until a notification is received from security device 273 via path 285 indicating that it no longer wishes to receive further packets of the same session for inspection, such that subsequent packets will be directly routed to destination node 274 via path 286 without routing to security device 273. If there are certain events that have been registered from security device 273, network device 272 may notify security device 274 via path 287 upon detecting the registered events.").<br><br>Shieh '088 at [0060] ("Some portions of the preceding detailed descriptions have been presented in terms of algorithms and symbolic representations of operations on data bits within a computer memory. These algorithmic descriptions and representations are the ways used by those skilled in the data processing arts to most effectively convey the Substance of their work to others skilled in the art. An algorithm is here, and generally, conceived to be a |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 807 of 1100

| No. | '111 Patent Claim 8 | Shieh '088 |
|---|---|---|
| | | self-consistent sequence of operations leading to a desired result. The operations are those requiring physical manipulations of physical quantities.")<br><br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Shieh '088 in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 8(b) of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references.<br><br>For example, Kempf discloses rules in which the flow table includes matching to the consecutive bytes of a packet header.<br><br>Kempf at [0081] ("In one embodiment, OpenFlow is modified to pro-vide rules for GTP TEID Routing. FIG. 17 is a diagram of one embodiment of the OpenFlow flow table modification for GTP TEID routing. An OpenFlow switch that supports TEID routing matches on the 2 byte (16 bit) collection of header fields and the 4 byte (32 bit) GTP TEID, in addition to other OpenFlow header fields, in at least one flow table ( e.g., the first flow table). The GTP TEID flag can be wildcarded (i.e. matches are "don't care"). In one embodiment, the EPC pro-tocols do not assign any meaning to TEIDs other than as an endpoint identifier for tunnels, like ports in standard UDP/ TCP transport protocols. In other embodiments, the TEIDs can have a correlated meaning or semantics. The GTP header flags field can also be wildcarded, this can be partially matched by combining the following bitmasks: 0xFF00- Match the Message Type field; 0xe0-Match the Version field; 0xl0-Match the PT field; 0x04-Match the E field; 0x02-Match the S field; and 0x0l-Match the PN field.")<br><br>Kempf at [0083] ("If a packet either needs encapsulation or arrives encapsulated with nonzero header flags, header extensions, and/or the GTP-U packet is not a G-PDU packet (i.e. it is a GTP-U control packet), the processing must proceed via the gateway's slow path (software) control plane. GTP-C and GTP' packets directed to the gateway's IP address are a result of mis-configuration and are in error. They must be sent to the OpenFlow controller, since these packets are handled by the S-GW-C and P-GW-C control plane entities in the cloud computing system or to the billing entity handling GTP' and not the S-GW-D and P-GW-D data plane switches.") |


~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 808 of 1100

| No. | '111 Patent Claim 8 | Shieh '088 |
|---|---|---|
| | | Kempf at [0087] ("In one embodiment, slow path support for GTP is implemented with an OpenFlow gateway switch. An Open-Flow mobile gateway switch also contains support on the software control plane for slow path packet processing. This path is taken by G-PDU (message type 255) packets with nonzero header fields or extension headers, and user data plane packets requiring encapsulation with such fields or addition of extension headers, and by G TP-U control packets. For this purpose, the switch supports three local ports in the software control plane: LOCAL_GTP _CONTROL-the switch fast path forwards GTP encapsulated packets directed to the gateway IP address that contain GTP-U control mes-sages and the local switch software control plane initiates local control plane actions depending on the GTP-U control message; LOCAL_GTP _U_DECAP-the switch fast path forwards G-PDU packets to this port that have nonzero header fields or extension headers (i.e. E!=0, S!=0, or PN!=0). These packets require specialized handling. The local switch software slow path processes the packets and performs the specialized handling; and LOCAL_GTP _U_ENCAP-the switch fast path forwards user data plane packets to this port that require encapsulation in a GTP tunnel with nonzero header fields or extension headers (i.e. E!=0, S!=0, or PN!=0). These packets require specialized handling. The local switch software slow path encapsulates the packets and performs the specialized handling. In addition to forwarding the packet, the switch fast path makes the OpenFlow metadata field avail-able to the slow path software.")<br><br>Kempf at [0091] ("When a packet header matches a rule associated with the virtual port, the GTP TEID is written into the lower 32 bits of the metadata and the packet is directed to the virtual port. The virtual port calculates the hash of the TEID and looks up the tunnel header information in the tunnel header table. If no such tunnel information is present, the packet is forwarded to the controller with an error indication. Other-wise, the virtual port constructs a GTP tunnel header and encapsulates the packet. Any DSCP bits or VLAN priority bits are additionally set in the IP or MAC tunnel headers, and any VLAN tags or MPLS labels are pushed onto the packet. The encapsulated packet is forwarded out the physical port to which the virtual port is bound.")<br><br>Kempf at [0092] ("In one embodiment, the system implements a GTP fast path decapsulation virtual port. When requested by the S-GW and P-GW control plane software running in the cloud computing system, the gateway switch installs rules and actions for routing GTP encapsulated packets out of GTP tunnels. The rules match the GTP header flags and the GTP |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 809 of 1100

| No. | '111 Patent Claim 8 | Shieh '088 |
|---|---|---|
| | | TEID for the packet, in the modified OpenFlow flow table shown in FIG. 17 as follows: the IP destination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); the UDP destination port is the GTP-U destination port (2152); and the header fields and message type field is wildcarded with the flag 0XFFF0 and the upper two bytes of the field match the G-PDU message type (255) while the lower two bytes match 0x30, i.e. the packet is a GTP packet not a GTP' packet and the version number is 1.")<br><br>Kempf at [0098] ("The header flags and message type fields for the three rules are wildcarded with the following bitmasks and match as follows: bitmask 0xFFF4 and the upper two bytes match the G-PDU message type (255) while the lower two bytes are Ox34, indicating that the version number is 1, the packet is a GTP packet, and there is an extension header present; bitmask 0xFFFF2 and the upper two bytes match the G-PDU message type (255) while the lower two bytes are 0x32, indicating that the version number is 1, the packet is a GTP packet, and there is a sequence number bitmask 0xFF0l and the upper two bytes match the G-PDU message type (255) while the lower two bytes are 0x31, indicating that the version number is 1, the packet is a GTP packet, and a N-PDU is present.")<br><br>Kempf at [0101] ("In one embodiment, the system implements han-dling of user data plane packets requiring GTP-U encapsula-tion with extension headers, sequence numbers, and N-PDU numbers. User data plane packets that require extension head-ers, sequence numbers, or N-PDU numbers during GTP encapsulation require special handling by the software slow path. For these packets, the OpenFlow controller programs a rule matching the 4 tuple: IP source address; IP destination address; UDP/TCP/SCTP source port; and UDP/TCP/SCTP destination port. The instructions for matching packets are:<br><br>Write-Metadata ( GTP-TEID, 0x FFFFFFFF)<br>Apply-Actions (Set-Output-Port LOCAL_GTP _U_ENCAP )")<br><br>For example, Copeland discloses identifying the sampled packet headers comprised of fragmented packets of smaller byte sizes.<br><br>Copeland [0079] ("Large packets tend to be fragmented by networks that cannot handle a large packet size. A 16-bit packet identification is used to reassemble fragmented packets. Three one-bit set of fragmentation flags control whether a packet is or may be fragmented. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 810 of 1100

| No. | '111 Patent Claim 8 | Shieh '088 |
|---|---|---|
| | | The 13-bit fragment offset is a sequence number for the 4-byte words in the packet when reassembled. In a series of fragments, the first offset will be zero.")<br><br>Copeland at [0080] ("After the fragmentation information, an 8-bit time to live field specifies the remaining life of a packet and is decremented each time the packet is relayed. If this field is 0, the packet is destroyed. Next is an 8-bit protocol field that specifies the transport protocol used in the data portion. The following 16-bit field is a header checksum on the header only. Finally, the last two fields illustrated contain the 32-bit source address and 32-bit destination address. IP packet data follows the address information.")<br><br>Copeland at [0081] ("In a TCP/IP datagram 210, the initial data of the IP datagram is the TCP header 230 information. The initial TCP header 230 information includes the 16-bit source and 16-bit destination port numbers. A 32-bit sequence number for the data in the packet follows the port numbers. Following the sequence number is a 32-bit acknowledgement number. If an ACK flag (discussed below) is set, this number is the next sequence number the sender of the packet expects to receive. Next is a 4-bit data offset, which is the number of 32-bit words in the TCP header. A 6-bit reserved field follows.") |

| No. | '111 Patent Claim 9 | Shieh '088 |
|---|---|---|
| 9 | The method according to claim 5, further comprising responsive to receiving the packet, analyzing the packet, by the controller. | Shieh '088 discloses the method according to claim 5, further comprising responsive to receiving the packet, analyzing the packet, by the controller.<br><br>For example, Shieh '088 discloses that when the central controller receives the packet, it decides which of the security processing modules is able to process the packets, and then forwards the packets to the designated security processing module.<br><br>Shieh '088 ¶ [0028] ("Firewall modules 209A-209C may be part of a distributed firewall described above. For example, firewall modules 209A-209C may be the IO functions of a firewall while nodes 211A-211B may be firewall processing nodes. That is, modules 211A-211B may be dedicated firewall processing devices that perform some firewall processing operations such as DPI, content inspection, antivirus, etc., while firewall modules 209A-209C are responsible for routing data packets. For example, when firewall module 209B receives a packet from node 206, it may forward the packet to firewall processing node 211A for content |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 811 of 1100

| No. | '111 Patent Claim 9 | Shieh '088 |
|---|---|---|
| | | inspection and/or forwards the packet to controller 208 for routing information. In response, firewall processing node 211A analyzes the received packet and/or further communicates with controller 208. Controller 208 may provide further routing information back to network access device 204B regarding how to route the packet. Each of the firewall processing nodes 211A-211B may further maintains a persistent connection or tunnel with controller 208, for example, using the OpenFlow communication protocol.").<br><br>Shieh '088 ¶ [0042] ("In one embodiment, central controller 208 is the central place to control forwarding of the packets amongst I/O modules 301-304, security processing modules 309-311, and service processing modules 312-313. When a virtual I/O module receives a packet, according to one embodiment, it forwards the packet to central controller 208 if it cannot find an existing connection in its local cache, as shown in FIG. 5. When central controller 208 receives the packet, it decides which of security processing modules 309-311 is able to process the packets, and then forwards the packets to the designated security processing module. It also instructs the virtual I/O module to create the local cache to store connection state information so the subsequent packets of the same connection session do not need to be forwarded to central controller 208; rather, they can be directly forwarded to the proper security processing module identified in the cache."). |

| No. | '111 Patent Claim 12 | Shieh '088 |
|---|---|---|
| 12 | The method according to claim 9, wherein the analyzing comprises applying security or data analytic application. | Shieh '088 discloses the method according to claim 9, wherein the analyzing comprises applying security or data analytic application.<br><br>For example, Shieh '088 discloses that the controller analyzes a received packet, in which the analysis performed on the packets includes applying a security application.<br><br>Shieh '088 ¶ [0002] ("Embodiments of the present invention relate generally to network security. More particularly, embodiments of the invention relate to enabling network security with network equipment.").<br><br>Shieh '088 ¶ [0017] ("According to some embodiments, a mechanism is utilized to dynamically perform security inspection in a network. In one embodiment, the mechanism includes two functions: 1) an input/output (IO) function that performs the distribution of network traffic; and 2) a security-processing function that performs security processing |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 812 of 1100

| No. | '111 Patent Claim 12 | Shieh '088 |
|---|---|---|
| | | including security inspection and policy enforcement. The IO function receives the packets and uses a session table to forward the packets to the security-processing function. A session table is a data structure that stores connection states, including the destination of the security-processing function. In one embodiment, the IO function determines, based on an internal data structure such as a session or flow table, whether the packet should be forwarded to the security processing function for security inspection. The configuration of the IO function to control whether to forward the packets to the security processing function can be set based on a command received from an administrator or alternatively, based on a signal received from the security processing function.").<br><br>Shieh '088 ¶ [0018] ("According to one embodiment, an administrator can configure, for example, via a controller or a management entity, a network access device to set up a set of filtering rules specifying whether and/or what types of packets should be forwarded to a security device and which of the security devices for security inspection. In this embodiment, the controller is configured to manage multiple network access devices and/or multiple security devices. Alternatively, a security device may inform a network access device that subsequent packets of a particular session should be forwarded from the network access device for security inspection. In one embodiment, a security device performs the security inspection at the beginning of the flow or session, and at a certain point, the security device decides that it no longer needs to inspect further packets of the same session.").<br><br>Shieh '088 ¶ [0019] ("Advantages of embodiments of the present invention include, without limitation, providing a way to integrate partial network security functions into other network equipment, such as switches or routers. The integration allows network administrators to turn on security inspection functionality when there are needs for such, thus one can flexibly perform security inspection if needed. The notification between I/O functions and security-processing functions can reduce the number of packets to be inspected, thus enhancing the performance without lax the network security.").<br><br>Shieh '088 ¶ [0021] ("According to one embodiment, network access device 204 is associated with a distributed firewall 212 that includes various firewall processing modules, for example, each being executed within a virtual machine (VM). In one embodiment, each firewall module is responsible for performing one or more firewall functions, but it does not include all of the firewall functions of a firewall. Examples of the firewall functions include, |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 813 of 1100

| No. | '111 Patent Claim 12 | Shieh '088 |
|---|---|---|
| | | but are not limited to, network address translation (NAT), virtual private network (VPN), deep packet inspection (DPI), and/or anti-virus, etc. In one embodiment, some of the firewall processing modules are located within network access device 204 (e.g., firewall modules 209) and some are located external to network access device 204 (e.g., firewall modules 210 maintained by firewall processing node(s) 211, which may be a dedicated firewall processing machine. All of the firewall modules 209-210 are managed by firewall controller 208, which may be located within network access device 204, or external to network access device 204, such as, for example, in a public cloud associated with network 203, or in a private cloud associated with network 205. Controller 208 and firewall processing modules 209-210 collectively are referred to herein as distributed firewall 212.").<br><br>Shieh '088 ¶ [0023] ("According to one embodiment, a mechanism is utilized to dynamically perform security inspection in a network. In one embodiment, the mechanism includes two functions: 1) an input/output (IO) function (e.g., firewall module(s) 209) that performs the distribution of network traffic; and 2) a security-processing function (e.g., firewall module(s) 210) that performs security processing, including security inspection and policy enforcement. IO function 209 receives the packets and uses a session table to forward the packets to security-processing function 210. A session table is a data structure that stores connection states, including the destination of security-processing function. In one embodiment, IO function 209 determines, based on an internal data structure such as a session or flow table (e.g., session table as shown in FIG. 5), whether the packet should be forwarded to security processing function 210 for security inspection. The configuration of IO function 209 to control whether to forward the packets to security processing function 210 can be set based on a command received from an administrator or alternatively, based on a signal received from security processing function 210.").<br><br>Shieh '088 ¶ [0029] ("According to one embodiment, an administrator 265 configures, for example, via a controller or a management entity 208, a network access device (e.g., network access devices 204A-204C) to set up a set of filtering rules concerning whether and/or what types of packets should be forwarded to a security device and which of the security devices (e.g., security devices 211A-211B) for security inspection. In this embodiment, controller 208 is configured to manage multiple network access devices 204A-204C and/or multiple security devices 211A-211B. Alternatively, a security device, such as security device 211A, may inform a network access device, such as network access device 204B, |

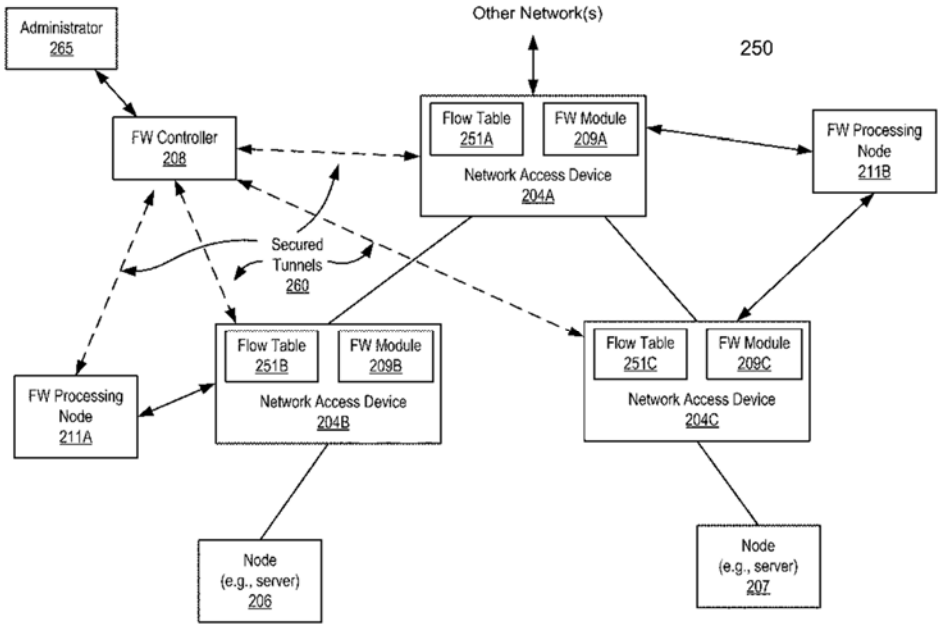


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 814 of 1100

| No. | '111 Patent Claim 12 | Shieh '088 |
|---|---|---|
| | | whether subsequent packets of a particular session should be forwarded from the network access device for security inspection. A security device may perform the security inspection on packets at the beginning of the flow or session, and at a certain point, the security device decides that it no longer needs to inspect further packets of the same session.").<br><br>Shieh '088 ¶ [0030] ("The configuration information may be stored in a memory or storage device of a network access device. In one embodiment, such configuration information may be stored as part of a flow table or session table as shown in FIG. 5. Referring to FIG. 5, a bypass flag 501 may be received from a security device indicating that the security device no longer wishes to receive further packets of the same session for security inspection. In addition, a security device may register certain notification events 502 with a network access device, such that when the network access device detects such events, it will notify the security device. Further, a set of one or more filtering rules 503 may be received from an administrator to filter and send only certain types of packets to a security device for inspection.").<br><br>Shieh '088 ¶ [0031] ("According to one embodiment, referring back to FIG. 2A, when a security-processing function (e.g., processing node 211A) receives the packets, it does the security inspection and security policy enforcement. The packets then are forwarded to the next I/O function (e.g., modules 209A-209C). The choices of the next I/O function could be from the decision from layer 2 such as Ethernet MAC address lookup, or IP address routing, or other methods."). |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 815 of 1100

| No. | '111 Patent Claim 12 | Shieh '088 |
|---|---|---|
| | |  <br>FIG. 2A<br><br>Fig. 2A<br><br>Shieh '088 ¶ [0035] ("An embodiment of the invention also controls the communication between I/O functions and security-processing functions to enable packets to bypass security-processing function if there is no more need to inspect the packets of the connection. Some of the security functions do not need to inspect all the packets of a connection. For examples, to identify the application of a connection, there may be only need to inspect first four or five packets to make the identification. In this case, the security-processing function can notify I/O functions to bypass the security-processing function for the rest of the packets of the connections. Once the I/O function receives the notification, it will forward the packets out without redirecting the packets to the security-processing functions. This would greatly improve the performance even when security inspection is turned on."). |

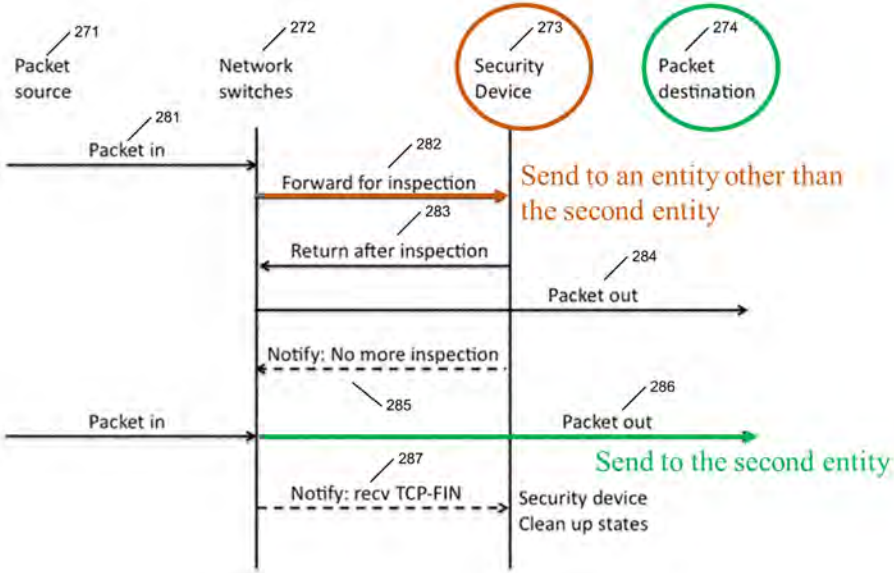


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 816 of 1100

| No. | '111 Patent Claim 12 | Shieh '088 |
|---|---|---|
| | | Shieh '088 ¶ [0036] ("During the bypass phase, the I/O function may notify the security-processing function if there are special events in the packet stream. These events could be receipt of TCP FIN or TCP RST packets, or not receiving any packets of the connection within a time threshold. The notification from I/O functions to security processing functions could help to clean up the state in the security-processing nodes.").<br><br>Shieh '088 ¶ [0037] ("FIG. 2B is a processing flow diagram illustrating a process of security inspection according to one embodiment of the invention. Referring to FIG. 2B, as an example, network switch 272 may represent any of network access devices 204A-204C and security device 273 may represents any of security processing devices 211A-211B as described above with respect to FIG. 2A. When device 272 receives a packet from a source node 271 via transaction 281, device 272 may determine whether the packet should be forwarded to security device 273. For example, device 272 may look up in its session table such as the one as shown in FIG. 5 to determine whether a bypass flag has been set to a predetermined value. If the bypass flag matches the predetermined value, the packet is forwarded to security device 273 via path 282; otherwise, the packet is routed to destination node 274. Alternatively, if there is no entry in the session table corresponding to the current session, the packet will also be transmitted to security device 273. After network device 272 receives a response from security device 273 via path 283, dependent upon the response, the packet may then be routed to destination node 274 via path 284. These processes may continue until a notification is received from security device 273 via path 285 indicating that it no longer wishes to receive further packets of the same session for inspection, such that subsequent packets will be directly routed to destination node 274 via path 286 without routing to security device 273. If there are certain events that have been registered from security device 273, network device 272 may notify security device 274 via path 287 upon detecting the registered events."). |

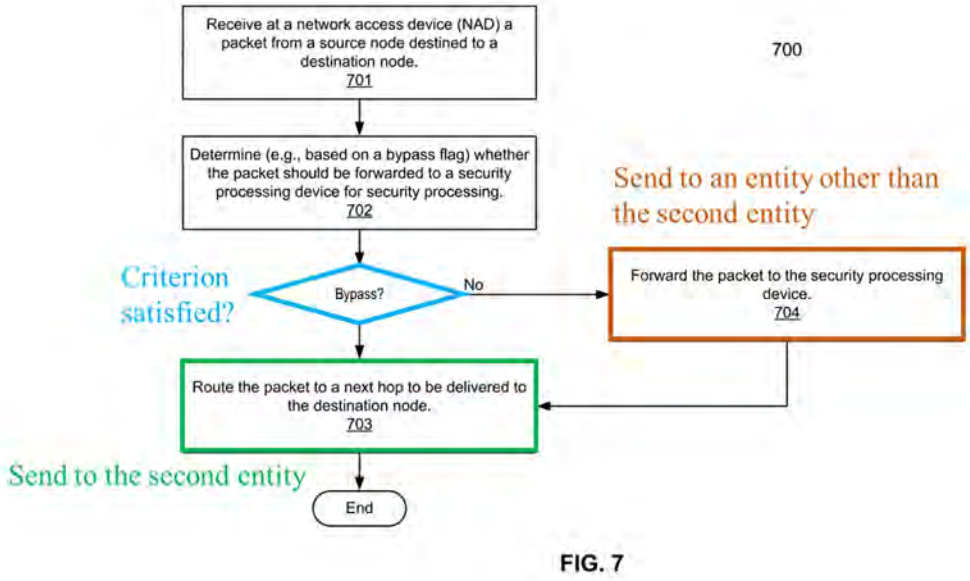


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 817 of 1100

| No. | '111 Patent Claim 12 | Shieh '088 |
|---|---|---|
| | |  <br>FIG. 2B<br><br>Fig. 2B (annotation added)<br><br>Shieh '088 ¶ [0042] ("In one embodiment, central controller 208 is the central place to control forwarding of the packets amongst I/O modules 301-304, security processing modules 309-311, and service processing modules 312-313. When a virtual I/O module receives a packet, according to one embodiment, it forwards the packet to central controller 208 if it cannot find an existing connection in its local cache, as shown in FIG. 5. When central controller 208 receives the packet, it decides which of security processing modules 309-311 is able to process the packets, and then forwards the packets to the designated security processing module. It also instructs the virtual I/O module to create the local cache to store connection state information so the subsequent packets of the same connection session do not need to be forwarded to central controller 208; rather, they can be directly forwarded to the proper security processing module identified in the cache."). |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 818 of 1100

| No. | '111 Patent Claim 12 | Shieh '088 |
|---|---|---|
| | | Shieh '088 ¶ [0049] ("FIG. 7 is a flow diagram illustrating a method for performing firewall operations using a distributed firewall according to one embodiment of the invention. Method 700 may be performed by processing logic that may include software, hardware, or a combination of both. For example, method 700 may be performed by distributed firewall 212 of FIG. 1. Referring to FIG. 7, at block 701, a network access device receives a packet from a source node destined to a destination node. At block 702, the network access device determines whether the packet should be forwarded to a security device for security inspection. For example, processing logic may check whether there is an entry exists in a session table for the current session. If not, it may forward the packet to the security device for security processing at block 704. Alternatively, the processing logic may check whether there is a bypass flag set to a predetermined value for the current session. If there is, the packet will not be forwarded to the security device; instead, the packet will be directly routed to the destination node at block 703.").   Fig. 7 (annotation added) |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 819 of 1100

| No. | '111 Patent Claim 13 | Shieh '088 |
|---|---|---|
| 13 | The method according to claim 9, wherein the analyzing comprises applying security application that comprises firewall or intrusion detection functionality. | Shieh '088 discloses the method according to claim 9, wherein the analyzing comprises applying security application that comprises firewall or intrusion detection functionality.<br><br>For example, Shieh ''088 discloses that its system analyzes packets by applying a security application that comprises firewall functionality.<br><br>Shieh '088 ¶ [0021] ("According to one embodiment, network access device 204 is associated with a distributed firewall 212 that includes various firewall processing modules, for example, each being executed within a virtual machine (VM). In one embodiment, each firewall module is responsible for performing one or more firewall functions, but it does not include all of the firewall functions of a firewall. Examples of the firewall functions include, but are not limited to, network address translation (NAT), virtual private network (VPN), deep packet inspection (DPI), and/or anti-virus, etc. In one embodiment, some of the firewall processing modules are located within network access device 204 (e.g., firewall modules 209) and some are located external to network access device 204 (e.g., firewall modules 210 maintained by firewall processing node(s) 211, which may be a dedicated firewall processing machine. All of the firewall modules 209-210 are managed by firewall controller 208, which may be located within network access device 204, or external to network access device 204, such as, for example, in a public cloud associated with network 203, or in a private cloud associated with network 205. Controller 208 and firewall processing modules 209-210 collectively are referred to herein as distributed firewall 212.").<br><br>Shieh '088 ¶ [0023] ("According to one embodiment, a mechanism is utilized to dynamically perform security inspection in a network. In one embodiment, the mechanism includes two functions: 1) an input/output (IO) function (e.g., firewall module(s) 209) that performs the distribution of network traffic; and 2) a security-processing function (e.g., firewall module(s) 210) that performs security processing, including security inspection and policy enforcement. IO function 209 receives the packets and uses a session table to forward the packets to security-processing function 210. A session table is a data structure that stores connection states, including the destination of security-processing function. In one embodiment, IO function 209 determines, based on an internal data structure such as a session or flow table (e.g., session table as shown in FIG. 5), whether the packet should be forwarded to security processing function 210 for security inspection. The configuration of IO function 209 to control whether to forward the packets to security processing |

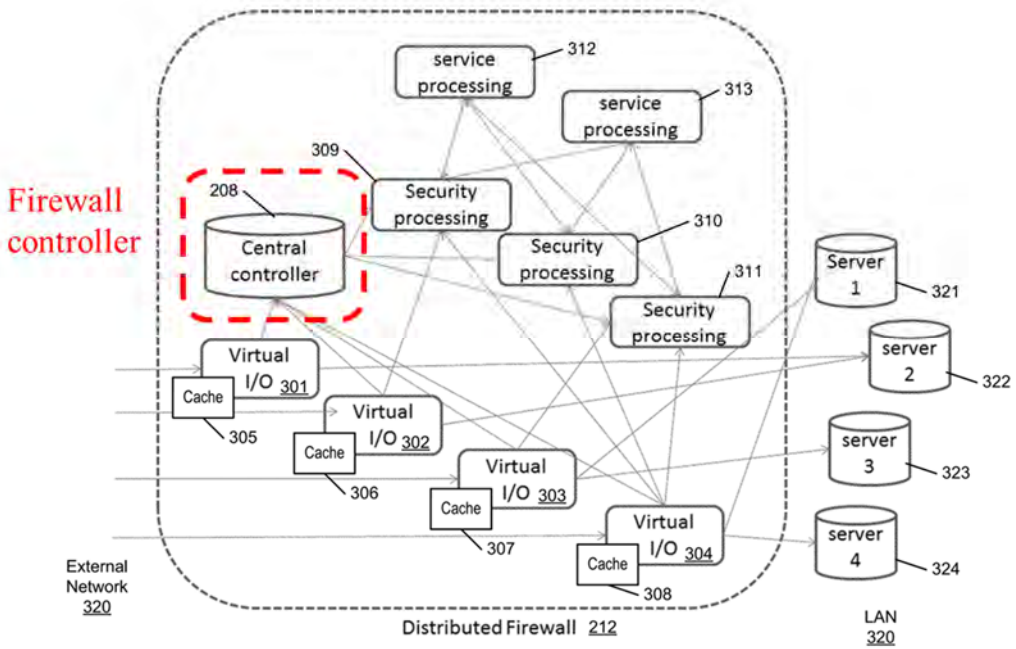


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 820 of 1100

| No. | '111 Patent Claim 13 | Shieh '088 |
|---|---|---|
| | | function 210 can be set based on a command received from an administrator or alternatively, based on a signal received from security processing function 210.").<br><br>Shieh '088 ¶ [0027] ("Referring back to FIG. 2A, in one embodiment, each of the network access devices 204A-204C maintains a flow table or session table (e.g., flow tables 251A-251C) and a firewall module (e.g., 209A-209C).").<br><br>Shieh '088 ¶ [0028] ("Firewall modules 209A-209C may be part of a distributed firewall described above. For example, firewall modules 209A-209C may be the IO functions of a firewall while nodes 211A-211B may be firewall processing nodes. That is, modules 211A-211B may be dedicated firewall processing devices that perform some firewall processing operations such as DPI, content inspection, antivirus, etc., while firewall modules 209A-209C are responsible for routing data packets. For example, when firewall module 209B receives a packet from node 206, it may forward the packet to firewall processing node 211A for content inspection and/or forwards the packet to controller 208 for routing information. In response, firewall processing node 211A analyzes the received packet and/or further communicates with controller 208. Controller 208 may provide further routing information back to network access device 204B regarding how to route the packet. Each of the firewall processing nodes 211A-211B may further maintains a persistent connection or tunnel with controller 208, for example, using the OpenFlow communication protocol.").<br><br>Shieh '088 ¶ [0038] ("FIG. 3 is a block diagram illustrating an example of a distributed firewall according to one embodiment of the invention. Referring to FIG. 3, distributed firewall 212 includes, for the purpose of illustration, four different types of modules: virtual I/O modules 301-304, security processing modules 309-311, service processing modules 312-313, and central controller 208. All these modules can run on the same virtual machine, or on different virtual machines, or on same or different physical hosts. In one embodiment, the communication protocol between the modules is IPC (inter-process communication) if they run on the same memory space, use layer-2 network protocol if they are on the same layer-2 network, or use IP protocols if they are connected through IP networks. Some or all of modules 301-304 and 309-313 may be executed within a virtual machine. Dependent upon the specific configuration, each of modules 301-304 and 309-313 may be executed by a respective virtual machine. In other configurations, multiple of modules 301-304 and 309-313 may be executed by the same virtual machine."). |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 821 of 1100

| No. | '111 Patent Claim 13 | Shieh '088 |
|---|---|---|
| | |  <br>**FIG. 3**<br><br>Fig. 3 (annotation added)<br><br>Shieh '088 ¶ [0043] ("By dividing a firewall into different modules, it allows putting virtual I/O and security processing functions at the best locations to protect the network entrance, while keeping the central control and monitoring functionality at the central controller. It also enhances the scalability of the system since all modules can be expanded to multiple instances to share the system load. Note that a service processing module is optional in the architecture, as it is only required when there are needs to use additional resources to handle the security functions.").<br><br>Shieh '088 ¶ [0044] ("FIG. 6 is a block diagram illustrating architecture of a processing module according to one embodiment of the invention. Referring to FIG. 6, any of processing |

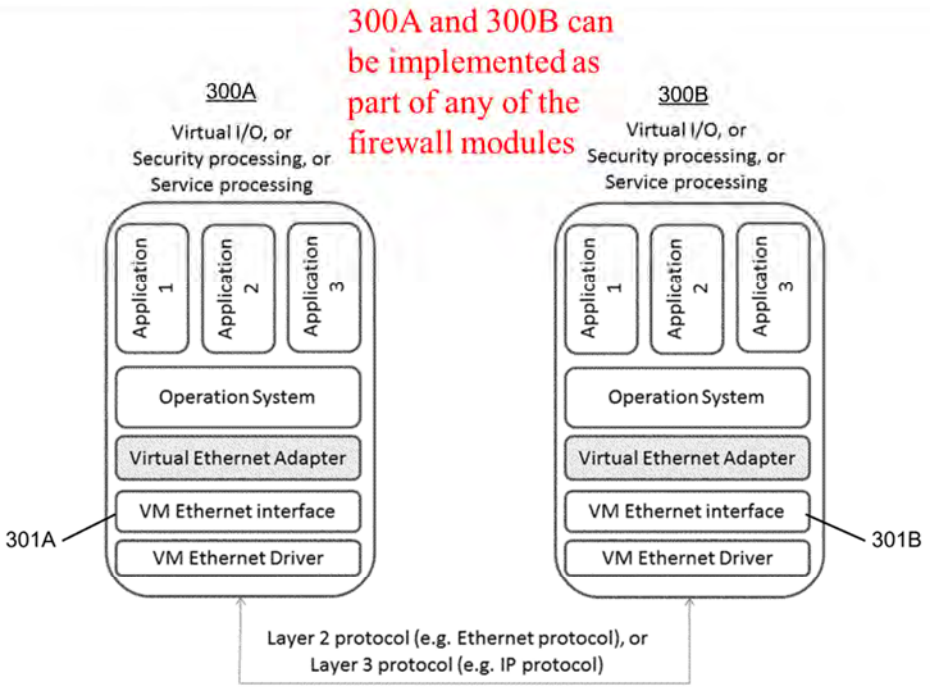


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 822 of 1100

| No. | ’111 Patent Claim 13 | Shieh ’088 |
|---|---|---|
| | | modules 300A and 300B can be implemented as part of any of the firewall modules (e.g., I/O module, security processing module, or service processing module) as shown in FIG. 3. In the example as shown in FIG. 6, multiple possible communication protocols can be utilized for the packet forwarding between firewall modules. If the firewall modules are on the same layer-2 networks, the packet can be forwarded through a layer-2 protocol, such as Ethernet protocol. In this example, it is assumed that each of firewall modules 300 a-300B has a dedicated virtual Ethernet interface (e.g., interfaces 301A and 301B) being used for the forwarding link and the packets are sent with Ethernet header of both sides' media access control (MAC) addresses. The packets can also be forwarded in a layer-3 protocol such as an IP protocol. During the layer-3 routing, original packets are encapsulated with another IP header, which carries the IP address of both sides. The encapsulation of the outer IP address would ensure the packets are sent, and received from the proper peer.”). |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 823 of 1100

| No. | '111 Patent Claim 13 | Shieh '088 |
|---|---|---|
| | |  <br>**FIG. 6**<br><br>Fig. 6 (annotation added)<br><br>Shieh '088 ¶ [0045] ("In one embodiment, firewall modules 300A and 300B can run on virtual machines or physical hosts. Running on virtual machines provides additional benefit that a firewall module can be added dynamically. Initially the distributed firewall may have only one virtual I/O module, one security processing module, and a central controller. When there is more traffic coming, it can add more virtual I/O modules to support increasing connections. If it needs more CPU resources to handle the security processing, it may add more security processing modules and/or add more service processing modules, to support the increasing load. This provides lots of flexibility to support various network conditions.") |

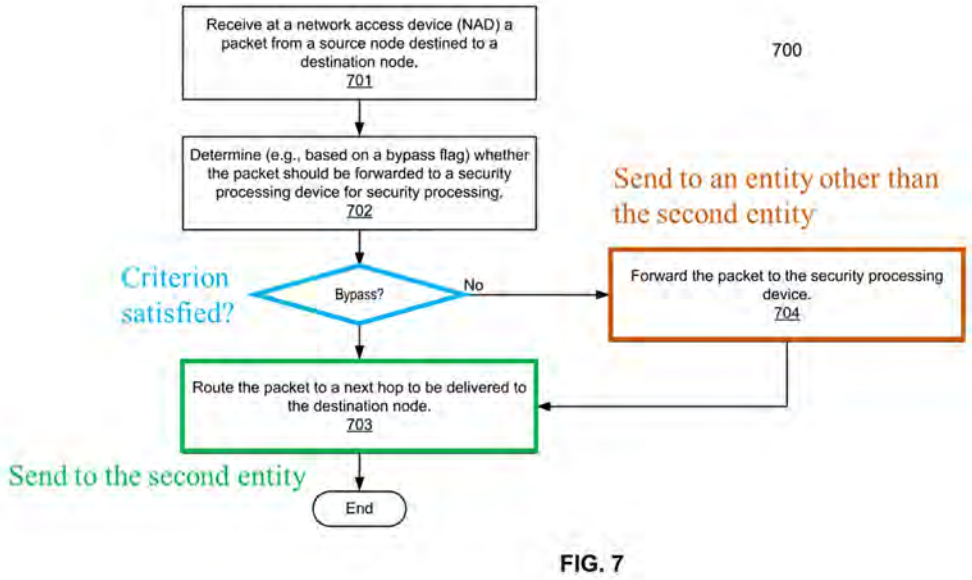


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 824 of 1100

| No. | '111 Patent Claim 13 | Shieh '088 |
|---|---|---|
| | | Shieh '088 ¶ [0046] ("In one embodiment, firewall modules 300A-300B could be distributed in different networks, even on different locations, as long as the modules can reach the module that is next in terms of processing and the central controller. In one embodiment, virtual I/O modules and corresponding security processing modules are in a public cloud and the central controller is in a private cloud. This configuration may provide the flexibility to secure and control packets coming from the public cloud, and allow central controller having overall view of traffic from Internet as well as from internal network.").<br><br>Shieh '088 ¶ [0047] ("One of the advantages of embodiments of the present invention includes, but not limited to, that the distributed firewall can employ a significantly large amount of CPU and memory resources for service processing and protect the networks at multiple geometric locations. The central controller decides which security processing module capable of processing particular connection, and is able to start a new security processing at the place deemed best for packet processing.").<br><br>Shieh '088 ¶ [0048] ("As a result, the location of the packet I/O is not limited on a single appliance. The I/O modules can be placed anywhere as virtual machines. The security processing power is significantly higher as packets and connections can be load balanced to any number of the security processing modules, and the modules could be added or deleted dynamically. Using such modules in a firewall cloud provides a security design that is best-fit for the emerging cloud computing, and provides great scalability and system availability.").<br><br>Shieh '088 ¶ [0049] ("FIG. 7 is a flow diagram illustrating a method for performing firewall operations using a distributed firewall according to one embodiment of the invention. Method 700 may be performed by processing logic that may include software, hardware, or a combination of both. For example, method 700 may be performed by distributed firewall 212 of FIG. 1. Referring to FIG. 7, at block 701, a network access device receives a packet from a source node destined to a destination node. At block 702, the network access device determines whether the packet should be forwarded to a security device for security inspection. For example, processing logic may check whether there is an entry exists in a session table for the current session. If not, it may forward the packet to the security device for security processing at block 704. Alternatively, the processing logic may check whether there is a bypass flag set to a predetermined value for the current session. If there is, the packet |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 825 of 1100

| No. | '111 Patent Claim 13 | Shieh '088 |
|---|---|---|
| | | will not be forwarded to the security device; instead, the packet will be directly routed to the destination node at block 703.").<br>  <br><br>Fig. 7 (annotation added) |

| No. | '111 Patent Claim 14 | Shieh '088 |
|---|---|---|
| 14 | The method according to claim 9, wherein the analyzing comprises performing Deep Packet Inspection (DPI) or using a DPI engine on the packet. | Shieh '088 discloses the method according to claim 9, wherein the analyzing comprises performing Deep Packet Inspection (DPI) or using a DPI engine on the packet.<br><br>For example, Shieh '088 discloses the use of deep packet inspection as part of the analysis functionality.<br><br>Shieh '088 ¶ [0021] ("According to one embodiment, network access device 204 is associated with a distributed firewall 212 that includes various firewall processing modules, for example, each being executed within a virtual machine (VM). In one embodiment, each |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 826 of 1100

| No. | '111 Patent Claim 14 | Shieh '088 |
|---|---|---|
| | | firewall module is responsible for performing one or more firewall functions, but it does not include all of the firewall functions of a firewall. Examples of the firewall functions include, but are not limited to, network address translation (NAT), virtual private network (VPN), deep packet inspection (DPI), and/or anti-virus, etc.").<br><br>Shieh '088 ¶ [0028] ("Firewall modules 209A-209C may be part of a distributed firewall described above. For example, firewall modules 209A-209C may be the IO functions of a firewall while nodes 211A-211B may be firewall processing nodes. That is, modules 211A-211B may be dedicated firewall processing devices that perform some firewall processing operations such as DPI, content inspection, antivirus, etc., while firewall modules 209A-209C are responsible for routing data packets. For example, when firewall module 209B receives a packet from node 206, it may forward the packet to firewall processing node 211A for content inspection and/or forwards the packet to controller 208 for routing information. In response, firewall processing node 211A analyzes the received packet and/or further communicates with controller 208. Controller 208 may provide further routing information back to network access device 204B regarding how to route the packet. Each of the firewall processing nodes 211A-211B may further maintains a persistent connection or tunnel with controller 208, for example, using the OpenFlow communication protocol.").<br><br>Shieh '088 ¶ [0040] ("In one embodiment, each of security processing modules 309-311 performs major security processing functions, such as, for example, NAT, VPN, DPI, and/or anti-virus, etc. A security processing module receives packets and runs the packets through one or more various security functions in the module for security processing. There could be several security modules and each handles the same or different security functions. If the packets need to go through another security or service processing, the module sends the packets to the other modules.").<br><br>Shieh '088 ¶ [0041] ("In one embodiment, each of service processing modules 312-313 performs one or more of the functions of security processing module, such as, for example, NAT, VPN, DPI, and/or anti-virus, etc. However, it is different from the security processing module in that it only receives and sends packets to the same security processing module. If the tasks cannot be done in a security processing module, for example, due to a resource limitation, system load, or the requirement of a different operation system, the packets can be forwarded to one or more of service processing modules 312-313 for further |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 827 of 1100

| No. | '111 Patent Claim 14 | Shieh '088 |
|---|---|---|
| | | processing. The packets then are sent back to the same security processing module for the next security function processing. To further share the system load, any of security processing modules 309-311 can load balance the computational-intensive services using multiple service processing modules.") |

| No. | '111 Patent Claim 15 | Shieh '088 |
|---|---|---|
| 15[a] | The method according to claim 9, wherein the packet comprises distinct header and payload fields, and | Shieh '088 discloses the method according to claim 9, wherein the packet comprises distinct header and payload fields.<br><br>For example, Shieh '088 disclose deep packet inspection, which would be understood to require inspection of at least part of the payload field. A person of ordinary skill in the art would understand that a packet comprises distinct header and payload fields. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Shieh '088 is found to not meet this limitation, wherein the packet comprises distinct header and payload fields would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>Shieh '088 ¶ [0021] ("According to one embodiment, network access device 204 is associated with a distributed firewall 212 that includes various firewall processing modules, for example, each being executed within a virtual machine (VM). In one embodiment, each firewall module is responsible for performing one or more firewall functions, but it does not include all of the firewall functions of a firewall. Examples of the firewall functions include, but are not limited to, network address translation (NAT), virtual private network (VPN), deep packet inspection (DPI), and/or anti-virus, etc.").<br><br>Shieh '088 ¶ [0028] ("Firewall modules 209A-209C may be part of a distributed firewall described above. For example, firewall modules 209A-209C may be the IO functions of a firewall while nodes 211A-211B may be firewall processing nodes. That is, modules 211A-211B may be dedicated firewall processing devices that perform some firewall processing operations such as DPI, content inspection, antivirus, etc., while firewall modules 209A-209C are responsible for routing data packets. For example, when firewall module 209B receives a packet from node 206, it may forward the packet to firewall processing node 211A for content inspection and/or forwards the packet to controller 208 for routing information. In response, |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 828 of 1100

| No. | '111 Patent Claim 15 | Shieh '088 |
|---|---|---|
| | | firewall processing node 211A analyzes the received packet and/or further communicates with controller 208. Controller 208 may provide further routing information back to network access device 204B regarding how to route the packet. Each of the firewall processing nodes 211A-211B may further maintains a persistent connection or tunnel with controller 208, for example, using the OpenFlow communication protocol.").<br><br>Shieh '088 ¶ [0040] ("In one embodiment, each of security processing modules 309-311 performs major security processing functions, such as, for example, NAT, VPN, DPI, and/or anti-virus, etc. A security processing module receives packets and runs the packets through one or more various security functions in the module for security processing. There could be several security modules and each handles the same or different security functions. If the packets need to go through another security or service processing, the module sends the packets to the other modules.").<br><br>Shieh '088 ¶ [0041] ("In one embodiment, each of service processing modules 312-313 performs one or more of the functions of security processing module, such as, for example, NAT, VPN, DPI, and/or anti-virus, etc. However, it is different from the security processing module in that it only receives and sends packets to the same security processing module. If the tasks cannot be done in a security processing module, for example, due to a resource limitation, system load, or the requirement of a different operation system, the packets can be forwarded to one or more of service processing modules 312-313 for further processing. The packets then are sent back to the same security processing module for the next security function processing. To further share the system load, any of security processing modules 309-311 can load balance the computational-intensive services using multiple service processing modules.")<br><br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Shieh '088 in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 15(a) of Exhibit E-4 renders the claim, including the present limitation, obvious. Below is an example.<br><br>For example, Swenson discloses packet flows with header and payload fields. |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 829 of 1100

| No. | '111 Patent Claim 15 | Shieh '088 |
|---|---|---|
| | | Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0040] ("The flow analyzer 312 monitors large flows in the network, analyzes collected flow statistics to determine net-work throughput, and accordingly selects flows to be opti-mized. The flow analyzer 312 does not need to see all the flows in order to make an accurate estimate of network con-ditions. The flow analyzer 312 processes the traffic statistics stored in the flow cache 3 22 and user information stored in the subscriber log 324, for example, by associating network flows identified by source IP addresses to a mobile subscriber or user, which is identified by his or her current subscriber ID or device ID. The user flows are also mapped to a congestion level at the current sub-network (e.g., a cell with which the user devices are associated), so that an optimization decision can be made at the beginning of the data transmission.")<br><br>Swenson at [0049] ("The policy engine 314 defines policies for optimiz-ing large flows with media objects to mitigate network con-gestion. Detecting and acting on congestion in the network, the design focus of the network controller 140 is built on this very flexible policy engine. The policy engine 314 is capable of taking virtually any input, either deduced from HTTP headers and payload ( e.g., through RADIUS/Gx interface), or provided by the network infrastructure via API, and making decisions on how to apply optimization based on individual or a combination of these inputs. The optimization policies can be applied to large flows all the time or on a time-of-day basis, a per user basis, and/or depending on the network condition.")<br><br>Swenson at [0061] ("Once a flow is reported to the network controller 140, a flow cache entry is created for the flow in the flow cache 322. The flow cache entry keeps track of the flow |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 830 of 1100

| No. | ’111 Patent Claim 15 | Shieh ’088 |
|---|---|---|
| | | and its associated bandwidth. For a flow that is marked in "continue" mode, each time the steering device 130 forwards a next portion of the flow payload to the network controller 140, the flow cache 3 22 updates the number of bytes for transmitted in the flow. By monitoring the number of bytes per flow over time, the flow analyzer 312 is capable of determining an estimate value of bandwidth associated with flow. Further-more, since the steering device 130 does not have infinite packet buffers, if congestion happens on the network link 416 from the steering device 130 to the user device 110, the TCP congestion control mechanism kicks in at the steering device 130, which may slows down and/or eventually stop receiving data over the network link 413 from origin server 160. During the congestion, the steering device 130 would not forward any data to the network controller 140, since the link 416 is congested and the network controller 140 would not be able to transmit data to the user device 110. Therefore, as an inline element, the network controller 140 can detect network con-gestions and estimate bandwidth associated with any flows of interest selected by the network controller 140. However, in the "continue" mode, the network controller 140 does not modify and transform the HTTP messaged it receives over the ICAP interface. The network controller 140 simply updates the flow statistics and returns the video or images to the steering device 130 for transmission to the user device 110.”)<br><br>Swenson at [0064] (Similar to the "continue" mode, after receiving the initial HTTP messages of a flow and determining to monitor the flow, the network controller 140 notify the steering device 130 to work in a "counting" mode for bandwidth monitoring. In contrast to the "continue" mode, when a matching flow is detected for "counting" mode, the steering device 130 for-wards the HTTP response directly to the user device 110. While at the same time, the steering device 130 send a cus-tomized ICAP message to the network controller 140 over the network link 425. In one embodiment, the customized ICAP message contains the HTTP request and response headers, as well as a count of payload size of the current flow. After updating the flow statistics, the network controller 140 may acknowledge the gateway over the network line 426. In the "counting" mode, the network controller 140 does not join the network response path as an inline network element, but simply listens to the counting of flow size. The benefit of the "counting" mode is to off-load the network controller 140 from ingesting and forwarding the network flow on the net- work response path, while still enabling the detection of con-gestions and estimation of bandwidth associated with the flows of interest.”) |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 831 of 1100

| No. | '111 Patent Claim 15 | Shieh '088 |
|---|---|---|
| | | Swenson at [0065] ("FIG. 5 is a block diagram illustrating an example event trace of "continue" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 512 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 512 to the intended origin server 160 and receives a response 514 back from the origin server 160. The steering device 130 then sends an ICAP request message 516 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 518. Upon receiving the instruc-tion, the steering device 130 re-writes the response 514 to an HTTP redirect response 520, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 520 directly to the user device 110. In case the flow dose not contain video or image objects, or the network controller 140 determines not to monitor the flow, the steering device 13 0 would forward the response to the user device 110.")<br><br>Swenson at [0069] ("FIG. 6 is a block diagram illustrating an example event trace of "counting" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 612 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 612 to the intended origin server 160 and receives a response 614 back from the origin server 160. The steering device 130 then sends an ICAP request message 616 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 618. Upon receiving the instruc-tion, the steering device 130 re-writes the response 614 to an HTTP redirect response 620, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 620 directly to the user device 110. In case the flow dose not contain video |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 832 of 1100

| No. | '111 Patent Claim 15 | Shieh '088 |
|---|---|---|
| | | or image objects that need to be redirected, the steering device 130 would forward the response to the user device 110.")<br><br>Swenson at [0073] ("FIG. 7 is a block diagram illustrating one embodi-ment of an example of internal components of the flow cache. The flow cache map 700 comprises a plurality of flow cache entries, such as flow cache entries 710 and 712 indexed by a hash. Not shown in the example diagram is a possible linked list behind each flow cache entry which allows chaining of flow cache entries for a given hash index. The hash into the flow cache may be based on source IP address, MAC address, subscriber ID, or other identifier indicative of a given sub-scriber, group of subscribers or subscriber's device.")<br><br>Swenson at [0079] ("In the bandwidth calculation, flows are categorized into buckets based on the size of the objects being transferred. Small objects may not be factored into the bandwidth calcu-lation since they may come and go within a single interval. For example, flows with payload size less than 50 kB may be ignored because a transfer of 50 kB may never reach the full potential throughput of the link. While larger flows may reach the full throughput of the link for a long period of time intervals, they are grouped into 50-75 kB, 75-100 kB and 100 kB+ buckets because the characteristics of these flow sizes can be different, hence the bandwidth for each of the buckets is measured and calculated separately. In other embodiments, the flow size ranges (e.g., 50-75 kB, 75-100 kB and l00kB+) of the buckets may be altered depending on the network traffic and size of objects transmitted. Furthermore, the bucket sizes can also be adjusted based on network topology, such as buffer size, prior to transmission to the client. The calculated bandwidth per bucket is stored in a queue structure that allows for the computing and updating of minimum, maximum, and/or average measurements for each bucket. In one embodiment, the 100 kB+ bucket's current tail entry is checked against the average bandwidth for the 100 kB+ bucket. If the current entry is less than the average multiplied by the number of entries in the queue, the current entry is added to the bandwidth calculation for the current interval. This scheme can filter out large bursts of data from tempo-rarily idle flows. If the bandwidth exceeds the value, a number of bytes (e.g., 125 kB) will be subtracted from the current entry to account for TCP buffers in the network.")<br><br>Swenson at [0083] ("When a new flow is observed, flow cache entries are searched by matching source IP address 722 if the subscriber id or other identifiers of the flow are not |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 833 of 1100

| No. | '111 Patent Claim 15 | Shieh '088 |
|---|---|---|
| | | available. In case of multiple users sharing an IP address, the flow analyzer 312 needs to find patterns or other identifiers in the flows to map them to particular subscribers. Flows without identified sub-scribers are added to the flow cache block under the default user flows 726, which is a default holding place for the new flows. The flow analyzer 312 later will scan through the default user flows that contain cookies or other identifiers that may be used to determine a real user or subscriber associated with the flow. If a flow contains identifiers not associated with an existing real user, a new user or subscriber is created and the user flow block is moved to newly created (or mapped) user or subscriber.") |
| 15[b] | wherein the analyzing comprises checking part of, or whole of, the payload field. | Shieh '088 discloses wherein the analyzing comprises checking part of, or whole of, the payload field.<br><br>For example, Shieh '088 discloses packet analysis, which includes checking the headers, which includes checking in whole or in part the payload field. A person of ordinary skill in the art would understand that deep packet inspection occurs on the payload field. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Shieh '088 is found to not meet this limitation, wherein the analyzing comprises checking part of, or whole of, the payload field would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>Shieh '088 ¶ [0036] ("During the bypass phase, the I/O function may notify the security-processing function if there are special events in the packet stream. These events could be receipt of TCP FIN or TCP RST packets, or not receiving any packets of the connection within a time threshold. The notification from I/O functions to security processing functions could help to clean up the state in the security-processing nodes.").<br><br>Shieh '088 ¶ [0039] "(An I/O module running within a virtual machine is referred to herein as a virtual I/O module. Each of virtual I/O modules 301-304 receives packets from any of servers 321-324 of LAN 320 and sends packets to external network 315 outside of the firewall. In one embodiment, each of I/O modules 301-304 keeps a local cache (e.g., caches 305-308) storing location(s) of a security processing module(s) (e.g., security processing modules 309-311) for each connection session. A cache maintained by each I/O module contains a forwarding table mapping certain connection sessions to any of security modules 309-311. An example of a forwarding table is shown in FIG. 5. Upon receiving a |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 834 of 1100

| No. | '111 Patent Claim 15 | Shieh '088 |
|---|---|---|
| | | packet, an I/O module performs a packet classification to find out the associated connection and forwards the packet to the corresponding security processing module identified by the forwarding table. If it cannot find the connection in its local cache, the packets are forwarded to central controller 208 for processing. In such a case, controller 208 assigns the connection to one of security processing modules 309-311 based on one or more of a variety of factors such as load balancing. The virtual I/O modules 302-304 can be located at multiple locations of the networks to receive and send out packets.").<br><br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Shieh '088 in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 15(b) of Exhibit E-4 renders the claim, including the present limitation, obvious. Below is an example.<br><br>For example, Swenson discloses inspecting the payload of a packet flow.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0040] ("The flow analyzer 312 monitors large flows in the network, analyzes collected flow statistics to determine net-work throughput, and accordingly selects flows to be opti-mized. The flow analyzer 312 does not need to see all the flows in order to make an accurate estimate of network con-ditions. The flow analyzer 312 processes the traffic statistics stored in the flow cache 3 22 and user information stored in the subscriber log 324, for example, by associating network flows identified by source IP addresses to a mobile subscriber or user, which is identified by his or her current subscriber ID or device ID. The |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 835 of 1100

| No. | '111 Patent Claim 15 | Shieh '088 |
|---|---|---|
| | | user flows are also mapped to a congestion level at the current sub-network (e.g., a cell with which the user devices are associated), so that an optimization decision can be made at the beginning of the data transmission.")<br><br>Swenson at [0049] ("The policy engine 314 defines policies for optimiz-ing large flows with media objects to mitigate network con-gestion. Detecting and acting on congestion in the network, the design focus of the network controller 140 is built on this very flexible policy engine. The policy engine 314 is capable of taking virtually any input, either deduced from HTTP headers and payload ( e.g., through RADIUS/Gx interface), or provided by the network infrastructure via API, and making decisions on how to apply optimization based on individual or a combination of these inputs. The optimization policies can be applied to large flows all the time or on a time-of-day basis, a per user basis, and/or depending on the network condition.")<br><br>Swenson at [0061] ("Once a flow is reported to the network controller 140, a flow cache entry is created for the flow in the flow cache 322. The flow cache entry keeps track of the flow and its associated bandwidth. For a flow that is marked in "continue" mode, each time the steering device 130 forwards a next portion of the flow payload to the network controller 140, the flow cache 3 22 updates the number of bytes for transmitted in the flow. By monitoring the number of bytes per flow over time, the flow analyzer 312 is capable of determining an estimate value of bandwidth associated with flow. Further-more, since the steering device 130 does not have infinite packet buffers, if congestion happens on the network link 416 from the steering device 130 to the user device 110, the TCP congestion control mechanism kicks in at the steering device 130, which may slows down and/or eventually stop receiving data over the network link 413 from origin server 160. During the congestion, the steering device 130 would not forward any data to the network controller 140, since the link 416 is congested and the network controller 140 would not be able to transmit data to the user device 110. Therefore, as an inline element, the network controller 140 can detect network con-gestions and estimate bandwidth associated with any flows of interest selected by the network controller 140. However, in the "continue" mode, the network controller 140 does not modify and transform the HTTP messaged it receives over the ICAP interface. The network controller 140 simply updates the flow statistics and returns the video or images to the steering device 130 for transmission to the user device 110.") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 836 of 1100

| No. | ’111 Patent Claim 15 | Shieh ’088 |
|---|---|---|
| | | Swenson at [0064] (Similar to the "continue" mode, after receiving the initial HTTP messages of a flow and determining to monitor the flow, the network controller 140 notify the steering device 130 to work in a "counting" mode for bandwidth monitoring. In contrast to the "continue" mode, when a matching flow is detected for "counting" mode, the steering device 130 for-wards the HTTP response directly to the user device 110. While at the same time, the steering device 130 send a cus-tomized ICAP message to the network controller 140 over the network link 425. In one embodiment, the customized ICAP message contains the HTTP request and response headers, as well as a count of payload size of the current flow. After updating the flow statistics, the network controller 140 may acknowledge the gateway over the network line 426. In the "counting" mode, the network controller 140 does not join the network response path as an inline network element, but simply listens to the counting of flow size. The benefit of the "counting" mode is to off-load the network controller 140 from ingesting and forwarding the network flow on the net- work response path, while still enabling the detection of con-gestions and estimation of bandwidth associated with the flows of interest.”)<br><br>Swenson at [0065] (“FIG. 5 is a block diagram illustrating an example event trace of "continue" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 512 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 512 to the intended origin server 160 and receives a response 514 back from the origin server 160. The steering device 130 then sends an ICAP request message 516 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 518. Upon receiving the instruc-tion, the steering device 130 re-writes the response 514 to an HTTP redirect response 520, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 520 directly to the user device 110. In case the flow dose not contain video or image objects, or the network controller 140 determines not to monitor the flow, the steering device 13 0 would forward the response to the user device 110.”) |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 837 of 1100

| No. | '111 Patent Claim 15 | Shieh '088 |
|---|---|---|
| | | Swenson at [0069] ("FIG. 6 is a block diagram illustrating an example event trace of "counting" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 612 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 612 to the intended origin server 160 and receives a response 614 back from the origin server 160. The steering device 130 then sends an ICAP request message 616 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 618. Upon receiving the instruc-tion, the steering device 130 re-writes the response 614 to an HTTP redirect response 620, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 620 directly to the user device 110. In case the flow dose not contain video or image objects that need to be redirected, the steering device 130 would forward the response to the user device 110.")<br><br>Swenson at [0073] ("FIG. 7 is a block diagram illustrating one embodi-ment of an example of internal components of the flow cache. The flow cache map 700 comprises a plurality of flow cache entries, such as flow cache entries 710 and 712 indexed by a hash. Not shown in the example diagram is a possible linked list behind each flow cache entry which allows chaining of flow cache entries for a given hash index. The hash into the flow cache may be based on source IP address, MAC address, subscriber ID, or other identifier indicative of a given sub-scriber, group of subscribers or subscriber's device.")<br><br>Swenson at [0079] ("In the bandwidth calculation, flows are categorized into buckets based on the size of the objects being transferred. Small objects may not be factored into the bandwidth calcu-lation since they may come and go within a single interval. For example, flows with payload size less than 50 kB may be ignored because a transfer of 50 kB may never reach the full potential throughput of the link. While larger flows may reach the full throughput of the link for a long period of time intervals, they are grouped into 50-75 kB, 75-100 kB and 100 kB+ buckets because the characteristics of these flow sizes can be different, hence the bandwidth for each of the buckets is measured and calculated separately. In other |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 838 of 1100

| No. | '111 Patent Claim 15 | Shieh '088 |
|---|---|---|
| | | embodiments, the flow size ranges (e.g., 50-75 kB, 75-100 kB and l00kB+) of the buckets may be altered depending on the network traffic and size of objects transmitted. Furthermore, the bucket sizes can also be adjusted based on network topology, such as buffer size, prior to transmission to the client. The calculated bandwidth per bucket is stored in a queue structure that allows for the computing and updating of minimum, maximum, and/or average measurements for each bucket. In one embodiment, the 100 kB+ bucket's current tail entry is checked against the average bandwidth for the 100 kB+ bucket. If the current entry is less than the average multiplied by the number of entries in the queue, the current entry is added to the bandwidth calculation for the current interval. This scheme can filter out large bursts of data from tempo-rarily idle flows. If the bandwidth exceeds the value, a number of bytes (e.g., 125 kB) will be subtracted from the current entry to account for TCP buffers in the network.")<br><br>Swenson at [0083] ("When a new flow is observed, flow cache entries are searched by matching source IP address 722 if the subscriber id or other identifiers of the flow are not available. In case of multiple users sharing an IP address, the flow analyzer 312 needs to find patterns or other identifiers in the flows to map them to particular subscribers. Flows without identified sub-scribers are added to the flow cache block under the default user flows 726, which is a default holding place for the new flows. The flow analyzer 312 later will scan through the default user flows that contain cookies or other identifiers that may be used to determine a real user or subscriber associated with the flow. If a flow contains identifiers not associated with an existing real user, a new user or subscriber is created and the user flow block is moved to newly created (or mapped) user or subscriber.") |

| No. | '111 Patent Claim 16 | Shieh '088 |
|---|---|---|
| 16[a] | The method according to claim 1, wherein the packet comprises distinct header and payload fields, | Shieh '088 discloses the method according to claim 1, wherein the packet comprises distinct header and payload fields.<br><br>*See supra* Claim 15[a]. |

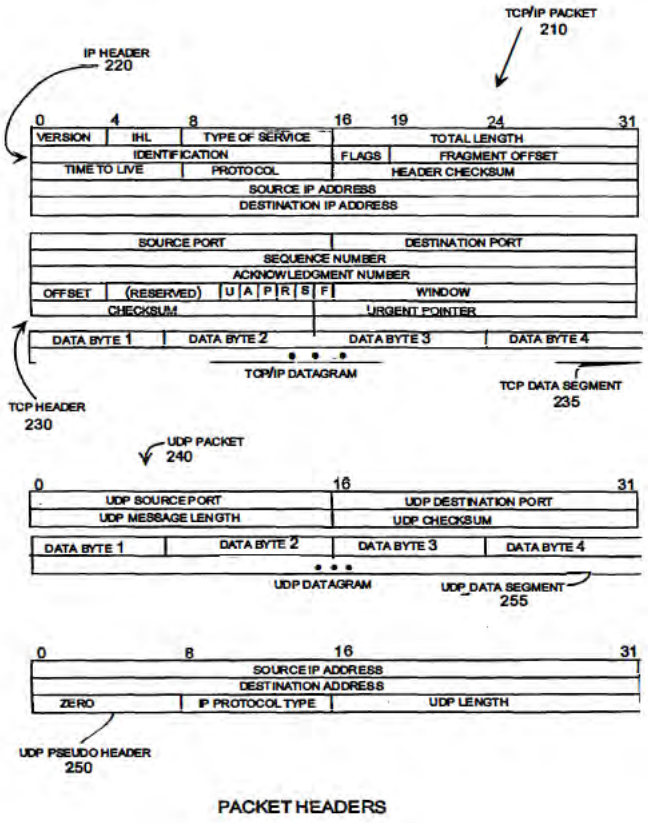


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 839 of 1100

| No. | '111 Patent Claim 16 | Shieh '088 |
|---|---|---|
| 16[b] | the header comprises one or more flag bits, and | Shieh '088 discloses that the header comprises one or more flag bits.<br><br>For example, Shieh '088 discloses header fields in a packet and how they are used to route packets. Shieh '088 further discloses that the packet streams passing through its system include TCP FIN or TCP RST packets; a TCP FIN packet includes a FIN flag bit in its header that is set, and a TCP RST packet includes a RST flag bit in its header that is set. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Shieh '088 is found to not meet this limitation, the header comprises one or more flag bits would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>Shieh '088 ¶ [0036] ("During the bypass phase, the I/O function may notify the security-processing function if there are special events in the packet stream. These events could be receipt of TCP FIN or TCP RST packets, or not receiving any packets of the connection within a time threshold. The notification from I/O functions to security processing functions could help to clean up the state in the security-processing nodes.").<br><br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Shieh '088 in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 16[b] of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references.<br><br>For example, Copeland discloses packet headers with flag bits.<br><br>Copeland at Figure 2 |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 840 of 1100

| No. | '111 Patent Claim 16 | Shieh '088 |
|---|---|---|
| | |  <br>FIG. 2<br><br>Copeland at [0076] ("FIG. 2 illustrates an exemplary TCP/IP packet or datagram 210 and an exemplary UDP datagram 240. In a typical TCP/IP packet like 210, each packet typically includes a header portion comprising an IP header 220 and a TCP header 230, followed by a data portion that contains the information to be communicated in the packet. The information in the IP header 220 contained in a TCP/IP packet 210, or any other IP packet, contains the IP addresses and assures that the packet is delivered to the right host. The transport layer protocol (TCP) header follows the Internet protocol header and specifies the port numbers for the associated service.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 841 of 1100

| No. | '111 Patent Claim 16 | Shieh '088 |
|---|---|---|
| | | Copeland at [0077] ("The header portion in the typical TCP/IP datagram 210 is 40 bytes including 20 bytes of IP header 220 information and 20 bytes of TCP header 230 information. The data portion or segment associated with the packet 210 follows the header information.")<br><br>Copeland at [0078] ("In regards to a typical IP packet 210, the first 4 bits of the IP header 220 identify the Internet protocol (IP) version. The following 4 bits identify the IP header length in 32 bit words. The next 8 bits differentiate the type of service by describing how the packet should be handled in transit. The following 16 bits convey the total packet length.")<br><br>Copeland at [0081] ("In a TCP/IP datagram 210, the initial data of the IP datagram is the TCP header 230 information. The initial TCP header 230 information includes the 16-bit source and 16-bit destination port numbers. A 32-bit sequence number for the data in the packet follows the port numbers. Following the sequence number is a 32-bit acknowledgement number. If an ACK flag (discussed below) is set, this number is the next sequence number the sender of the packet expects to receive. Next is a 4-bit data offset, which is the number of 32-bit words in the TCP header. A 6-bit reserved field follows.")<br><br>Copeland at [0082] ("Following the reserved field, the next 6 bits are a series of one-bit flags, shown in FIG. 2 as flags U, A, P, R, S, F. The first flag is the urgent flag (U). If the U flag is set, it indicates that the urgent pointer is valid and points to urgent data that should be acted upon as soon as possible. The next flag is the A ( or ACK or "acknowledgment") flag. The ACK flag indicates that an acknowledgment number is valid, and acknowledges that data has been received. The next flag, the push (P) flag, tells the receiving end to push all buffered data to the receiving application. The reset (R) flag is the following flag, which terminates both ends of the TCP connection. Next, the S (or SYN for "synchronize") flag is set in the initial packet of a TCP connection where both ends have to synchronize their TCP buffers. Following the SYN flag is the F (for FIN or "finish") flag. This flag signifies that the sending end of the communication and the host will not send any more data but still may acknowledge data that is received.")<br><br>Copeland at [0083] ("Following the TCP flag bits is a 16-bit receive window size field that specifies the amount of space avail-able in the receive buffer for the TCP connection. The |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 842 of 1100

| No. | '111 Patent Claim 16 | Shieh '088 |
|---|---|---|
| | | checksum of the TCP header is a 16-bit field. Following the checksum is a 16 bit urgent pointer that points to the urgent data. The TCP/IP datagram data follows the TCP header.")<br><br>Copeland at [0116] ("These steps generally require manipulations of quantities such as IP addresses, packet length, header length, start times, end times, port numbers, and other packet related information. Usually, though not necessarily, these quanti-ties take the form of electrical, magnetic, or optical signals capable of being stored, transferred, combined, compared, or otherwise manipulated. It is conventional for those skilled in the art to refer to these signals as bits, bytes, words, values, elements, symbols, characters, terms, numbers, points, records, objects, images, files or the like. It should be kept in mind, however, that these and similar terms should be associated with appropriate quantities for computer opera-tions and that these terms are merely conventional labels applied to quantities that exist within and during operation of the computer.")<br><br>As another example, Kempf discloses packet headers with flag bits.<br><br>Kempf at [0081] ("In one embodiment, OpenFlow is modified to pro-vide rules for GTP TEID Routing. FIG. 17 is a diagram of one embodiment of the OpenFlow flow table modification for GTP TEID routing. An OpenFlow switch that supports TEID routing matches on the 2 byte (16 bit) collection of header fields and the 4 byte (32 bit) GTP TEID, in addition to other OpenFlow header fields, in at least one flow table ( e.g., the first flow table). The GTP TEID flag can be wildcarded (i.e. matches are "don't care"). In one embodiment, the EPC pro-tocols do not assign any meaning to TEIDs other than as an endpoint identifier for tunnels, like ports in standard UDP/ TCP transport protocols. In other embodiments, the TEIDs can have a correlated meaning or semantics. The GTP header flags field can also be wildcarded, this can be partially matched by combining the following bitmasks: 0xFF00- Match the Message Type field; 0xe0-Match the Version field; 0xl0-Match the PT field; 0x04-Match the E field; 0x02-Match the S field; and 0x0l-Match the PN field.")<br><br>Kempf at [0082] ("In one embodiment, OpenFlow can be modified to support virtual ports for fast path GTP TEID encapsulation and decapsulation. An OpenFlow mobile gateway can be used to support GTP encapsulation and decapsulation with virtual ports. The GTP encapsulation and decapsulation virtual ports can be used for fast encapsulation and decapsulation of user data packets within GTP-U tunnels, and can be designed simply enough |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 843 of 1100

| No. | '111 Patent Claim 16 | Shieh '088 |
|---|---|---|
| | | that they can be implemented in hardware or firmware. For this reason, GTP virtual ports may have the following restrictions on traffic they will handle: Protocol Type (PT) field= 1, where GTP encapsulation ports only sup-port GTP, not GTP' (PT field=0); Extension Header flag (E)=0, where no extension headers are supported, Sequence Number flag (S)=0, where no sequence numbers are sup-ported; N-PDU flag (PN)=0; and Message type=255, where Only G-PDU messages, i.e. tunneled user data, is supported in the fast path.")<br><br>Kempf at [0083] ("If a packet either needs encapsulation or arrives encapsulated with nonzero header flags, header extensions, and/or the GTP-U packet is not a G-PDU packet (i.e. it is a GTP-U control packet), the processing must proceed via the gateway's slow path (software) control plane. GTP-C and GTP' packets directed to the gateway's IP address are a result of mis-configuration and are in error. They must be sent to the OpenFlow controller, since these packets are handled by the S-GW-C and P-GW-C control plane entities in the cloud computing system or to the billing entity handling GTP' and not the S-GW-D and P-GW-D data plane switches.")<br><br>Kempf at [0088] ("To support slow path encapsulation, the software control plane on the switch maintains a hash table with keys calculated from the GTP-U TEID. The TEID hash keys are calculated using a suitable hash algorithm with low collision frequency, for example SHA-1. The flow table entries contain a record of how the packet header, including the GTP encap-sulation header, should be configured. This includes: the same header fields as for the hardware or firmware encapsu-lation table in FIG.18; values for the GTP header flags (PT, E, S, and PN); the sequence number and/or the N-PDU number if any; if the E flag is 1, then the flow table contains a list of the extension headers, including their types, which the slow path should insert into the GTP header.")<br><br>Kempf at [0092] ("In one embodiment, the system implements a GTP fast path decapsulation virtual port. When requested by the S-GW and P-GW control plane software running in the cloud computing system, the gateway switch installs rules and actions for routing GTP encapsulated packets out of GTP tunnels. The rules match the GTP header flags and the GTP TEID for the packet, in the modified OpenFlow flow table shown in FIG. 17 as follows: the IP destination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); the UDP destination port is the GTP-U destination port (2152); and the header fields and message type field is wildcarded with the flag 0XFFF0 and the upper |

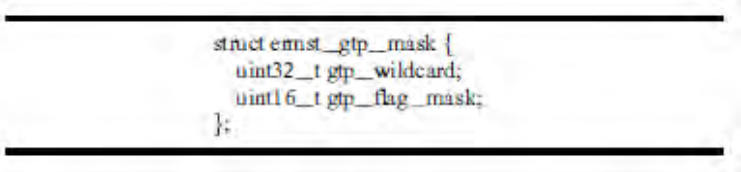


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 844 of 1100

| No. | '111 Patent Claim 16 | Shieh '088 |
|---|---|---|
| | | two bytes of the field match the G-PDU message type (255) while the lower two bytes match 0x30, i.e. the packet is a GTP packet not a GTP' packet and the version number is 1.")<br><br>Kempf at [0094] ("In one embodiment, the system implements han-dling of GTP-U control packets. The OpenFlow controller programs the gateway switch flow tables with 5 rules for each gateway switch IP address used for GTP traffic. These rules contain specified values for the following fields: the IP des-tination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); the UDP destination port is the GTP-U destination port (2152); the GTP header flags and message type field is wildcarded with 0xFFF0; the value of the header flags field is 0x30, i.e. the version number is 1 and the PT field is 1; and the value of the message type field is one of 1 (Echo Request), 2 (Echo Response), 26 (Error Indication), 31 (Support for Extension Headers Notification), or 254 (End Marker).")<br><br>Kempf at [0098] ("The header flags and message type fields for the three rules are wildcarded with the following bitmasks and match as follows: bitmask 0xFFF4 and the upper two bytes match the G-PDU message type (255) while the lower two bytes are Ox34, indicating that the version number is 1, the packet is a GTP packet, and there is an extension header present; bitmask 0xFFFF2 and the upper two bytes match the G-PDU message type (255) while the lower two bytes are 0x32, indicating that the version number is 1, the packet is a GTP packet, and there is a sequence number present; and bitmask 0xFF0l and the upper two bytes match the G-PDU message type (255) while the lower two bytes are 0x31, indicating that the version number is 1, the packet is a GTP packet, and a N-PDU is present.")<br><br>Kempf at [0114] ("The gtp_type_n_flags field contains the GTP mes-sage type in the upper 8 bits and the GTP header flags in the lower 8 bits. The gtp_teid field contains the GTP TEID. The gtp_ wildcard field indicates whether the GTP type and flags and TEID should be matched. If the lower four bits are 1, the type and flags field should be ignored, while if the upper four bits are 1, the TEID should be ignored. If the lower bits are 0, the type and fields flag should be matched subject to the flags in the gtp_flag_mask field, while if the upper bits are 0 the TEID should be matched. The mask is combined with the message type and header field of the packet using logical AND; the result becomes the value of the match. Only those parts of the field in which the mask has a 1 value are matched.") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 845 of 1100

| No. | '111 Patent Claim 16 | Shieh '088 |
|---|---|---|
| | | Kempf at [0117] ("The gtp_type_n_flags field contains the GTP mes-sage type in the upper 8 bits and the GTP header flags in the lower 8 bits. The gtp_<br>teid field contains the GRP TEID. When the value of the oxm_type ( oxm_class+oxm_field is GTP _ MATCH and the HM bit is zero, the flaw's GTP header must match these values exactly. If the HM flag is one, the value contains an ersmt_gtp_match field and an ermst_gtp_mask field, as specified by the OpenFlow 1.2 specification. We define ermst_gtp_mask field for selecting flows based on the settings of flag bits:<br>  <br>struct ermst_gtp_mask {<br>uint32_t gtp_wildcard;<br>uint16_t gtp_flag_mask;<br>};<br><br>Kempf at [0118] ("The gtp_ wildcard field indicates whether the TEID should be matched. If the value is 0xFFFFFFFF, the TEID should be matched and not the flags, if the value is 0x00000000, the flags should be matched and not the TEID. If the gtp_ wildcard indicates the flags should be matched, the gtp_flag_mask is combined with the message type and header field of the packet using logical AND, the result becomes the value of the match. Only those parts of the field in which the mask has a 1 value are matched.") |
| 16[c] | wherein the packet-applicable criterion is that one or more of the flag bits is set. | Shieh '088 discloses wherein the packet-applicable criterion is that one or more of the flag bits is set.<br><br>For example, Shieh '088 discloses that the packet streams passing through its system include TCP FIN or TCP RST packets; a TCP FIN packet includes a FIN flag bit in its header that is set, and a TCP RST packet includes a RST flag bit in its header that is set. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Shieh '088 is found to not meet this limitation, wherein the packet applicable criterion is that one or more of the flag bits is set would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>Shieh '088 ¶ [0036] ("During the bypass phase, the I/O function may notify the security-processing function if there are special events in the packet stream. These events could be |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 846 of 1100

| No. | ’111 Patent Claim 16 | Shieh ’088 |
|---|---|---|
| | | receipt of TCP FIN or TCP RST packets, or not receiving any packets of the connection within a time threshold. The notification from I/O functions to security processing functions could help to clean up the state in the security-processing nodes.”).<br><br>Under at least the apparent claim scope alleged by Orckit’s Infringement Disclosures, Shieh ‘088 in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 16[c] of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references.<br><br>For example, Copeland discloses packet specific characteristics including flag bits that are set.<br><br>Copeland at [0081] (“In a TCP/IP datagram 210, the initial data of the IP datagram is the TCP header 230 information. The initial TCP header 230 information includes the 16-bit source and 16-bit destination port numbers. A 32-bit sequence number for the data in the packet follows the port numbers. Following the sequence number is a 32-bit acknowledgement number. If an ACK flag (discussed below) is set, this number is the next sequence number the sender of the packet expects to receive. Next is a 4-bit data offset, which is the number of 32-bit words in the TCP header. A 6-bit reserved field follows.”)<br><br>Copeland at [0082] (“Following the reserved field, the next 6 bits are a series of one-bit flags, shown in FIG. 2 as flags U, A, P, R, S, F. The first flag is the urgent flag (U). If the U flag is set, it indicates that the urgent pointer is valid and points to urgent data that should be acted upon as soon as possible. The next flag is the A ( or ACK or "acknowledgment") flag. The ACK flag indicates that an acknowledgment number is valid, and acknowledges that data has been received. The next flag, the push (P) flag, tells the receiving end to push all buffered data to the receiving application. The reset (R) flag is the following flag, which terminates both ends of the TCP connection. Next, the S (or SYN for "synchronize") flag is set in the initial packet of a TCP connection where both ends have to synchronize their TCP buffers. Following the SYN flag is the F (for FIN or "finish") flag. This flag signifies that the sending end of the communication and the host will not send any more data but still may acknowledge data that is received.”) |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 847 of 1100

| No. | '111 Patent Claim 16 | Shieh '088 |
|---|---|---|
| | | Copeland at [0083] ("Following the TCP flag bits is a 16-bit receive window size field that specifies the amount of space avail-able in the receive buffer for the TCP connection. The checksum of the TCP header is a 16-bit field. Following the checksum is a 16 bit urgent pointer that points to the urgent data. The TCP/IP datagram data follows the TCP header.")<br><br>Copeland at [0089] ("FIG. 3 illustrates an exemplary TCP/IP session 300. As discussed in reference to FIG. 2, the SYN flag is set whenever one host initiates a session with another host. In the initial packet, Hostl sends a message with only the SYN flag set. The SYN flag is designed to establish a TCP connection and allow both ends to synchronize their TCP buffers. Hostl provides the sequence of the first data packet it will send.")<br><br>Copeland at [0125] ("For purposes of the description, which follows, the IP address with the lower value, when considered as a 32-bit unsigned integer, is designated ip[0] and the corresponding port number is designated pt[0]. The higher IP address is designated ip[l] and the corresponding TCP or UDP port number is designated pt[l]. At some point, either pt[0] or pt[l] may be designated the "server" port by setting an appropriate bit in a bit map that is part of the flow record (record "state", bit 1 or 2 is set).")<br><br>Copeland at [0145] ("A list IP of addresses contacted or probed by each host can be maintained. When this list indicates that more than a threshold number of other hosts (e.g., 8) have been contacted in the same subnet, CI is added to the to the host and a bit in the host record is set to indicate that the host has received CI for "address scanning." Note that the number of hosts to designate a scan is not required to be a fixed value, but could be adjusted based on the sample rate or other means to enhance the accuracy making the number of hosts scanned "statistically significant". These and other values of concern index are shown for non-flow based events in FIG. 7.")<br><br>As another example, Kempf flow table matches in which the flag bits is set,<br><br>Kempf at [0081] ("In one embodiment, OpenFlow is modified to pro-vide rules for GTP TEID Routing. FIG. 17 is a diagram of one embodiment of the OpenFlow flow table modification for GTP TEID routing. An OpenFlow switch that supports TEID routing matches on the 2 byte (16 bit) collection of header fields and the 4 byte (32 bit) GTP TEID, in addition to other OpenFlow header fields, in at least one flow table ( e.g., the first flow table). The GTP TEID |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 848 of 1100

| No. | ’111 Patent Claim 16 | Shieh ’088 |
|---|---|---|
| | | flag can be wildcarded (i.e. matches are "don't care"). In one embodiment, the EPC pro-tocols do not assign any meaning to TEIDs other than as an endpoint identifier for tunnels, like ports in standard UDP/ TCP transport protocols. In other embodiments, the TEIDs can have a correlated meaning or semantics. The GTP header flags field can also be wildcarded, this can be partially matched by combining the following bitmasks: 0xFF00- Match the Message Type field; 0xe0-Match the Version field; 0xl0-Match the PT field; 0x04-Match the E field; 0x02-Match the S field; and 0x0l-Match the PN field.”)<br><br>Kempf at [0082] (“In one embodiment, OpenFlow can be modified to support virtual ports for fast path GTP TEID encapsulation and decapsulation. An OpenFlow mobile gateway can be used to support GTP encapsulation and decapsulation with virtual ports. The GTP encapsulation and decapsulation virtual ports can be used for fast encapsulation and decapsulation of user data packets within GTP-U tunnels, and can be designed simply enough that they can be implemented in hardware or firmware. For this reason, GTP virtual ports may have the following restrictions on traffic they will handle: Protocol Type (PT) field= 1, where GTP encapsulation ports only sup-port GTP, not GTP' (PT field=0); Extension Header flag (E)=0, where no extension headers are supported, Sequence Number flag (S)=0, where no sequence numbers are sup-ported; N-PDU flag (PN)=0; and Message type=255, where Only G-PDU messages, i.e. tunneled user data, is supported in the fast path.”)<br><br>Kempf at [0083] (“If a packet either needs encapsulation or arrives encapsulated with nonzero header flags, header extensions, and/or the GTP-U packet is not a G-PDU packet (i.e. it is a GTP-U control packet), the processing must proceed via the gateway's slow path (software) control plane. GTP-C and GTP' packets directed to the gateway's IP address are a result of mis-configuration and are in error. They must be sent to the OpenFlow controller, since these packets are handled by the S-GW-C and P-GW-C control plane entities in the cloud computing system or to the billing entity handling GTP' and not the S-GW-D and P-GW-D data plane switches.”)<br><br>Kempf at [0088] (“To support slow path encapsulation, the software control plane on the switch maintains a hash table with keys calculated from the GTP-U TEID. The TEID hash keys are calculated using a suitable hash algorithm with low collision frequency, for example SHA-1. The flow table entries contain a record of how the packet header, including the GTP encap-sulation header, should be configured. This includes: the same header fields as for the |

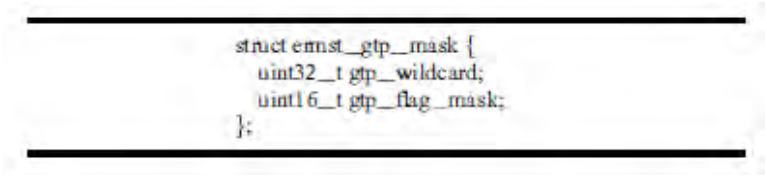


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 849 of 1100

| No. | '111 Patent Claim 16 | Shieh '088 |
|---|---|---|
| | | hardware or firmware encapsu-lation table in FIG.18; values for the GTP header flags (PT, E, S, and PN); the sequence number and/or the N-PDU number if any; if the E flag is 1, then the flow table contains a list of the extension headers, including their types, which the slow path should insert into the GTP header.")<br><br>Kempf at [0092] ("In one embodiment, the system implements a GTP fast path decapsulation virtual port. When requested by the S-GW and P-GW control plane software running in the cloud computing system, the gateway switch installs rules and actions for routing GTP encapsulated packets out of GTP tunnels. The rules match the GTP header flags and the GTP TEID for the packet, in the modified OpenFlow flow table shown in FIG. 17 as follows: the IP destination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); the UDP destination port is the GTP-U destination port (2152); and the header fields and message type field is wildcarded with the flag 0XFFF0 and the upper two bytes of the field match the G-PDU message type (255) while the lower two bytes match 0x30, i.e. the packet is a GTP packet not a GTP' packet and the version number is 1.")<br><br>Kempf at [0094] ("In one embodiment, the system implements han-dling of GTP-U control packets. The OpenFlow controller programs the gateway switch flow tables with 5 rules for each gateway switch IP address used for GTP traffic. These rules contain specified values for the following fields: the IP des-tination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); the UDP destination port is the GTP-U destination port (2152); the GTP header flags and message type field is wildcarded with 0xFFF0; the value of the header flags field is 0x30, i.e. the version number is 1 and the PT field is 1; and the value of the message type field is one of 1 (Echo Request), 2 (Echo Response), 26 (Error Indication), 31 (Support for Extension Headers Notification), or 254 (End Marker).")<br><br>Kempf at [0098] ("The header flags and message type fields for the three rules are wildcarded with the following bitmasks and match as follows: bitmask 0xFFF4 and the upper two bytes match the G-PDU message type (255) while the lower two bytes are Ox34, indicating that the version number is 1, the packet is a GTP packet, and there is an extension header present; bitmask 0xFFFF2 and the upper two bytes match the G-PDU message type (255) while the lower two bytes are 0x32, indicating that the version number is 1, the packet is a GTP packet, and there is a sequence number present; and bitmask 0xFF0l and the upper two bytes match |



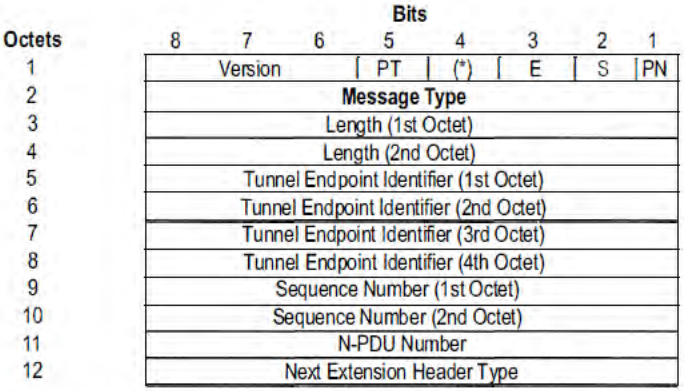
Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 850 of 1100

| No. | '111 Patent Claim 16 | Shieh '088 |
|---|---|---|
| | | the G-PDU message type (255) while the lower two bytes are 0x31, indicating that the version number is 1, the packet is a GTP packet, and a N-PDU is present.")<br><br>Kempf at [0114] ("The gtp_type_n_flags field contains the GTP mes-sage type in the upper 8 bits and the GTP header flags in the lower 8 bits. The gtp_teid field contains the GTP TEID. The gtp_ wildcard field indicates whether the GTP type and flags and TEID should be matched. If the lower four bits are 1, the type and flags field should be ignored, while if the upper four bits are 1, the TEID should be ignored. If the lower bits are 0, the type and fields flag should be matched subject to the flags in the gtp_flag_mask field, while if the upper bits are 0 the TEID should be matched. The mask is combined with the message type and header field of the packet using logical AND; the result becomes the value of the match. Only those parts of the field in which the mask has a 1 value are matched.")<br><br>Kempf at [0117] ("The gtp_type_n_flags field contains the GTP mes-sage type in the upper 8 bits and the GTP header flags in the lower 8 bits. The gtp_teid field contains the GRP TEID. When the value of the oxm_type ( oxm_class+oxm_field is GTP _ MATCH and the HM bit is zero, the flaw's GTP header must match these values exactly. If the HM flag is one, the value contains an ersmt_gtp_match field and an ermst_gtp_mask field, as specified by the OpenF!ow 1.2 specification. We define ermst_gtp_mask field for selecting flows based on the settings of flag bits:<br><br>  <br>struct ermst_gtp_mask {<br>uint32_t gtp_wildcard;<br>uint16_t gtp_flag_mask;<br>};<br><br>Kempf at [0118] ("The gtp_ wildcard field indicates whether the TEID should be matched. If the value is 0xFFFFFFFF, the TEID should be matched and not the flags, if the value is 0x00000000, the flags should be matched and not the TEID. If the gtp_ wildcard indicates the flags should be matched, the gtp_flag_mask is combined with the message type and header field of the packet using logical AND, the result becomes the value of the match. Only those parts of the field in which the mask has a 1 value are matched.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 851 of 1100

| No. | '111 Patent Claim 16 | Shieh '088 |
|---|---|---|
| | | Kempf at Figure 10 |



| Octets | Bits 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
|---|---|---|---|---|---|---|---|---|
| 1 | Version | | | PT | (*) | E | S | PN |
| 2 | Message Type | | | | | | | |
| 3 | Length (1st Octet) | | | | | | | |
| 4 | Length (2nd Octet) | | | | | | | |
| 5 | Tunnel Endpoint Identifier (1st Octet) | | | | | | | |
| 6 | Tunnel Endpoint Identifier (2nd Octet) | | | | | | | |
| 7 | Tunnel Endpoint Identifier (3rd Octet) | | | | | | | |
| 8 | Tunnel Endpoint Identifier (4th Octet) | | | | | | | |
| 9 | Sequence Number (1st Octet) | | | | | | | |
| 10 | Sequence Number (2nd Octet) | | | | | | | |
| 11 | N-PDU Number | | | | | | | |
| 12 | Next Extension Header Type | | | | | | | |

NOTE 0: (*) This bit is a spare bit. It shall be sent as '0'. The receiver shl not evaluate this bit,
NOTE 1: 1) This field shall only be evaluated when indicated by the S flag set to 1.
NOTE 2: 2) This field shall only be evaluated when indicated by the PN flag set to 1.
NOTE 3: 3) This field shall only be evaluated when indicated by the E flag set to 1.
NOTE 4: 4) This field shall be present if and only if any one or more of the S. PN and E flags are set.

FIG. 10

| No. | '111 Patent Claim 17 | Shieh '088 |
|---|---|---|
| 17[a] | The method according to claim 16, wherein the packet is an Transmission Control Protocol (TCP) packet, and | Shieh '088 discloses the method according to claim 16, wherein the packet is a Transmission Control Protocol (TCP) packet.<br><br>For example, Shieh '088 discloses that the packet streams passing through its system include TCP packets.<br><br>Shieh '088 ¶ [0027] ("Referring back to FIG. 2A, in one embodiment, each of the network access devices 204A-204C maintains a flow table or session table (e.g., flow tables 251A-251C) and a firewall module (e.g., 209A-209C). A network flow refers to a sequence of packets from a source computer to a destination, which may be another host, a multicast |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 852 of 1100

| No. | '111 Patent Claim 17 | Shieh '088 |
|---|---|---|
| | | group, or a broadcast domain. For example, a TCP/IP flow can be uniquely identified by the following parameters within a certain time period: 1) Source and Destination IP address; 2) Source and Destination Port; and 3) Layer 4 Protocol (TCP/UDP/ICMP). A session is a semi-permanent interactive information interchange, also known as a dialogue, a conversation or a meeting, between two or more communicating devices. A session is set up or established at a certain point in time and torn down at a later point in time. An established communication session may involve more than one message in each direction. A session is typically, but not always, stateful, meaning that at least one of the communicating entities needs to save information about the session history in order to be able to communicate, as opposed to stateless communication, where the communication consists of independent requests with responses. Flow tables 251A-251C may be implemented as a combination of a flow table and a session table.").<br><br>Shieh '088 ¶ [0036] ("During the bypass phase, the I/O function may notify the security-processing function if there are special events in the packet stream. These events could be receipt of TCP FIN or TCP RST packets, or not receiving any packets of the connection within a time threshold. The notification from I/O functions to security processing functions could help to clean up the state in the security-processing nodes."). |
| 17[b] | wherein the one or more flag bits comprises comprise a SYN flag bit, an ACK flag bit, a FIN flag bit, a RST flag bit, or any combination thereof. | Shieh '088 discloses wherein the one or more flag bits comprises comprise a SYN flag bit, an ACK flag bit, a FIN flag bit, a RST flag bit, or any combination thereof.<br><br>For example, Shieh '088 discloses that some of the packets can be TCP FIN or TCP RST packets.<br><br>Shieh '088 ¶ [0036] ("During the bypass phase, the I/O function may notify the security-processing function if there are special events in the packet stream. These events could be receipt of TCP FIN or TCP RST packets, or not receiving any packets of the connection within a time threshold. The notification from I/O functions to security processing functions could help to clean up the state in the security-processing nodes."). |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 853 of 1100

| No. | '111 Patent Claim 18 | Shieh '088 |
|---|---|---|
| 18[a] | The method according to claim 1, wherein the packet comprises distinct header and payload fields, | Shieh '088 discloses the method according to claim 1, wherein the packet comprises distinct header and payload fields.<br><br>*See supra* Claim 15[a]. |
| 18[b] | the header comprises at least the first and second entities addresses in the packet network, and | Shieh '088 discloses the header comprises at least the first and second entities addresses in the packet network.<br><br>For example, Shieh '088 discloses unique parameters of a TCP/IP packet flow such as source and destination addresses in packet headers.<br><br>Shieh '088 ¶ [0031] ("According to one embodiment, referring back to FIG. 2A, when a security-processing function (e.g., processing node 211A) receives the packets, it does the security inspection and security policy enforcement. The packets then are forwarded to the next I/O function (e.g., modules 209A-209C). The choices of the next I/O function could be from the decision from layer 2 such as Ethernet MAC address lookup, or IP address routing, or other methods.").<br><br>Shieh '088 ¶ [0027] ("Referring back to FIG. 2A, in one embodiment, each of the network access devices 204A-204C maintains a flow table or session table (e.g., flow tables 251A-251C) and a firewall module (e.g., 209A-209C). A network flow refers to a sequence of packets from a source computer to a destination, which may be another host, a multicast group, or a broadcast domain. For example, a TCP/IP flow can be uniquely identified by the following parameters within a certain time period: 1) Source and Destination IP address; 2) Source and Destination Port; and 3) Layer 4 Protocol (TCP/UDP/ICMP). A session is a semi-permanent interactive information interchange, also known as a dialogue, a conversation or a meeting, between two or more communicating devices. A session is set up or established at a certain point in time and torn down at a later point in time. An established communication session may involve more than one message in each direction. A session is typically, but not always, stateful, meaning that at least one of the communicating entities needs to save information about the session history in order to be able to communicate, as opposed to stateless communication, where the communication consists of independent requests with responses. Flow tables 251A-251C may be implemented as a combination of a flow table and a session table."). |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 854 of 1100

| No. | '111 Patent Claim 18 | Shieh '088 |
|---|---|---|
| | | Shieh '088 ¶ [0026] ("The OpenFlow technology consists of three parts: flow tables installed on switches, a controller, and an OpenFlow protocol for the controller to talk securely with switches. Flow tables are set up on switches or routers. Controllers talk to the switches via the OpenFlow Protocol, which is secure, and impose policies on flows. For example, a simple flow might be defined as any traffic from a given IP address. The rule governing it might be to route the flow through a given switch port. With its knowledge of the network, the controller could set up paths through the network optimized for speed, fewest number of hops or reduced latency, among other characteristics. Using OpenFlow takes control of how traffic flows through the network out of the hands of the infrastructure, the switches and routers, and puts it in the hands of the network owner (such as a corporation), individual users or individual applications.").<br><br>Shieh '088 ¶ [0044] ("FIG. 6 is a block diagram illustrating architecture of a processing module according to one embodiment of the invention. Referring to FIG. 6, any of processing modules 300A and 300B can be implemented as part of any of the firewall modules (e.g., I/O module, security processing module, or service processing module) as shown in FIG. 3. In the example as shown in FIG. 6, multiple possible communication protocols can be utilized for the packet forwarding between firewall modules. If the firewall modules are on the same layer-2 networks, the packet can be forwarded through a layer-2 protocol, such as Ethernet protocol. In this example, it is assumed that each of firewall modules 300 a-300B has a dedicated virtual Ethernet interface (e.g., interfaces 301A and 301B) being used for the forwarding link and the packets are sent with Ethernet header of both sides' media access control (MAC) addresses. The packets can also be forwarded in a layer-3 protocol such as an IP protocol. During the layer-3 routing, original packets are encapsulated with another IP header, which carries the IP address of both sides. The encapsulation of the outer IP address would ensure the packets are sent, and received from the proper peer."). |
| 18[c] | wherein the packet-applicable criterion is that the first entity address, the second entity address, or both | Shieh '088 discloses wherein the packet-applicable criterion is that the first entity address, the second entity address, or both match a predetermined address or addresses. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 855 of 1100

| No. | '111 Patent Claim 18 | Shieh '088 |
|---|---|---|
| | match a predetermined address or addresses. | For example, Shieh '088 discloses that IP or MAC addresses are packet-applicable criterion used to determine whether to forward packets to a security processing function based on whether the IP or MAC addresses match a predetermined address or addresses.<br><br>Shieh '088 ¶ [0031] ("According to one embodiment, referring back to FIG. 2A, when a security-processing function (e.g., processing node 211A) receives the packets, it does the security inspection and security policy enforcement. The packets then are forwarded to the next I/O function (e.g., modules 209A-209C). The choices of the next I/O function could be from the decision from layer 2 such as Ethernet MAC address lookup, or IP address routing, or other methods.").<br><br>Shieh '088 ¶ [0027] ("Referring back to FIG. 2A, in one embodiment, each of the network access devices 204A-204C maintains a flow table or session table (e.g., flow tables 251A-251C) and a firewall module (e.g., 209A-209C). A network flow refers to a sequence of packets from a source computer to a destination, which may be another host, a multicast group, or a broadcast domain. For example, a TCP/IP flow can be uniquely identified by the following parameters within a certain time period: 1) Source and Destination IP address; 2) Source and Destination Port; and 3) Layer 4 Protocol (TCP/UDP/ICMP). A session is a semi-permanent interactive information interchange, also known as a dialogue, a conversation or a meeting, between two or more communicating devices. A session is set up or established at a certain point in time and torn down at a later point in time. An established communication session may involve more than one message in each direction. A session is typically, but not always, stateful, meaning that at least one of the communicating entities needs to save information about the session history in order to be able to communicate, as opposed to stateless communication, where the communication consists of independent requests with responses. Flow tables 251A-251C may be implemented as a combination of a flow table and a session table.").<br><br>Shieh '088 ¶ [0026] ("The OpenFlow technology consists of three parts: flow tables installed on switches, a controller, and an OpenFlow protocol for the controller to talk securely with switches. Flow tables are set up on switches or routers. Controllers talk to the switches via the OpenFlow Protocol, which is secure, and impose policies on flows. For example, a simple flow might be defined as any traffic from a given IP address. The rule governing it might be to route the flow through a given switch port. With its knowledge of the network, the |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 856 of 1100

| No. | '111 Patent Claim 18 | Shieh '088 |
|---|---|---|
| | | controller could set up paths through the network optimized for speed, fewest number of hops or reduced latency, among other characteristics. Using OpenFlow takes control of how traffic flows through the network out of the hands of the infrastructure, the switches and routers, and puts it in the hands of the network owner (such as a corporation), individual users or individual applications.").<br><br>Shieh '088 ¶ [0044] ("FIG. 6 is a block diagram illustrating architecture of a processing module according to one embodiment of the invention. Referring to FIG. 6, any of processing modules 300A and 300B can be implemented as part of any of the firewall modules (e.g., I/O module, security processing module, or service processing module) as shown in FIG. 3. In the example as shown in FIG. 6, multiple possible communication protocols can be utilized for the packet forwarding between firewall modules. If the firewall modules are on the same layer-2 networks, the packet can be forwarded through a layer-2 protocol, such as Ethernet protocol. In this example, it is assumed that each of firewall modules 300 a-300B has a dedicated virtual Ethernet interface (e.g., interfaces 301A and 301B) being used for the forwarding link and the packets are sent with Ethernet header of both sides' media access control (MAC) addresses. The packets can also be forwarded in a layer-3 protocol such as an IP protocol. During the layer-3 routing, original packets are encapsulated with another IP header, which carries the IP address of both sides. The encapsulation of the outer IP address would ensure the packets are sent, and received from the proper peer."). |

| No. | '111 Patent Claim 19 | Shieh '088 |
|---|---|---|
| 19 | The method according to claim 18, wherein the addresses are Internet Protocol (IP) addresses. | Shieh '088 discloses the method according to claim 18, wherein the addresses are Internet Protocol (IP) addresses.<br><br>For example, Shieh '088 discloses that IP addresses are used to determine whether to forward packets to a security processing function based on whether the IP address matches a predetermined address or addresses.<br><br>*See supra* Claim 18. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 857 of 1100

| No. | '111 Patent Claim 19 | Shieh '088 |
|---|---|---|
| | | Shieh '088 ¶ [0031] ("According to one embodiment, referring back to FIG. 2A, when a security-processing function (e.g., processing node 211A) receives the packets, it does the security inspection and security policy enforcement. The packets then are forwarded to the next I/O function (e.g., modules 209A-209C). The choices of the next I/O function could be from the decision from layer 2 such as Ethernet MAC address lookup, or IP address routing, or other methods."). <br><br> Shieh '088 ¶ [0027] ("Referring back to FIG. 2A, in one embodiment, each of the network access devices 204A-204C maintains a flow table or session table (e.g., flow tables 251A-251C) and a firewall module (e.g., 209A-209C). A network flow refers to a sequence of packets from a source computer to a destination, which may be another host, a multicast group, or a broadcast domain. For example, a TCP/IP flow can be uniquely identified by the following parameters within a certain time period: 1) Source and Destination IP address; 2) Source and Destination Port; and 3) Layer 4 Protocol (TCP/UDP/ICMP). A session is a semi-permanent interactive information interchange, also known as a dialogue, a conversation or a meeting, between two or more communicating devices. A session is set up or established at a certain point in time and torn down at a later point in time. An established communication session may involve more than one message in each direction. A session is typically, but not always, stateful, meaning that at least one of the communicating entities needs to save information about the session history in order to be able to communicate, as opposed to stateless communication, where the communication consists of independent requests with responses. Flow tables 251A-251C may be implemented as a combination of a flow table and a session table."). <br><br> Shieh '088 ¶ [0026] ("The OpenFlow technology consists of three parts: flow tables installed on switches, a controller, and an OpenFlow protocol for the controller to talk securely with switches. Flow tables are set up on switches or routers. Controllers talk to the switches via the OpenFlow Protocol, which is secure, and impose policies on flows. For example, a simple flow might be defined as any traffic from a given IP address. The rule governing it might be to route the flow through a given switch port. With its knowledge of the network, the controller could set up paths through the network optimized for speed, fewest number of hops or reduced latency, among other characteristics. Using OpenFlow takes control of how traffic flows through the network out of the hands of the infrastructure, the switches and routers, and |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 858 of 1100

| No. | '111 Patent Claim 19 | Shieh '088 |
|---|---|---|
| | | puts it in the hands of the network owner (such as a corporation), individual users or individual applications.").<br><br>Shieh '088 ¶ [0044] ("FIG. 6 is a block diagram illustrating architecture of a processing module according to one embodiment of the invention. Referring to FIG. 6, any of processing modules 300A and 300B can be implemented as part of any of the firewall modules (e.g., I/O module, security processing module, or service processing module) as shown in FIG. 3. In the example as shown in FIG. 6, multiple possible communication protocols can be utilized for the packet forwarding between firewall modules. If the firewall modules are on the same layer-2 networks, the packet can be forwarded through a layer-2 protocol, such as Ethernet protocol. In this example, it is assumed that each of firewall modules 300 a-300B has a dedicated virtual Ethernet interface (e.g., interfaces 301A and 301B) being used for the forwarding link and the packets are sent with Ethernet header of both sides' media access control (MAC) addresses. The packets can also be forwarded in a layer-3 protocol such as an IP protocol. During the layer-3 routing, original packets are encapsulated with another IP header, which carries the IP address of both sides. The encapsulation of the outer IP address would ensure the packets are sent, and received from the proper peer."). |

| No. | '111 Patent Claim 20 | Shieh '088 |
|---|---|---|
| 20[a] | The method according to claim 1, wherein the packet is an Transmission Control Protocol (TCP) packet that comprises source and destination TCP ports, a TCP sequence number, and a TCP sequence mask fields, and | Shieh '088 discloses the method according to claim 1, wherein the packet is an Transmission Control Protocol (TCP) packet that comprises source and destination TCP ports, a TCP sequence number, and a TCP sequence mask fields.<br><br>For example, Shieh '088 discloses the transportation of TCP packets in a TCP network using TCP protocol, and describes tracking TCP packets that can be identified by their source and destination port.<br><br>Shieh '088 ¶ [0027] ("Referring back to FIG. 2A, in one embodiment, each of the network access devices 204A-204C maintains a flow table or session table (e.g., flow tables 251A-251C) and a firewall module (e.g., 209A-209C). A network flow refers to a sequence of packets from a source computer to a destination, which may be another host, a multicast group, or a broadcast domain. For example, a TCP/IP flow can be uniquely identified by the following parameters within a certain time period: 1) Source and Destination IP address; 2) |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 859 of 1100

| No. | '111 Patent Claim 20 | Shieh '088 |
|---|---|---|
| | | Source and Destination Port; and 3) Layer 4 Protocol (TCP/UDP/ICMP). A session is a semi-permanent interactive information interchange, also known as a dialogue, a conversation or a meeting, between two or more communicating devices. A session is set up or established at a certain point in time and torn down at a later point in time. An established communication session may involve more than one message in each direction. A session is typically, but not always, stateful, meaning that at least one of the communicating entities needs to save information about the session history in order to be able to communicate, as opposed to stateless communication, where the communication consists of independent requests with responses. Flow tables 251A-251C may be implemented as a combination of a flow table and a session table.").<br><br>Shieh '088 ¶ [0036] ("During the bypass phase, the I/O function may notify the security-processing function if there are special events in the packet stream. These events could be receipt of TCP FIN or TCP RST packets, or not receiving any packets of the connection within a time threshold. The notification from I/O functions to security processing functions could help to clean up the state in the security-processing nodes."). |
| 20[b] | wherein the packet-applicable criterion is that the source TCP port, the destination TCP port, the TCP sequence number, the TCP sequence mask, or any combination thereof, matches a predetermined value or values. | Shieh '088 discloses wherein the packet-applicable criterion is that the source TCP port or the destination TCP port.<br><br>For example, Shieh '088 discloses using a source TCP port or a destination TCP port are used to determine whether to take an action on the packet.<br><br>Shieh '088 ¶ [0027] ("Referring back to FIG. 2A, in one embodiment, each of the network access devices 204A-204C maintains a flow table or session table (e.g., flow tables 251A-251C) and a firewall module (e.g., 209A-209C). A network flow refers to a sequence of packets from a source computer to a destination, which may be another host, a multicast group, or a broadcast domain. For example, a TCP/IP flow can be uniquely identified by the following parameters within a certain time period: 1) Source and Destination IP address; 2) Source and Destination Port; and 3) Layer 4 Protocol (TCP/UDP/ICMP). A session is a semi-permanent interactive information interchange, also known as a dialogue, a conversation or a meeting, between two or more communicating devices. A session is set up or established at a certain point in time and torn down at a later point in time. An established communication session may involve more than one message in each direction. A session is typically, but not |



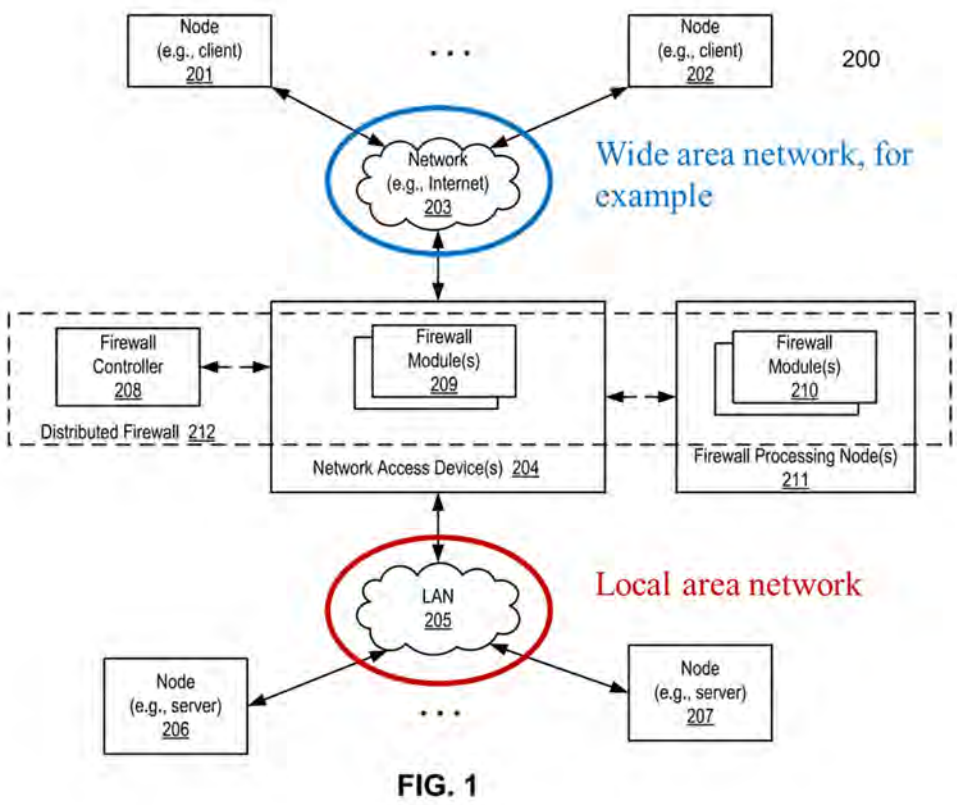
Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 860 of 1100

| No. | '111 Patent Claim 20 | Shieh '088 |
|---|---|---|
| | | always, stateful, meaning that at least one of the communicating entities needs to save information about the session history in order to be able to communicate, as opposed to stateless communication, where the communication consists of independent requests with responses. Flow tables 251A-251C may be implemented as a combination of a flow table and a session table."). <br><br> Shieh '088 ¶ [0036] ("During the bypass phase, the I/O function may notify the security-processing function if there are special events in the packet stream. These events could be receipt of TCP FIN or TCP RST packets, or not receiving any packets of the connection within a time threshold. The notification from I/O functions to security processing functions could help to clean up the state in the security-processing nodes."). |

| No. | '111 Patent Claim 21 | Shieh '088 |
|---|---|---|
| 21 | The method according to claim 1, wherein the packet network comprises a Wide Area Network (WAN), Local Area Network (LAN), the Internet, Metropolitan Area Network (MAN), Internet Service Provider (ISP) backbone datacenter network, or inter - datacenter network. | Shieh '088 discloses wherein the packet network comprises a Wide Area Network (WAN) or Local Area Network (LAN) or Internet Service Provider (ISP) backbone data center network. <br><br> For example, Shieh '088 discloses that its packet network includes the Internet, a WAN or LAN, and that its system can be in communication with an ISP. <br><br> Shieh '088 ¶ [0020] ("FIG. 1 is a block diagram illustrating an example of network configuration according to one embodiment of the invention. Referring to FIG. 1, network access device 204, which may be a router or gateway, a switch or an access point, etc., provides an interface between network 203 and network 205. Network 203 may be an external network such as a wide area network (WAN) (e.g., Internet) while network 205 represents a local area network (LAN). Nodes 206-207 go through gateway device 204 in order to reach nodes 201-202, or vice versa. Any of nodes 201-202 and 206-207 may be a client device (e.g., a desktop, laptop, Smartphone, gaming device) or a server."). |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 861 of 1100

| No. | '111 Patent Claim 21 | Shieh '088 |
|---|---|---|
| | |  <br>FIG. 1<br>Fig. 1 (annotation added)<br><br>Shieh '088 ¶ [0053] ("Modem 447 may provide a direct connection to a remote server via a telephone link or to the Internet via an internet service provider (ISP). Network interface 448 may provide a direct connection to a remote server. Network interface 448 may provide a direct connection to a remote server via a direct network link to the Internet via a POP (point of presence). Network interface 448 may provide such connection using wireless techniques, including digital cellular telephone connection, a packet connection, digital satellite data connection or the like."). |

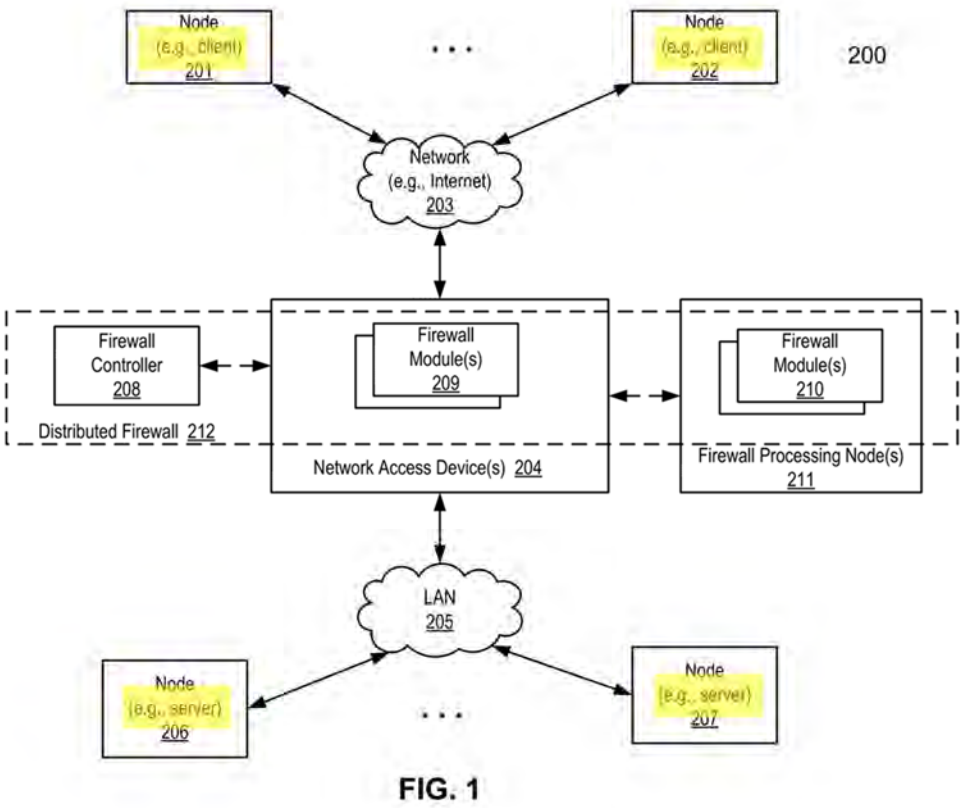


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 862 of 1100

| No. | '111 Patent Claim 22 | Shieh '088 |
|---|---|---|
| 22 | The method according to claim 1, wherein the first entity is a server device and the second entity is a client device, or wherein the first entity is a client device and the second entity is a server device. | Shieh '088 discloses the method according to claim 1, wherein the first entity is a server device and the second entity is a client device, or wherein the first entity is a client device and the second entity is a server device.<br><br>For example, Shieh '088 discloses, in Figure 1, that the source of the network packets can be a server device, and the destination of the claimed packets can be a client device, or vice versa.<br><br>Shieh '088 ¶ [0020] ("FIG. 1 is a block diagram illustrating an example of network configuration according to one embodiment of the invention. Referring to FIG. 1, network access device 204, which may be a router or gateway, a switch or an access point, etc., provides an interface between network 203 and network 205. Network 203 may be an external network such as a wide area network (WAN) (e.g., Internet) while network 205 represents a local area network (LAN). Nodes 206-207 go through gateway device 204 in order to reach nodes 201-202, or vice versa. Any of nodes 201-202 and 206-207 may be a client device (e.g., a desktop, laptop, Smartphone, gaming device) or a server."). |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 863 of 1100

| No. | '111 Patent Claim 22 | Shieh '088 |
|---|---|---|
| | |  <br>Fig. 1 (annotation added)<br><br>Shieh '088 ¶ [0022] ("A virtual machine represents a completely isolated operating environment with a dedicated set of resources associated therewith. A virtual machine may be installed or launched as a guest operating system (OS) hosted by a host OS. In one embodiment, a host OS represents a virtual machine monitor (VMM) (also referred to as a hypervisor) for managing the hosted virtual machines. A guest OS may be of the same or different types with respect to the host OS. For example, a guest OS may be a Windows™ operating system and a host OS may be a LINUX operating system. In addition, the guest operating systems (OSes) running on a host can be of the same or different types. A virtual |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 864 of 1100

| No. | '111 Patent Claim 22 | Shieh '088 |
|---|---|---|
| | | machine can be any type of virtual machine, such as, for example, hardware emulation, full virtualization, para-virtualization, and operating system-level virtualization virtual machines. Different virtual machines hosted by a server may have the same or different privilege levels for accessing different resources.").<br><br>Shieh '088 ¶ [0057] ("Referring to FIG. 8, the memory 460 includes a monitoring module 801 which when executed by a processor is responsible for performing traffic monitoring of traffic from the VMs as described above. Memory 460 also stores one or more IO modules 802 which, when executed by a processor, is responsible for performing forwarding inbound and outbound packets. Memory 460 further stores one or more security processing modules 803 which, when executed by a processor, is responsible for security processes on the packets provided by IO modules 802. Memory 460 also stores one or more optional service processing modules 804, which when executed by a processor performs a particular security process on behalf of security processing modules 803. The memory also includes a network communication module 805 used for performing network communication and communication with the other devices (e.g., servers, clients, etc.)."). |

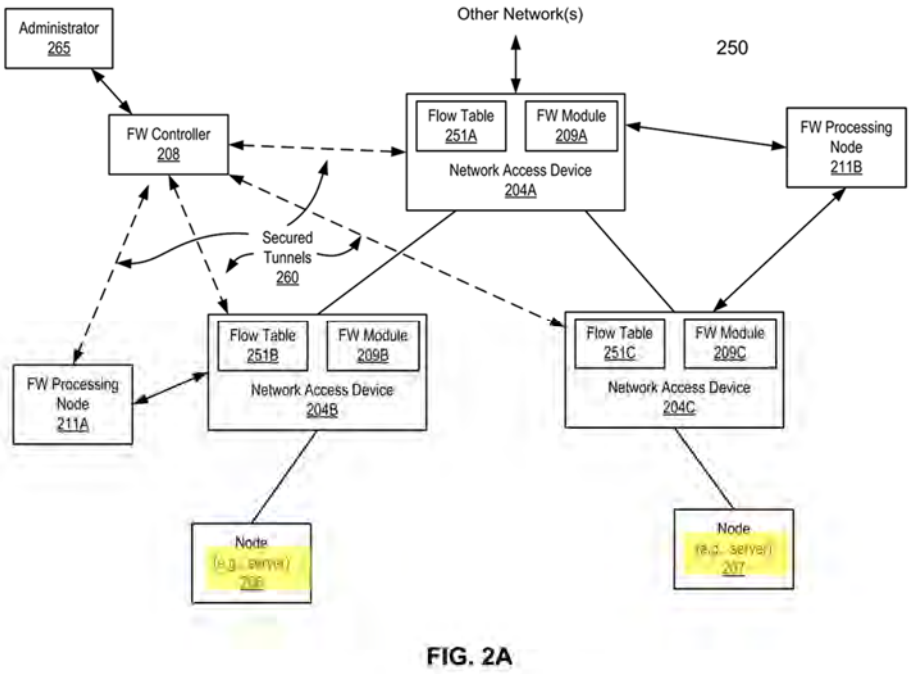
| No. | '111 Patent Claim 23 | Shieh '088 |
|---|---|---|
| 23[a] | The method according to claim 22, wherein the server device comprises a web server, and | Shieh '088 discloses the method according to claim 22, wherein the server device comprises a web server.<br><br>For example, Shieh '088 discloses that the source or destination of the packets can be a server device. Shieh '088 further discloses that the network interface my provide a direct connection to a remote server via a direct link to the Internet via POP.<br><br>Shieh '088 ¶ [0039] ("An I/O module running within a virtual machine is referred to herein as a virtual I/O module. Each of virtual I/O modules 301-304 receives packets from any of servers 321-324 of LAN 320 and sends packets to external network 315 outside of the firewall. In one embodiment, each of I/O modules 301-304 keeps a local cache (e.g., caches 305-308) storing location(s) of a security processing module(s) (e.g., security processing modules 309-311) for each connection session. A cache maintained by each I/O module contains a forwarding table mapping certain connection sessions to any of security modules 309-311. An example of a forwarding table is shown in FIG. 5. Upon receiving a |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 865 of 1100

| No. | '111 Patent Claim 23 | Shieh '088 |
|---|---|---|
| | | packet, an I/O module performs a packet classification to find out the associated connection and forwards the packet to the corresponding security processing module identified by the forwarding table. If it cannot find the connection in its local cache, the packets are forwarded to central controller 208 for processing. In such a case, controller 208 assigns the connection to one of security processing modules 309-311 based on one or more of a variety of factors such as load balancing. The virtual I/O modules 302-304 can be located at multiple locations of the networks to receive and send out packets.").<br><br>Shieh '088 ¶ [0053] ("Modem 447 may provide a direct connection to a remote server via a telephone link or to the Internet via an internet service provider (ISP). Network interface 448 may provide a direct connection to a remote server. Network interface 448 may provide a direct connection to a remote server via a direct network link to the Internet via a POP (point of presence). Network interface 448 may provide such connection using wireless techniques, including digital cellular telephone connection, a packet connection, digital satellite data connection or the like."). |

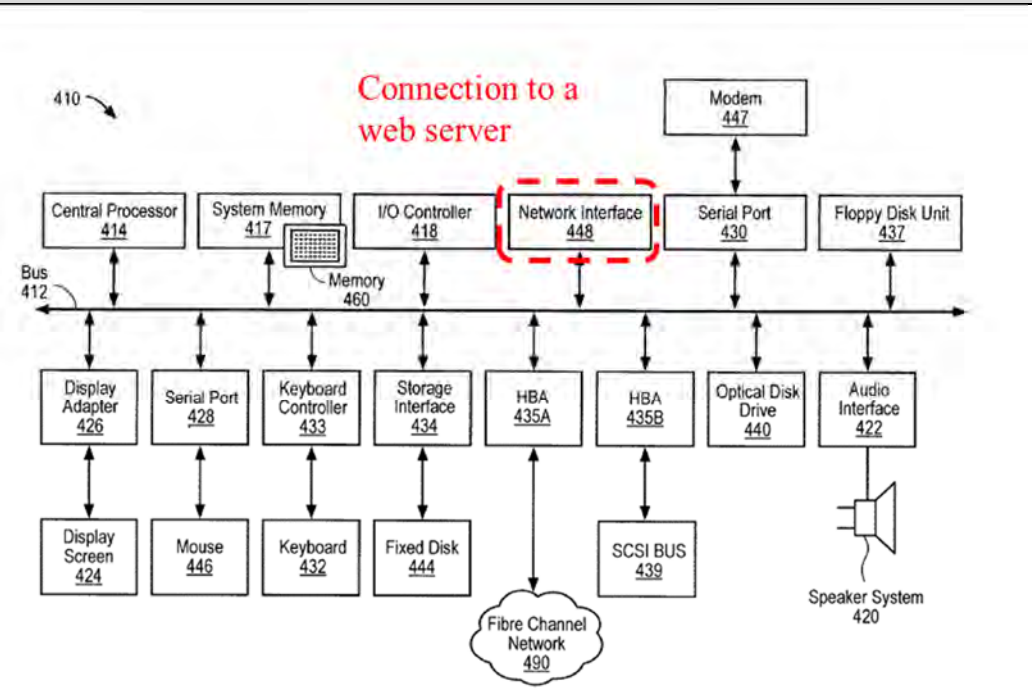


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 866 of 1100

| No. | '111 Patent Claim 23 | Shieh '088 |
|---|---|---|
| | |  <br>Fig. 2A (annotation added) |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 867 of 1100

| No. | '111 Patent Claim 23 | Shieh '088 |
|---|---|---|
| | |  <br>**FIG. 4**<br><br>Fig. 4 (annotation added)<br><br>*See supra* Claim 22. |
| 23[b] | wherein the client device comprises a smartphone, a tablet computer, a personal computer, a laptop computer, or a wearable computing device. | Shieh '088 discloses wherein the client device comprises a smartphone, a tablet computer, a personal computer, a laptop computer, or a wearable computing device.<br><br>For example, Shieh '088 discloses that any of the nodes may be a client device such as a desktop, laptop, smartphone, gaming device, etc.<br><br>Shieh '088 ¶ [0020] ("FIG. 1 is a block diagram illustrating an example of network configuration according to one embodiment of the invention. Referring to FIG. 1, network |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 868 of 1100

| No. | '111 Patent Claim 23 | Shieh '088 |
|---|---|---|
| | | access device 204, which may be a router or gateway, a switch or an access point, etc., provides an interface between network 203 and network 205. Network 203 may be an external network such as a wide area network (WAN) (e.g., Internet) while network 205 represents a local area network (LAN). Nodes 206-207 go through gateway device 204 in order to reach nodes 201-202, or vice versa. Any of nodes 201-202 and 206-207 may be a client device (e.g., a desktop, laptop, Smartphone, gaming device) or a server."). |

| No. | '111 Patent Claim 24 | Shieh '088 |
|---|---|---|
| 24 | The method according to claim 22, wherein the communication between the network node and the controller is based on, or uses, a standard protocol. | Shieh '088 discloses the method according to claim 22, wherein the communication between the network node and the controller is based on, or uses, a standard protocol.<br><br>For example, Shieh '088 discloses the use of the standard OpenFlow protocol for communication between the network access devices and the controller.<br><br>Shieh '088 ¶ [0025] ("According to one embodiment, each of network access devices 204A-204C maintains a persistent connection such as secure connections or tunnels 260 with a controller or management entity 208 for exchanging management messages and configurations, or distributing routing information to network access devices 204A-204C, etc. In one embodiment, controller 208 communicates with each of the network access devices 204A-204C using a management protocol such as the OpenFlow™ protocol. OpenFlow is a Layer 2 communications protocol (e.g., media access control or MAC layer) that gives access to the forwarding plane of a network switch or router over the network. In simpler terms, OpenFlow allows the path of network packets through the network of switches to be determined by software running on multiple routers (minimum two of them, primary and secondary, having a role of observers). This separation of the control from the forwarding allows for more sophisticated traffic management than is feasible using access control lists (ACLs) and routing protocols.").<br><br>Shieh '088 ¶ [0026] ("The OpenFlow technology consists of three parts: flow tables installed on switches, a controller, and an OpenFlow protocol for the controller to talk securely with switches. Flow tables are set up on switches or routers. Controllers talk to the switches via the OpenFlow Protocol, which is secure, and impose policies on flows. For example, a simple flow might be defined as any traffic from a given IP address. The rule governing it might be |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 869 of 1100

| No. | '111 Patent Claim 24 | Shieh '088 |
|---|---|---|
| | | to route the flow through a given switch port. With its knowledge of the network, the controller could set up paths through the network optimized for speed, fewest number of hops or reduced latency, among other characteristics. Using OpenFlow takes control of how traffic flows through the network out of the hands of the infrastructure, the switches and routers, and puts it in the hands of the network owner (such as a corporation), individual users or individual applications."). |

| No. | '111 Patent Claim 27 | Shieh '088 |
|---|---|---|
| 27 | The method according to claim 1, wherein the network node comprises a router, a switch, or a bridge. | Shieh '088 discloses the method according to claim 1, wherein the network node comprises a router, a switch, or a bridge.<br><br>For example, Shieh '088 discloses network access devices may be a router or a gateway, a switch or an access point.<br><br>*See supra* at Claim 1.<br><br>Shieh '088 ¶ [0020] ("FIG. 1 is a block diagram illustrating an example of network configuration according to one embodiment of the invention. Referring to FIG. 1, network access device 204, which may be a router or gateway, a switch or an access point, etc., provides an interface between network 203 and network 205. Network 203 may be an external network such as a wide area network (WAN) (e.g., Internet) while network 205 represents a local area network (LAN). Nodes 206-207 go through gateway device 204 in order to reach nodes 201-202, or vice versa. Any of nodes 201-202 and 206-207 may be a client device (e.g., a desktop, laptop, Smartphone, gaming device) or a server."). |

| No. | '111 Patent Claim 28 | Shieh '088 |
|---|---|---|
| 28 | The method according to claim 1, wherein the packet network is an Internet Protocol (IP) network, and the packet is an IP packet. | Shieh '088 discloses the method according to claim 1, wherein the packet network is an Internet Protocol (IP) network, and the packet is an IP packet.<br><br>For example, Shieh '088 discloses the use of IP addresses to identify packets being transmitted across the packet network.<br><br>*See supra* at Claim 1. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 870 of 1100

| No. | '111 Patent Claim 28 | Shieh '088 |
|---|---|---|
| | | Shieh '088 ¶ [0027] ("Referring back to FIG. 2A, in one embodiment, each of the network access devices 204A-204C maintains a flow table or session table (e.g., flow tables 251A-251C) and a firewall module (e.g., 209A-209C). A network flow refers to a sequence of packets from a source computer to a destination, which may be another host, a multicast group, or a broadcast domain. For example, a TCP/IP flow can be uniquely identified by the following parameters within a certain time period: 1) Source and Destination IP address; 2) Source and Destination Port; and 3) Layer 4 Protocol (TCP/UDP/ICMP). A session is a semi-permanent interactive information interchange, also known as a dialogue, a conversation or a meeting, between two or more communicating devices. A session is set up or established at a certain point in time and torn down at a later point in time. An established communication session may involve more than one message in each direction. A session is typically, but not always, stateful, meaning that at least one of the communicating entities needs to save information about the session history in order to be able to communicate, as opposed to stateless communication, where the communication consists of independent requests with responses. Flow tables 251A-251C may be implemented as a combination of a flow table and a session table."). |
| | | Shieh '088 ¶ [0031] ("According to one embodiment, referring back to FIG. 2A, when a security-processing function (e.g., processing node 211A) receives the packets, it does the security inspection and security policy enforcement. The packets then are forwarded to the next I/O function (e.g., modules 209A-209C). The choices of the next I/O function could be from the decision from layer 2 such as Ethernet MAC address lookup, or IP address routing, or other methods."). |
| | | Shieh '088 ¶ [0038] (" FIG. 3 is a block diagram illustrating an example of a distributed firewall according to one embodiment of the invention. Referring to FIG. 3, distributed firewall 212 includes, for the purpose of illustration, four different types of modules: virtual I/O modules 301-304, security processing modules 309-311, service processing modules 312-313, and central controller 208. All these modules can run on the same virtual machine, or on different virtual machines, or on same or different physical hosts. In one embodiment, the communication protocol between the modules is IPC (inter-process communication) if they run on the same memory space, use layer-2 network protocol if they are on the same layer-2 network, or use IP protocols if they are connected through IP |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 871 of 1100

| No. | '111 Patent Claim 28 | Shieh '088 |
|---|---|---|
| | | networks. Some or all of modules 301-304 and 309-313 may be executed within a virtual machine. Dependent upon the specific configuration, each of modules 301-304 and 309-313 may be executed by a respective virtual machine. In other configurations, multiple of modules 301-304 and 309-313 may be executed by the same virtual machine."). |

| No. | '111 Patent Claim 29 | Shieh '088 |
|---|---|---|
| 29 | The method according to claim 28, wherein the packet network is an Transmission Control Protocol (TCP) network, and the packet is an TCP packet. | Shieh '088 discloses the method according to claim 28, wherein the packet network is an Transmission Control Protocol (TCP) network, and the packet is an TCP packet.<br><br>For example, Shieh '088 discloses that the packet streams passing through its system include TCP packets. Shieh '088 further discloses the transportation of TCP packets across a TCP network using TCP protocol.<br><br>*See supra* Claim 28.<br><br>Shieh '088 ¶ [0027] ("Referring back to FIG. 2A, in one embodiment, each of the network access devices 204A-204C maintains a flow table or session table (e.g., flow tables 251A-251C) and a firewall module (e.g., 209A-209C). A network flow refers to a sequence of packets from a source computer to a destination, which may be another host, a multicast group, or a broadcast domain. For example, a TCP/IP flow can be uniquely identified by the following parameters within a certain time period: 1) Source and Destination IP address; 2) Source and Destination Port; and 3) Layer 4 Protocol (TCP/UDP/ICMP). A session is a semi-permanent interactive information interchange, also known as a dialogue, a conversation or a meeting, between two or more communicating devices. A session is set up or established at a certain point in time and torn down at a later point in time. An established communication session may involve more than one message in each direction. A session is typically, but not always, stateful, meaning that at least one of the communicating entities needs to save information about the session history in order to be able to communicate, as opposed to stateless communication, where the communication consists of independent requests with responses. Flow tables 251A-251C may be implemented as a combination of a flow table and a session table."). |


~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 872 of 1100

| No. | '111 Patent Claim 29 | Shieh '088 |
|---|---|---|
| | | Shieh '088 ¶ [0036] ("During the bypass phase, the I/O function may notify the security-processing function if there are special events in the packet stream. These events could be receipt of TCP FIN or TCP RST packets, or not receiving any packets of the connection within a time threshold. The notification from I/O functions to security processing functions could help to clean up the state in the security-processing nodes."). |

| No. | '111 Patent Claim 30 | Shieh '088 |
|---|---|---|
| 30[a] | The method according to claim 1, further comprising: receiving, by the network node from the first entity over the packet network, one or more additional packets; | Shieh '088 discloses the method according to claim 1, further comprising: receiving, by the network node from the first entity over the packet network, one or more additional packets.<br><br>For example, Shieh '088 discloses that its method can be applied to subsequent packets of a particular packet flow or session.<br><br>*See also* Claim 1.<br><br>Shieh '088 ¶ [0018] ("According to one embodiment, an administrator can configure, for example, via a controller or a management entity, a network access device to set up a set of filtering rules specifying whether and/or what types of packets should be forwarded to a security device and which of the security devices for security inspection. In this embodiment, the controller is configured to manage multiple network access devices and/or multiple security devices. Alternatively, a security device may inform a network access device that subsequent packets of a particular session should be forwarded from the network access device for security inspection. In one embodiment, a security device performs the security inspection at the beginning of the flow or session, and at a certain point, the security device decides that it no longer needs to inspect further packets of the same session.).<br><br>Shieh '088 ¶ [0037] ("FIG. 2B is a processing flow diagram illustrating a process of security inspection according to one embodiment of the invention. Referring to FIG. 2B, as an example, network switch 272 may represent any of network access devices 204A-204C and security device 273 may represents any of security processing devices 211A-211B as described above with respect to FIG. 2A. When device 272 receives a packet from a source node 271 via transaction 281, device 272 may determine whether the packet should be forwarded to security device 273. For example, device 272 may look up in its session table |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 873 of 1100

| No. | '111 Patent Claim 30 | Shieh '088 |
|---|---|---|
| | | such as the one as shown in FIG. 5 to determine whether a bypass flag has been set to a predetermined value. If the bypass flag matches the predetermined value, the packet is forwarded to security device 273 via path 282; otherwise, the packet is routed to destination node 274. Alternatively, if there is no entry in the session table corresponding to the current session, the packet will also be transmitted to security device 273. After network device 272 receives a response from security device 273 via path 283, dependent upon the response, the packet may then be routed to destination node 274 via path 284. These processes may continue until a notification is received from security device 273 via path 285 indicating that it no longer wishes to receive further packets of the same session for inspection, such that subsequent packets will be directly routed to destination node 274 via path 286 without routing to security device 273. If there are certain events that have been registered from security device 273, network device 272 may notify security device 274 via path 287 upon detecting the registered events."). |
| 30[b] | checking, by the network node, if any one of the one or more additional packets satisfies the criterion; | Shieh '088 discloses checking, by the network node, if any one of the one or more additional packets satisfies the criterion.<br><br>For example, Shieh '088 discloses classifying each incoming packet, not limited to a single packet, to see if it satisfies the criterion in the forwarding table *i.e.*, checking, by the network node, if any one of the one or more additional packets satisfies the criterion.<br><br>*See also* Claim 1[d].<br><br>Shieh '088 ¶ [0039] "(An I/O module running within a virtual machine is referred to herein as a virtual I/O module. Each of virtual I/O modules 301-304 receives packets from any of servers 321-324 of LAN 320 and sends packets to external network 315 outside of the firewall. In one embodiment, each of I/O modules 301-304 keeps a local cache (e.g., caches 305-308) storing location(s) of a security processing module(s) (e.g., security processing modules 309-311) for each connection session. A cache maintained by each I/O module contains a forwarding table mapping certain connection sessions to any of security modules 309-311. An example of a forwarding table is shown in FIG. 5. Upon receiving a packet, an I/O module performs a packet classification to find out the associated connection and forwards the packet to the corresponding security processing module identified by the forwarding table. If it cannot find the connection in its local cache, the packets are forwarded |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 874 of 1100

| No. | '111 Patent Claim 30 | Shieh '088 |
|---|---|---|
| | | to central controller 208 for processing. In such a case, controller 208 assigns the connection to one of security processing modules 309-311 based on one or more of a variety of factors such as load balancing. The virtual I/O modules 302-304 can be located at multiple locations of the networks to receive and send out packets."). |
| 30[c] | responsive to an additional packet not satisfying the criterion, sending, by the network node over the packet network, the additional packet to the second entity; and | Shieh '088 discloses responsive to an additional packet not satisfying the criterion, sending, by the network node over the packet network, the additional packet to the second entity.<br><br>Shieh '088 discloses transmitting each packet to the destination node without forwarding the packet to the security device, if the data member does not match the predetermined value, where the data member is inspected respective to each packet received, *i.e.*, responsive to an additional packet not satisfying the criterion, sending, by the network node over the packet network, the additional packet to the second entity.<br><br>*See supra* Claim 30[b].<br><br>*See also* Claim 1[e]. |
| 30[d] | responsive to the additional packet satisfying the criterion, sending the additional packet, by the network node over the packet network, in response to the instruction. | Shieh '088 discloses responsive to the additional packet satisfying the criterion, sending the additional packet, by the network node over the packet network, in response to the instruction.<br><br>For example, Shieh '088 discloses an embodiment in which a "virtual I/O module" functions as a network nodes that receives packets from servers and sends packets to an external network, depending on existing connections in a local cache, *i.e.*, responsive to the additional packet satisfying the criterion, sending the additional packet, by the network node over the packet network, in response to the instruction.<br><br>*See supra* Claim 30[b].<br><br>*See also* Claim 1[f]. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 875 of 1100

| No. | '111 Patent Claim 31 | Shieh '088 |
|---|---|---|
| 31[a] | The method according to claim 1, wherein the packet network is a Software Defined Network (SDN), | Shieh '088 discloses the method according to claim 1, wherein the packet network is a Software Defined Network (SDN).<br><br>For example, Shieh '088 discloses that it uses an OpenFlow protocol (an SDN networking standard) in which a packet is routed as part of the data plane.<br><br>Shieh '088 ¶ [0025] ("According to one embodiment, each of network access devices 204A-204C maintains a persistent connection such as secure connections or tunnels 260 with a controller or management entity 208 for exchanging management messages and configurations, or distributing routing information to network access devices 204A-204C, etc. In one embodiment, controller 208 communicates with each of the network access devices 204A-204C using a management protocol such as the OpenFlow™ protocol. OpenFlow is a Layer 2 communications protocol (e.g., media access control or MAC layer) that gives access to the forwarding plane of a network switch or router over the network. In simpler terms, OpenFlow allows the path of network packets through the network of switches to be determined by software running on multiple routers (minimum two of them, primary and secondary, having a role of observers). This separation of the control from the forwarding allows for more sophisticated traffic management than is feasible using access control lists (ACLs) and routing protocols."). |
| 31[b] | the packet is routed as part of a data plane and | Shieh '088 discloses that the packet is routed as part of a data plane.<br><br>For example, Shieh '088 discloses that it uses an Openflow protocol (an SDN networking standard) in which a packet is routed as part of the data plane.<br><br>Shieh '088 ¶ [0025] ("According to one embodiment, each of network access devices 204A-204C maintains a persistent connection such as secure connections or tunnels 260 with a controller or management entity 208 for exchanging management messages and configurations, or distributing routing information to network access devices 204A-204C, etc. In one embodiment, controller 208 communicates with each of the network access devices 204A-204C using a management protocol such as the OpenFlow™ protocol. OpenFlow is a Layer 2 communications protocol (e.g., media access control or MAC layer) that gives access to the forwarding plane of a network switch or router over the network. In simpler terms, OpenFlow allows the path of network packets through the network of switches |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 876 of 1100

| No. | '111 Patent Claim 31 | Shieh '088 |
|---|---|---|
| | | to be determined by software running on multiple routers (minimum two of them, primary and secondary, having a role of observers). This separation of the control from the forwarding allows for more sophisticated traffic management than is feasible using access control lists (ACLs) and routing protocols."). |
| 31[c] | the network node communication with the controller serves as a control plane. | Shieh '088 discloses that the network node communication with the controller serves as a control plane.<br><br>For example, Shieh '088 discloses that the controller communicates with the switches via the Openflow protocol.<br><br>Shieh '088 ¶ [0025] ("According to one embodiment, each of network access devices 204A-204C maintains a persistent connection such as secure connections or tunnels 260 with a controller or management entity 208 for exchanging management messages and configurations, or distributing routing information to network access devices 204A-204C, etc. In one embodiment, controller 208 communicates with each of the network access devices 204A-204C using a management protocol such as the OpenFlow™ protocol. OpenFlow is a Layer 2 communications protocol (e.g., media access control or MAC layer) that gives access to the forwarding plane of a network switch or router over the network. In simpler terms, OpenFlow allows the path of network packets through the network of switches to be determined by software running on multiple routers (minimum two of them, primary and secondary, having a role of observers). This separation of the control from the forwarding allows for more sophisticated traffic management than is feasible using access control lists (ACLs) and routing protocols.").<br><br>Shieh '088 ¶ [0026] ("The OpenFlow technology consists of three parts: flow tables installed on switches, a controller, and an OpenFlow protocol for the controller to talk securely with switches. Flow tables are set up on switches or routers. Controllers talk to the switches via the OpenFlow Protocol, which is secure, and impose policies on flows. For example, a simple flow might be defined as any traffic from a given IP address. The rule governing it might be |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 877 of 1100

| No. | '111 Patent Claim 31 | Shieh '088 |
|---|---|---|
| | | to route the flow through a given switch port. With its knowledge of the network, the controller could set up paths through the network optimized for speed, fewest number of hops or reduced latency, among other characteristics. Using OpenFlow takes control of how traffic flows through the network out of the hands of the infrastructure, the switches and routers, and puts it in the hands of the network owner (such as a corporation), individual users or individual applications."). |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 878 of 1100

**EXHIBIT D-6**
Defendant's Preliminary Invalidity Contentions
*Orckit Corporation v. Cisco Systems, Inc.*, 2:22-cv-00276-JRG-RSP

---

**Chart for U.S. Patent 10,652,111 ("the '111 Patent")**
**Cisco Intelligent WAN ("Cisco IWAN System")**

As shown in the chart below, all Asserted Claims of the '111 Patent are invalid under (1) AIA-35 U.S.C. § 102 (a) because Cisco IWAN System meets each element of those claims, (2) invalid under AIA-35 U.S.C. § 102 (a) because the references describing the Cisco IWAN System disclose every limitation of every Asserted Claim, and/or (3) 35 U.S.C. § 103 because Cisco IWAN System renders those claims obvious either alone, or in combination with the knowledge of a person having ordinary skill in the art, and in further combination with the references specifically identified below and in the following claim chart and/or one or more references identified in Defendant's Preliminary Invalidity Contentions. The Cisco IWAN System comprises various Cisco switches and routers from before April 22, 2014 that implemented Cisco's IWAN feature. The following quotations and diagrams come from documentation describing Cisco IWAN System and its functionalities that were published prior to April 22, 2014.

- https://www.cisco.com/c/dam/global/hr_hr/assets/ciscoconnect/2014/pdfs/cc_see_2014_iwan_next_generation_branch.pdf ("IWAN Next Gen")
- Cisco Performance Routing (PfR) Solution Guides ("PfR Solution Guides")
- Cisco Intelligent WAN (IWAN): Right-Size Your Network without Compromise ("IWAN Right Size")
- Cisco presentation titled "Cisco Next Generation Branch Architecture", dated November 2013 ("Cisco Next Generation")
- Cisco presentation titled "Cisco Intelligent WAN (IWAN)", dated November 2013 ("Cisco IWAN")
- Cisco presentation titled "Cisco Intelligent WAN (IWAN) – Uncompromised Experience over Any Link", dated November 2013 ("Cisco IWAN – Uncompromised Experience")
- Cisco document titled, "Cisco Network Architecture Discovery, Planning, Design and Implementation Services for Intelligent WAN" ("IWAN At-A-Glance")
- https://www.youtube.com/watch?v=GQuRzr__N-c ("DMVPN QoS for Intelligent WAN")
- https://www.youtube.com/watch?v=XFsqTENxopo ("2014 March Webinar LiveAction IWAN Management")
- https://www.youtube.com/watch?v=8mWSXKIz2hk ("IWAN Management Technical Presentation and Demo")
- https://web.archive.org/web/20140226054405/http://www.cisco.com/c/en/us/solutions/enterprise-networks/intelligent-wan/index.html ("Cisco Intelligent WAN")


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 879 of 1100

- https://web.archive.org/web/20140813171810/http://www.cisco.com/c/dam/en/us/solutions/collateral/enterprise-networks/intelligent-wan/white-paper-c11-729752.pdf ("Cisco IWAN and Akamai Intelligent Platform™: Maximize Your
- WAN Investment")
- https://web.archive.org/web/20140225095408/http://www.cisco.com/c/dam/en/us/solutions/collateral/borderless-networks/application-experience/connect_to_the_cloud.pdf ("Cisco Application Services Platform")

Motivations to combine the disclosures in Cisco IWAN System with disclosures in other publications known in the art, as explained in this chart, include at least the similarity in subject matter between the references to the extent they concern methods relating to routing certain network traffic to entities for further analysis and inspection. Insofar as the references cite other patents or publications, or suggest additional changes, one of ordinary skill in the art would look beyond a single reference to other references in the field.

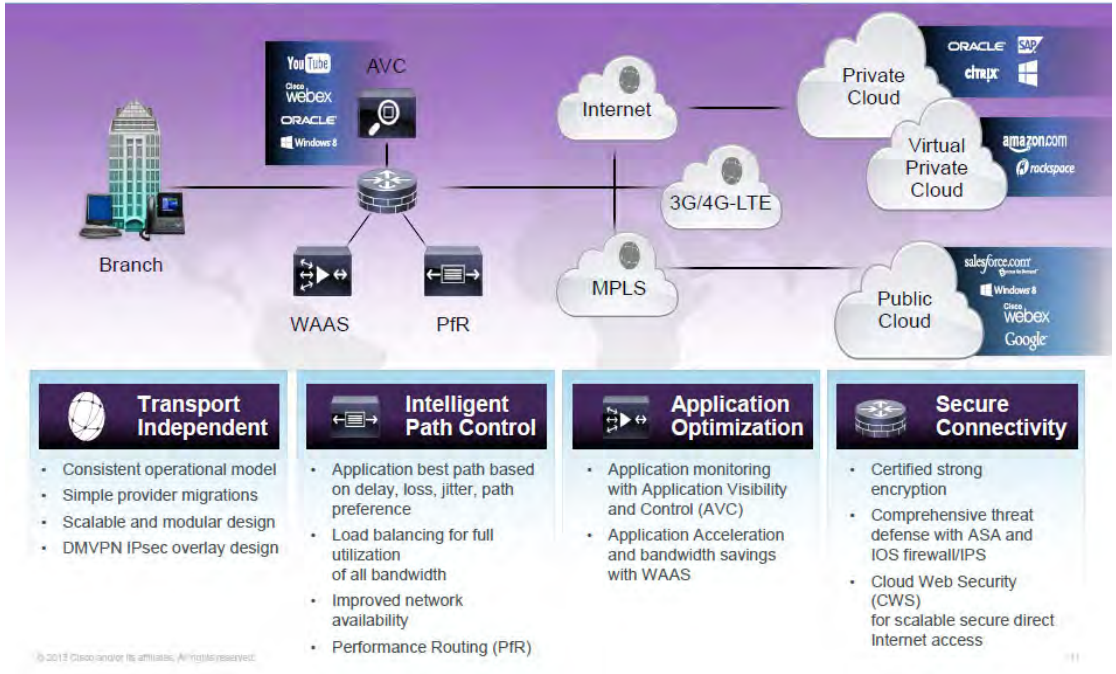
These invalidity contentions are based on Defendant's present understanding of the Asserted Claims, and Orckit's apparent construction of the claims in its November 3, 2022 Disclosure of Asserted Claims and Infringement Contentions Pursuant to P.R. 3-1, and Orckit's January 19, 2023 First Amended Disclosure of Asserted Claims and Infringement Contentions Pursuant to P.R. 3-1 (Orckit's "Infringement Disclosures"), which is deficient at least insofar as it fails to cite any documents or identify accused structures, acts, or materials in the Accused Products with particularity. Defendant does not agree with Orckit's application of the claims, or that the claims satisfy the requirements of 35 U.S.C. § 112. Defendant's contentions herein are not, and should in no way be seen as, admissions or adoptions as to any particular claim scope or construction, or as any admission that any particular element is met by any accused product in any particular way. Defendant objects to any attempt to imply claim construction from this chart. Defendant's prior art invalidity contentions are made in a variety of alternatives and do not represent Defendant's agreement or view as to the meaning, definiteness, written description support for, or enablement of any claim contained therein.

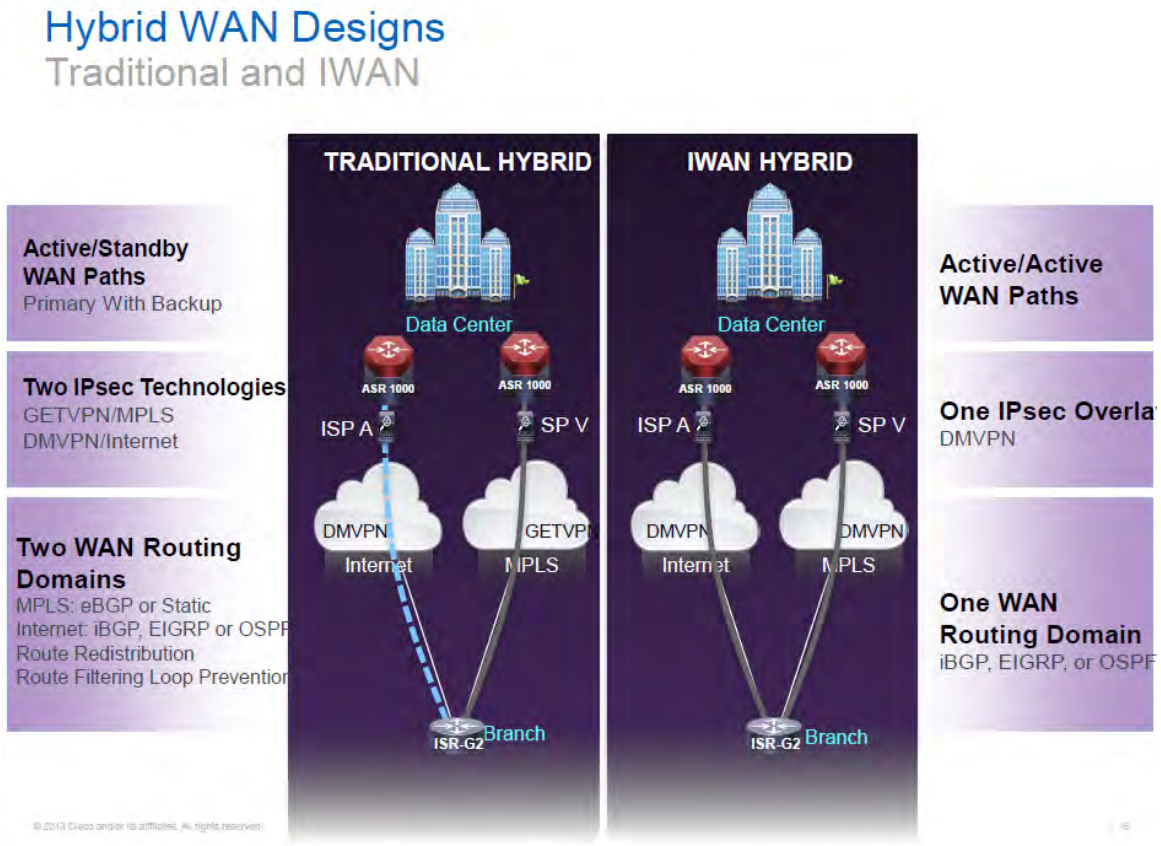
The following contentions are subject to revision and amendment pursuant to Federal Rule of Civil Procedure 26(e), the Local Rules, and the Orders of record in this matter subject to further investigation and discovery regarding the prior art and the Court's construction of the claims at issue.

| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| 1[preamble] | A method for use with a packet network including a network node for transporting packets between first and second entities | Cisco IWAN System discloses a method for use with a packet network including a network node for transporting packets between first and second entities under control of a controller that is external to the network node, the method comprising.<br><br>For example, Cisco IWAN System discloses a communication method utilizing branch devices/border routers that transport data packets between user devices and/or data center |



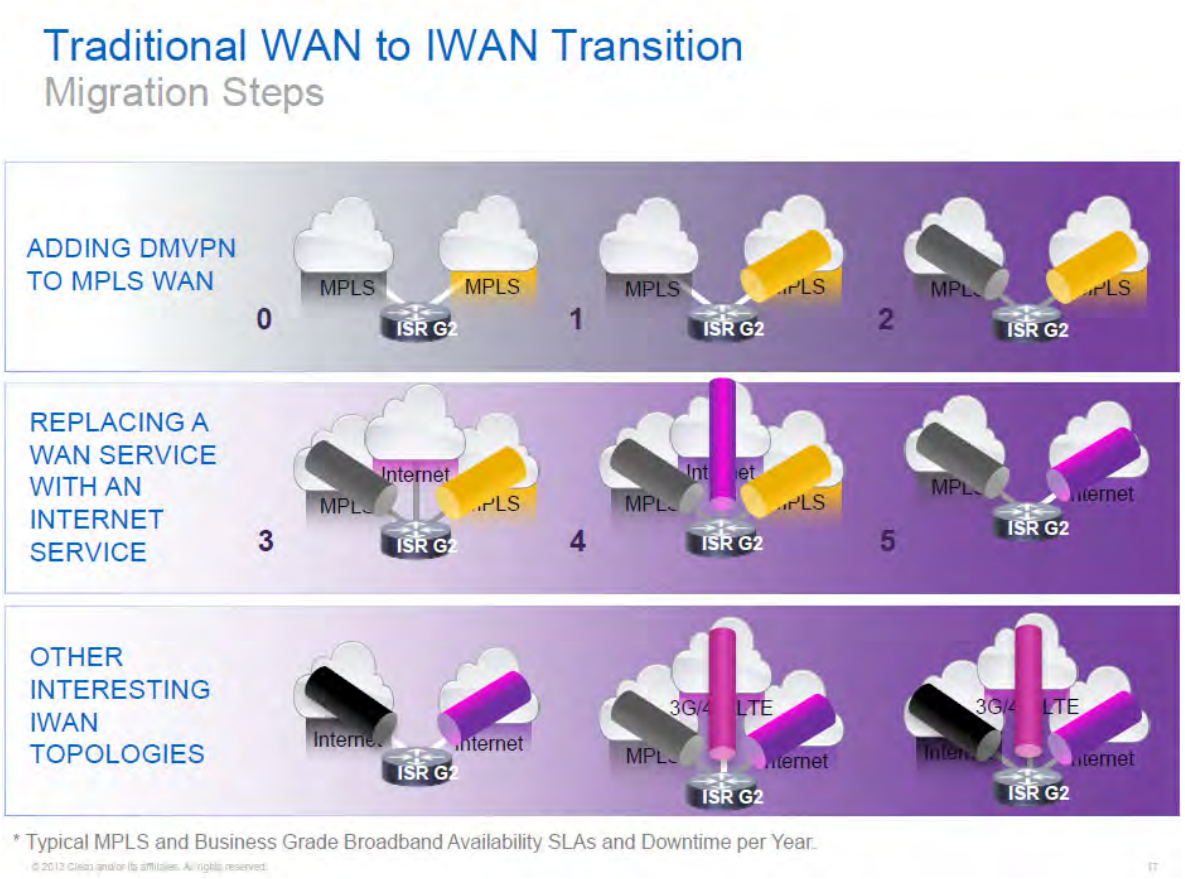
Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 880 of 1100

| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | under control of a controller that is external to the network node, the method comprising: | devices. Cisco IWAN System further discloses communication within a network using border routers under the control of an external hub/master controller. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met.<br><br>IWAN Next Gen at 11<br><br>Intelligent WAN Solution Components<br><br>  <br><br><br>IWAN Next Gen at 16 |

| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | |  <br>IWAN Next Gen at 17 |

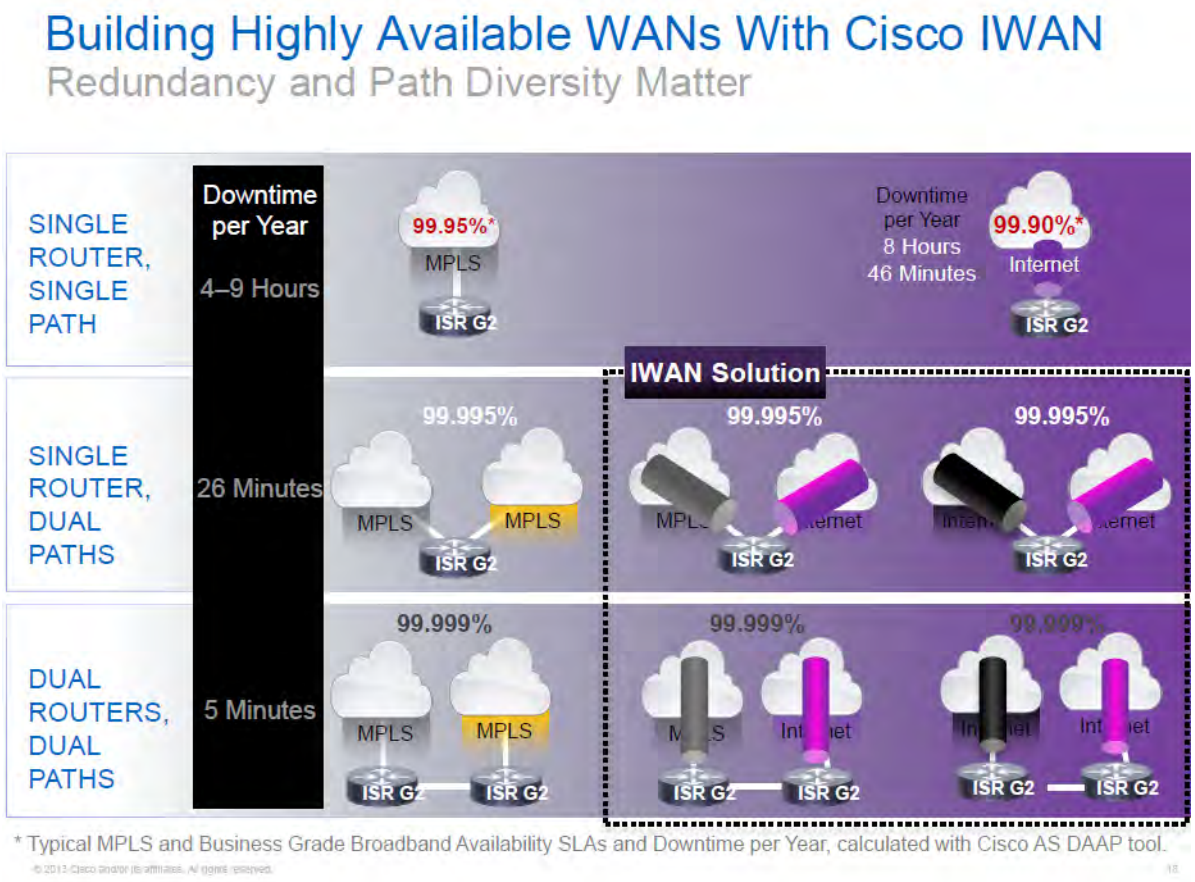


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 882 of 1100

| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | |  <br>IWAN Next Gen at 18 |



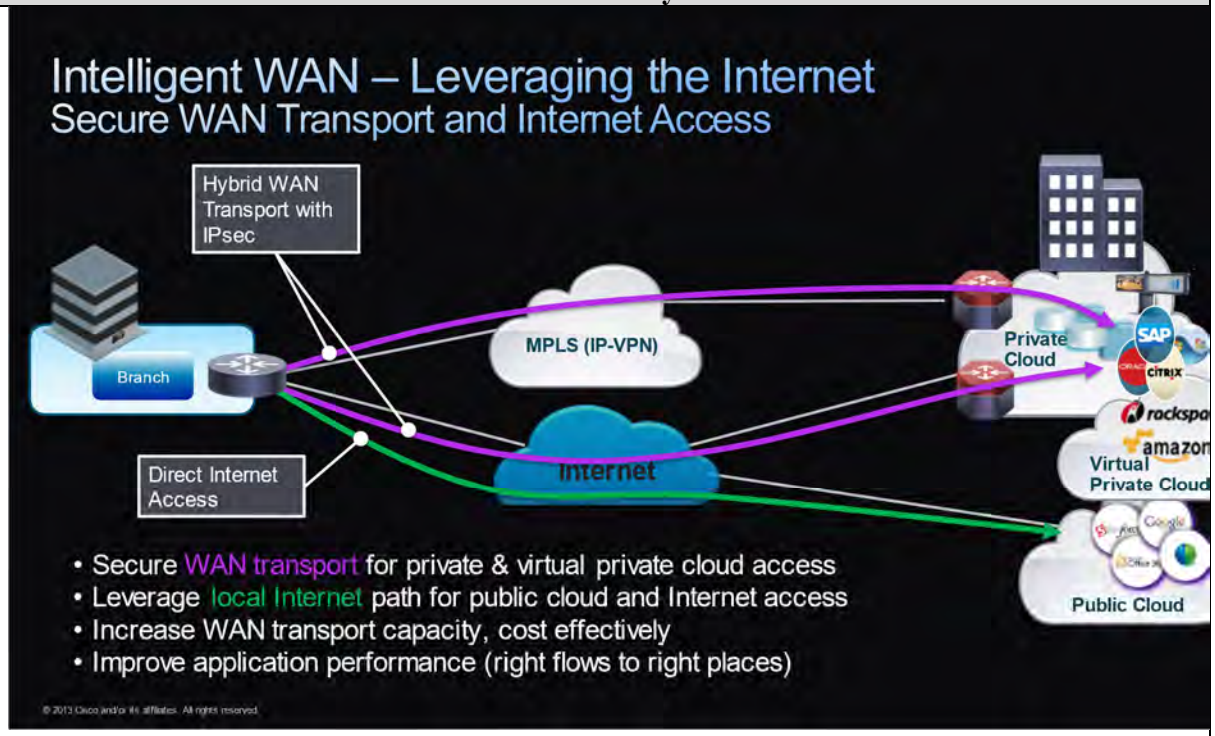
Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 883 of 1100

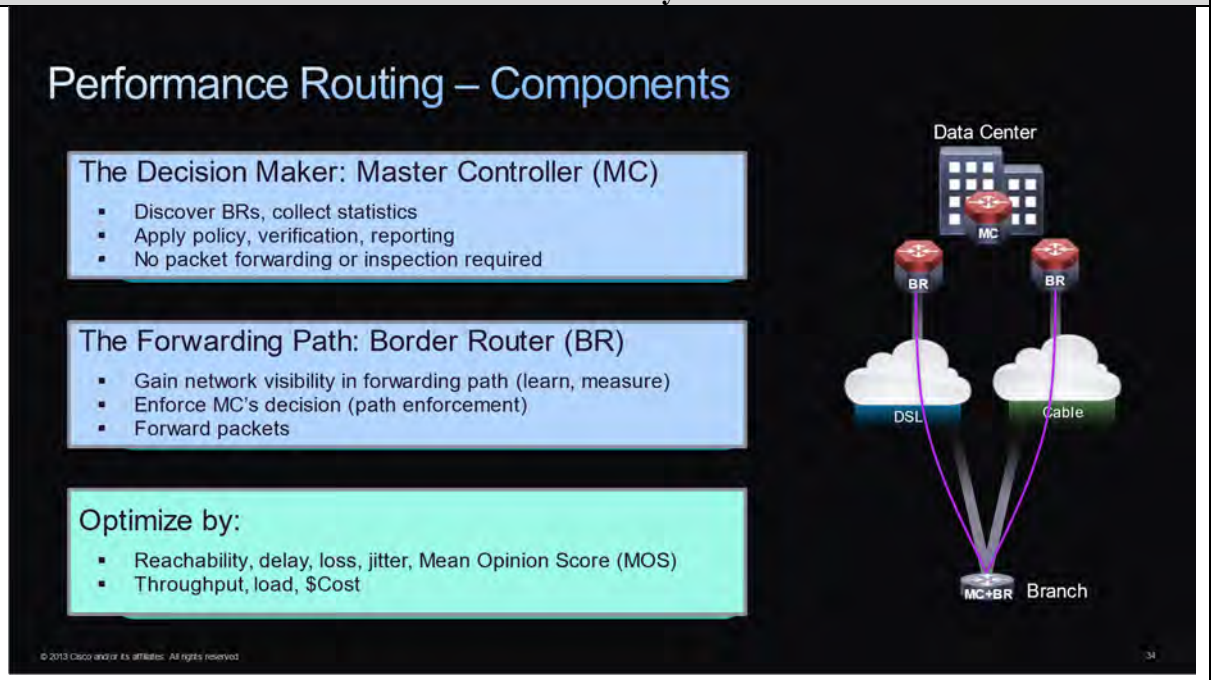
| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | |  <br>Cisco Next Generation |

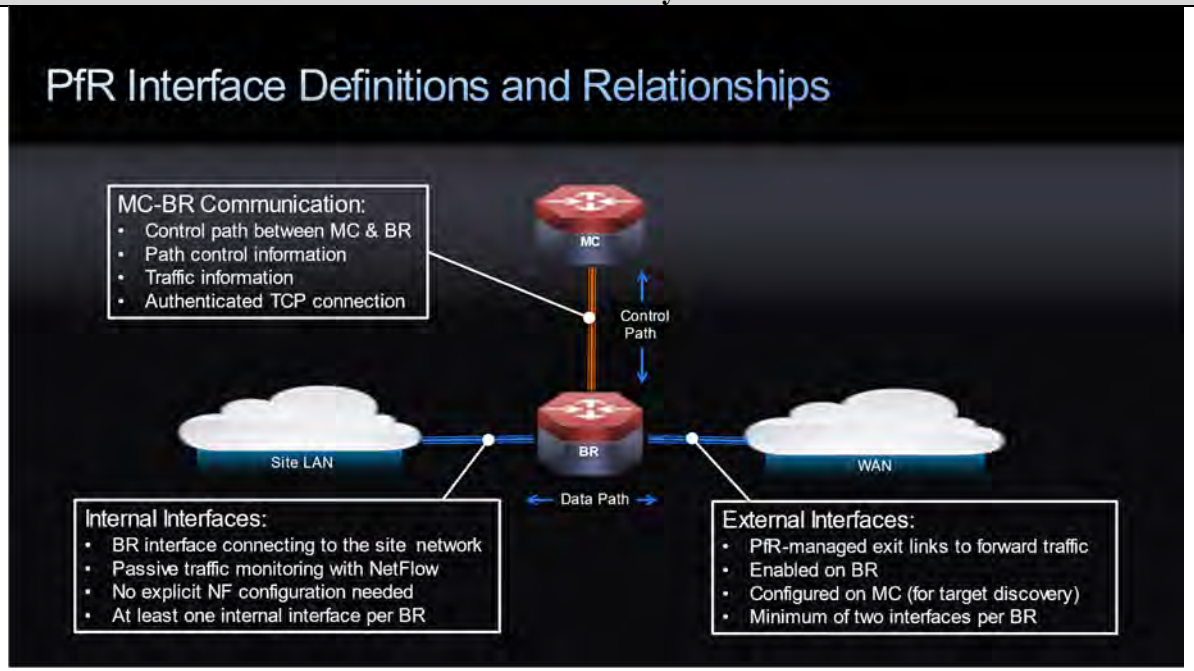
| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | |  <br>Cisco IWAN |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 885 of 1100

| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | |  <br>Cisco IWAN - Uncompromised Experience |

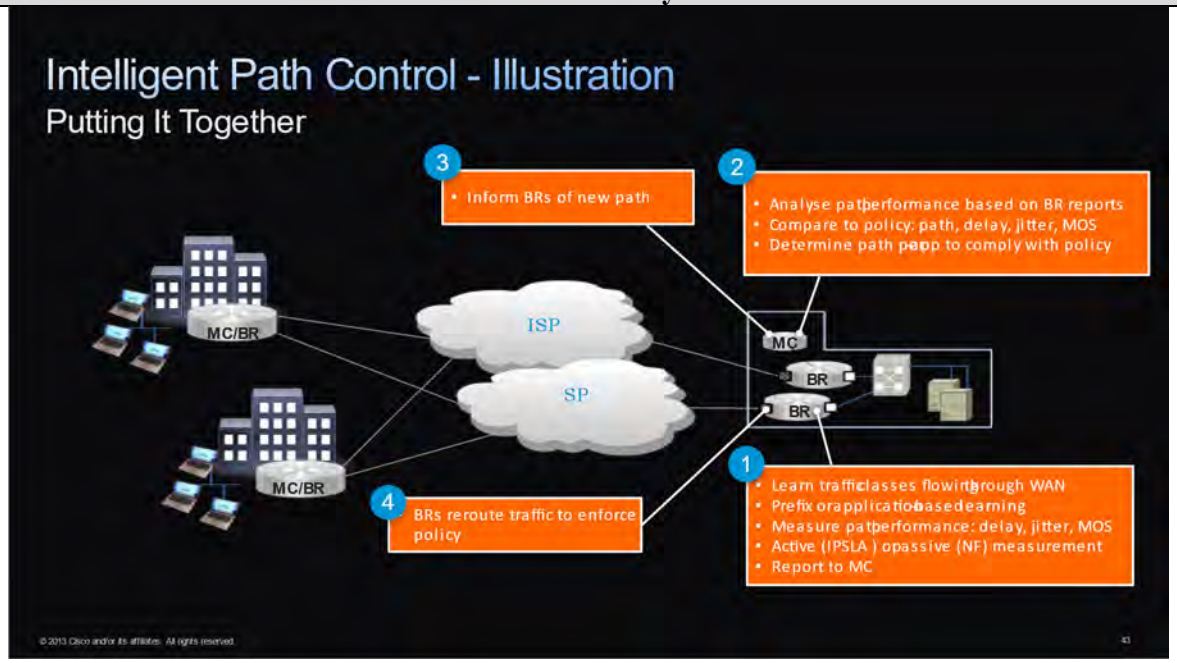
| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | |  |

| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | |  |



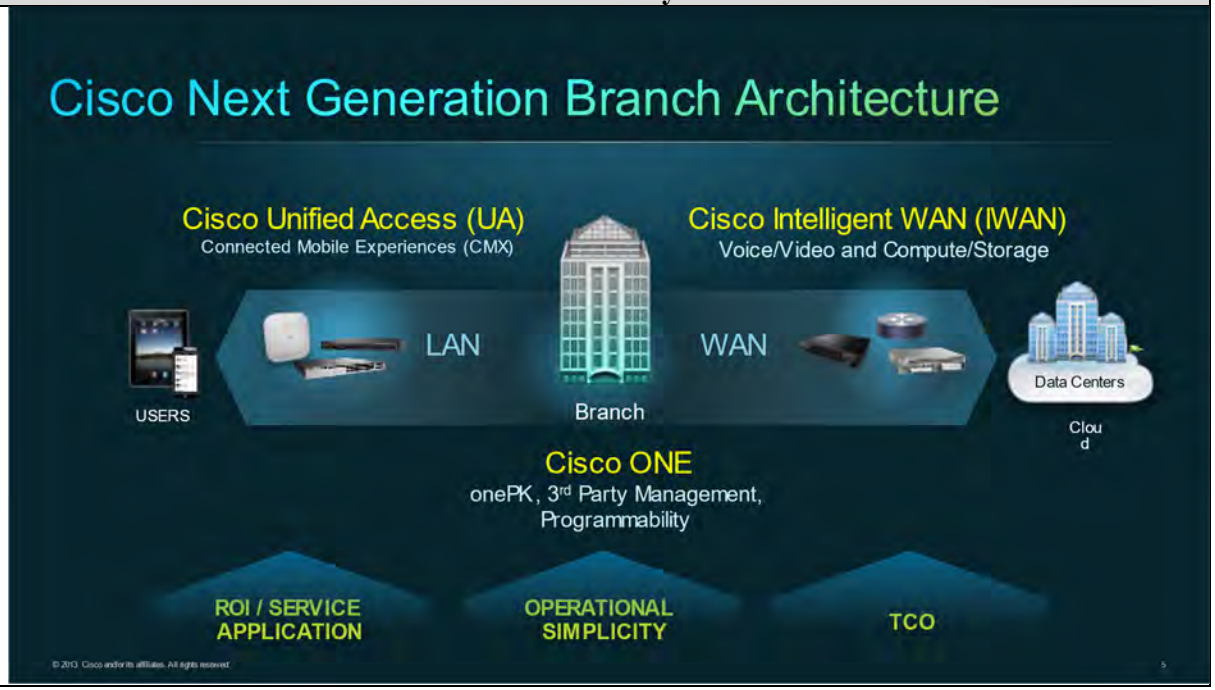


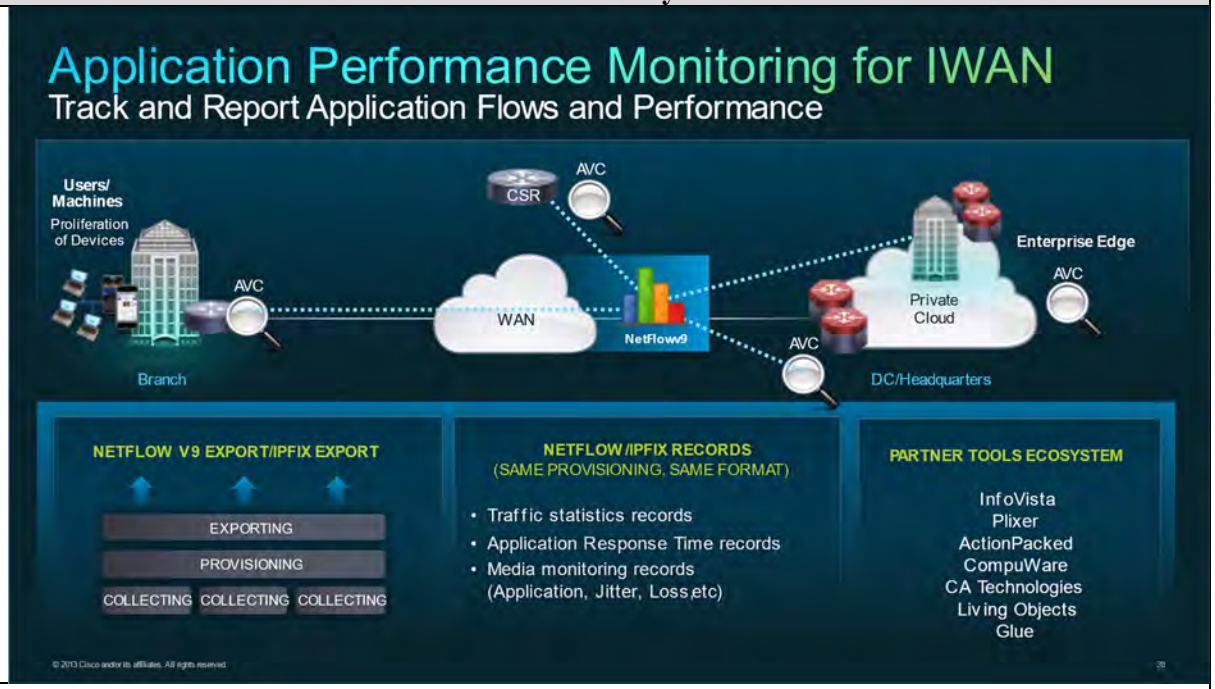
Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 888 of 1100

| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | |  <br>IWAN At-A-Glance |



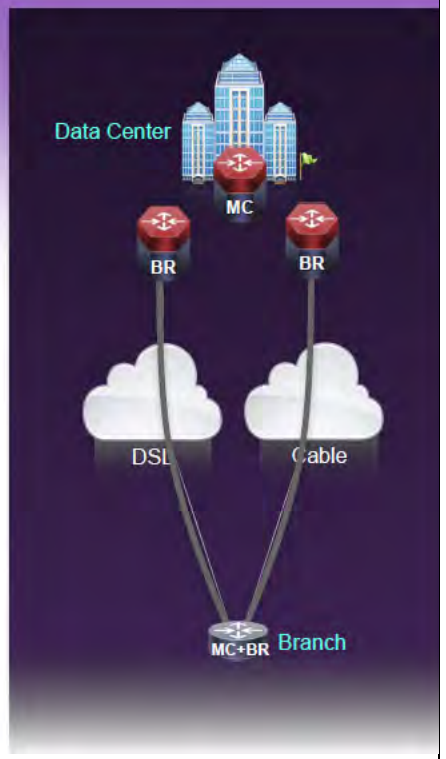
Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 889 of 1100

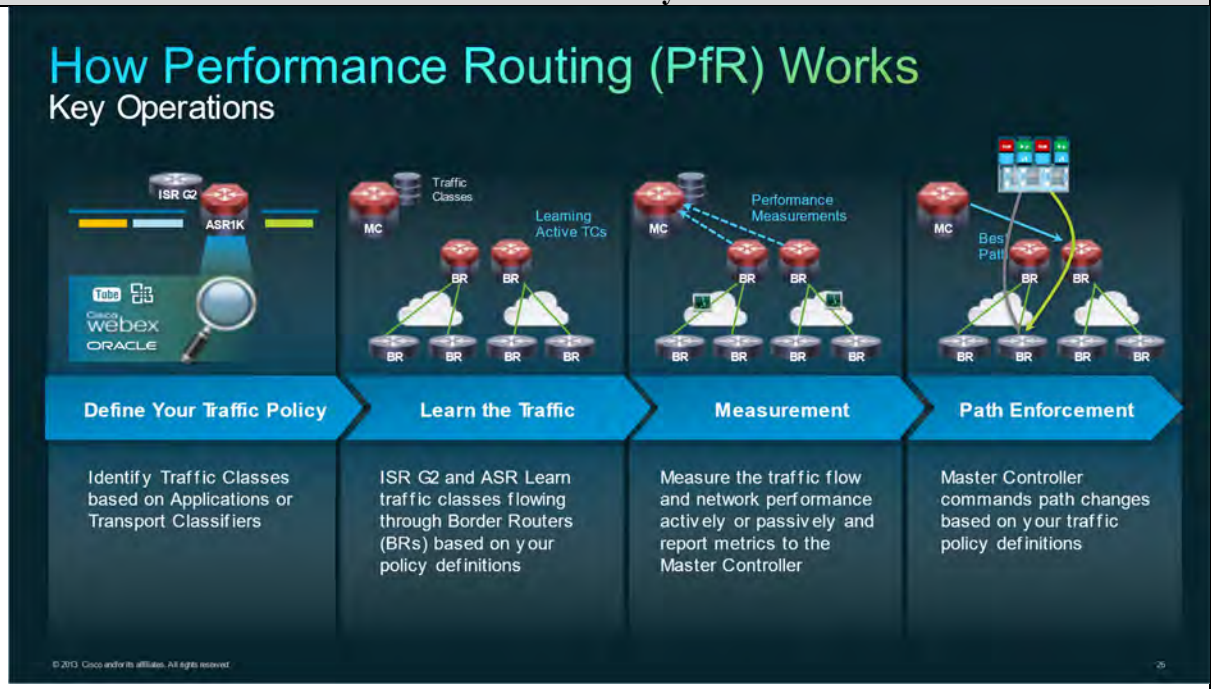
| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | |  |

| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | |  |
| 1[a] | sending, by the controller to the network node over the packet network, an instruction and a packet-applicable criterion; | Cisco IWAN System discloses sending, by the controller to the network node over the packet network, an instruction and a packet-applicable criterion.<br><br>For example, Cisco IWAN System discloses sending by the hub/master controller (MC) to the branch devices/border routers.<br><br>IWAN Next Gen at 25 |



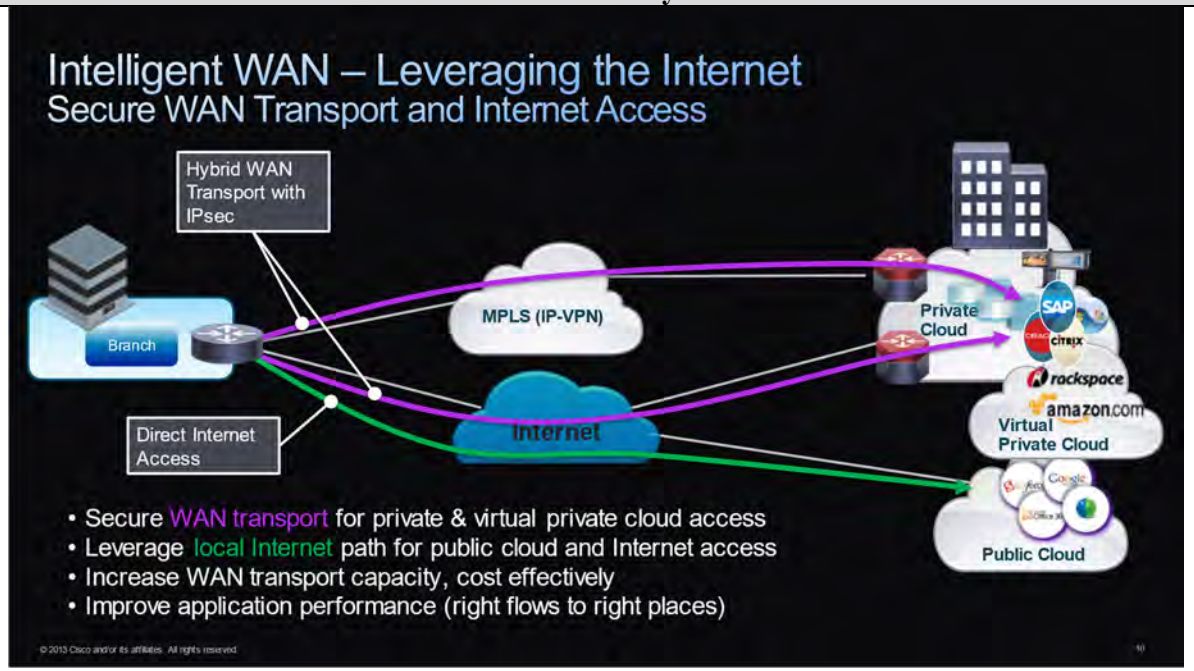
Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 891 of 1100

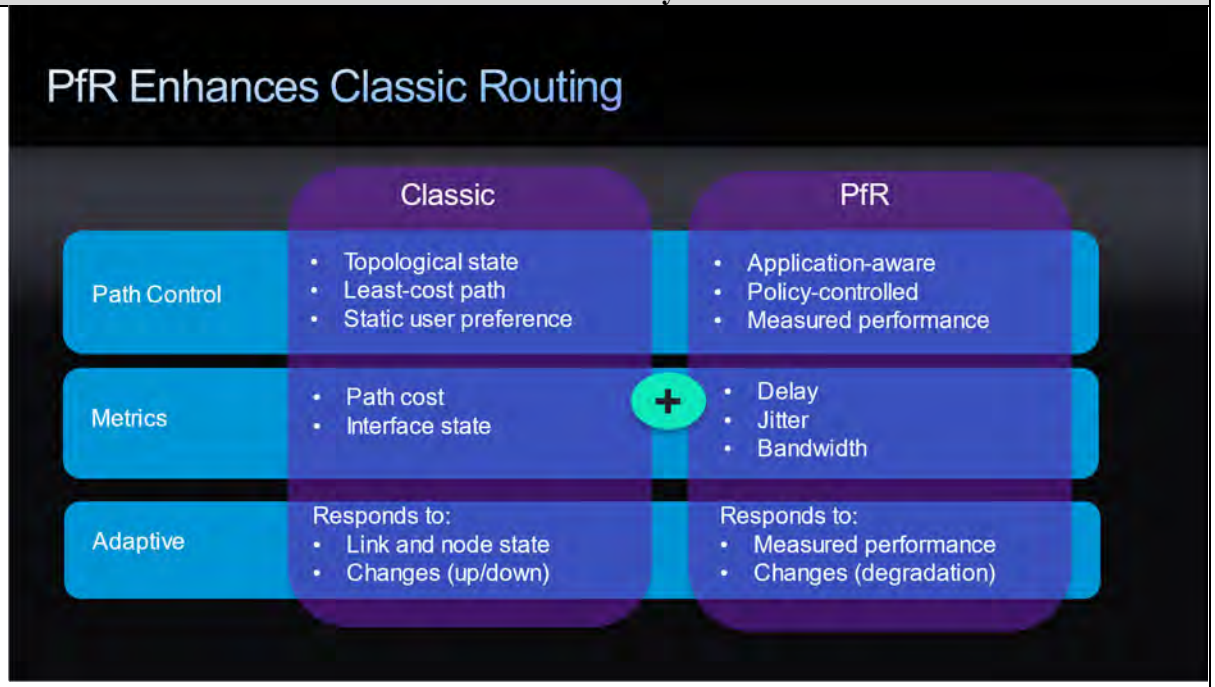
| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | | Performance Routing—Components<br>The Decision Maker: Master Controller (MC)<br>• Discover BRs, collect statistics<br>• Apply policy, verification, reporting<br>• No packet forwarding/inspection required<br>The Forwarding Path: Border Router (BR)<br>• Gain network visibility in forwarding path (Learn, measure)<br>• Enforce MC's decision (path enforcement)<br>• Does all packet forwarding<br>Optimize By:<br>• Reachability, Delay, Loss, Jitter, MOS,<br>• Throughput, Load, and/or $Cost<br>  <br>Cisco Next Generation |

| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | |  <br>Cisco IWAN |



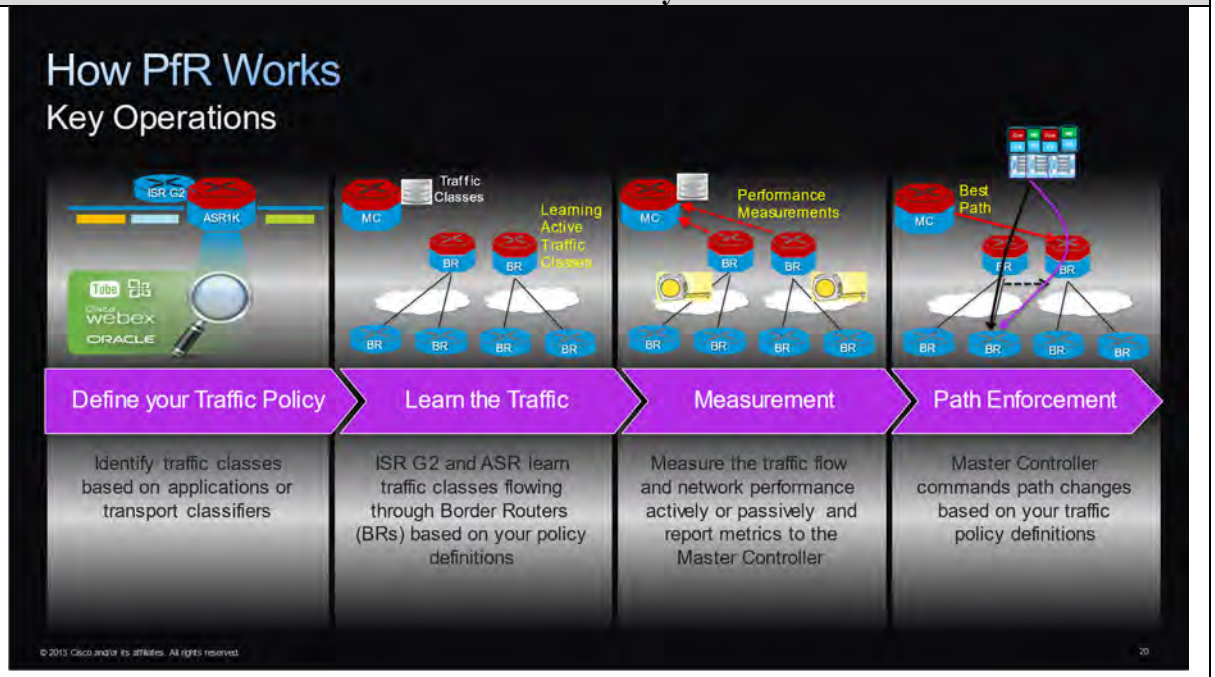
Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 893 of 1100

| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | |  <br>Cisco IWAN |

| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | |  <br>Cisco IWAN |

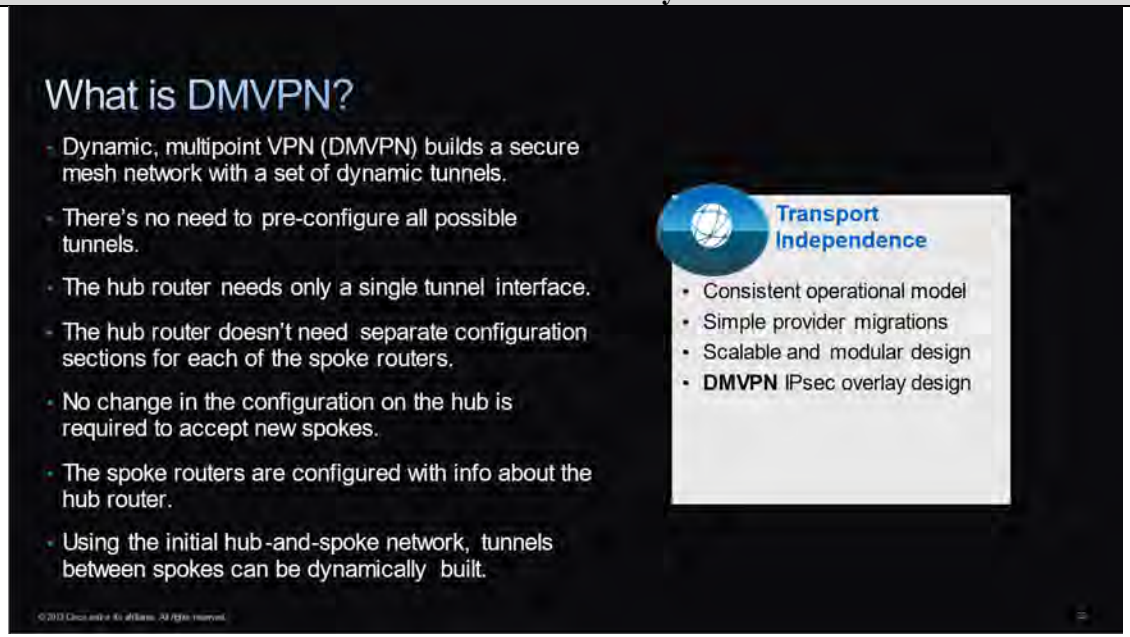


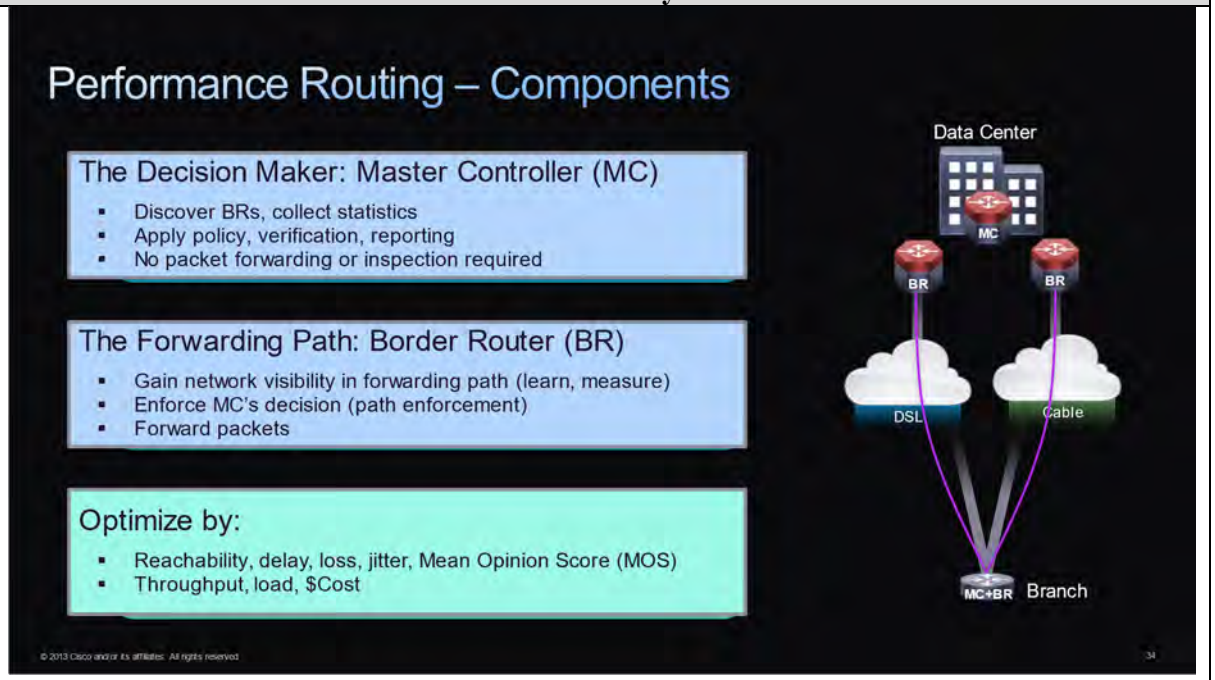
Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 895 of 1100

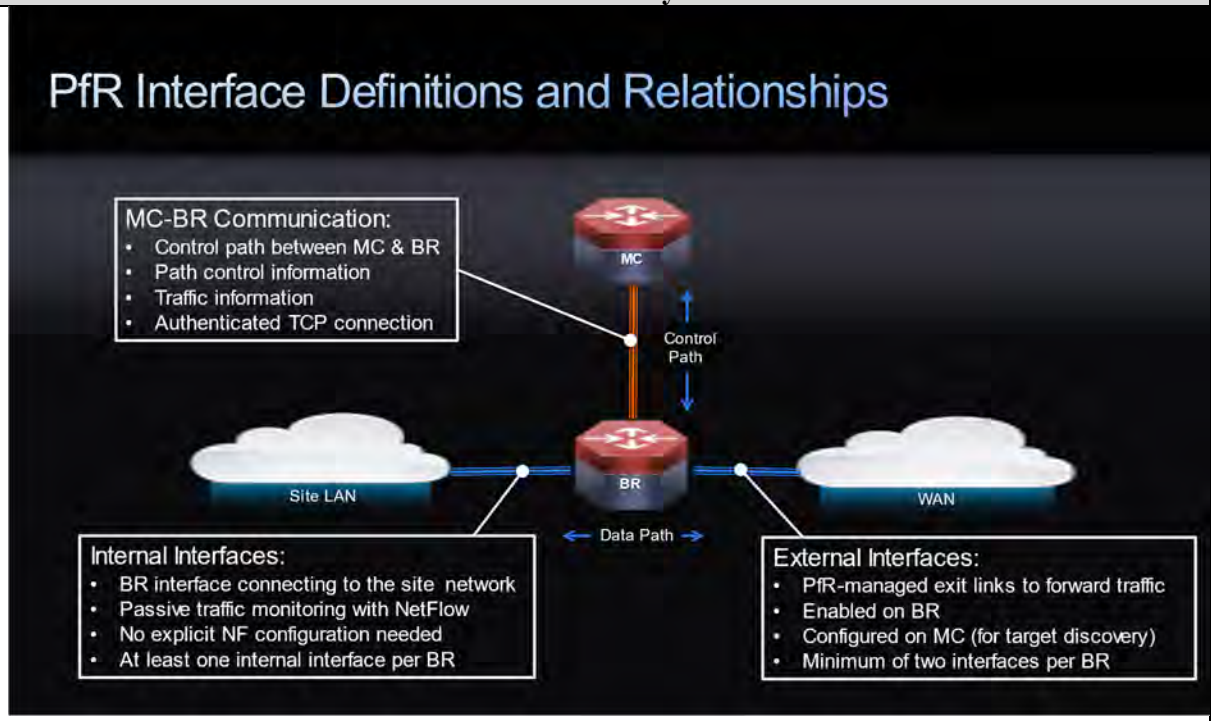
| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | |  <br>Cisco IWAN - Uncompromised Experience |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 896 of 1100

| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | |  <br>Cisco IWAN - Uncompromised Experience |

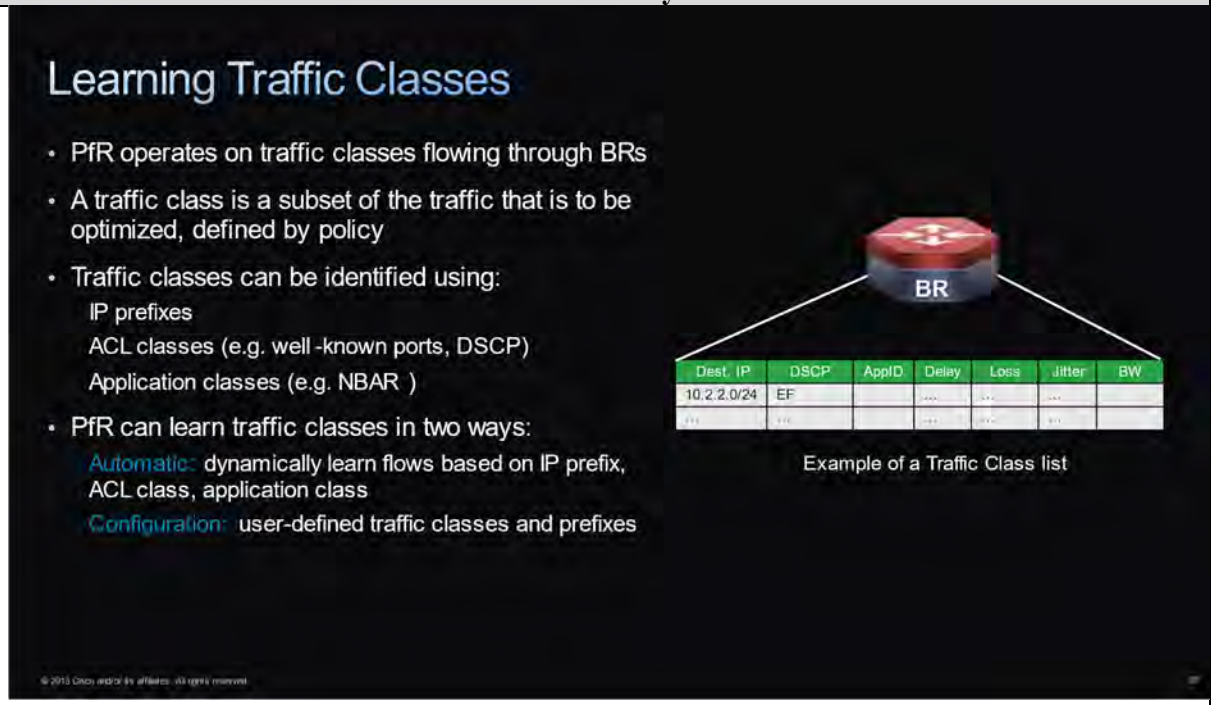
| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | |  |

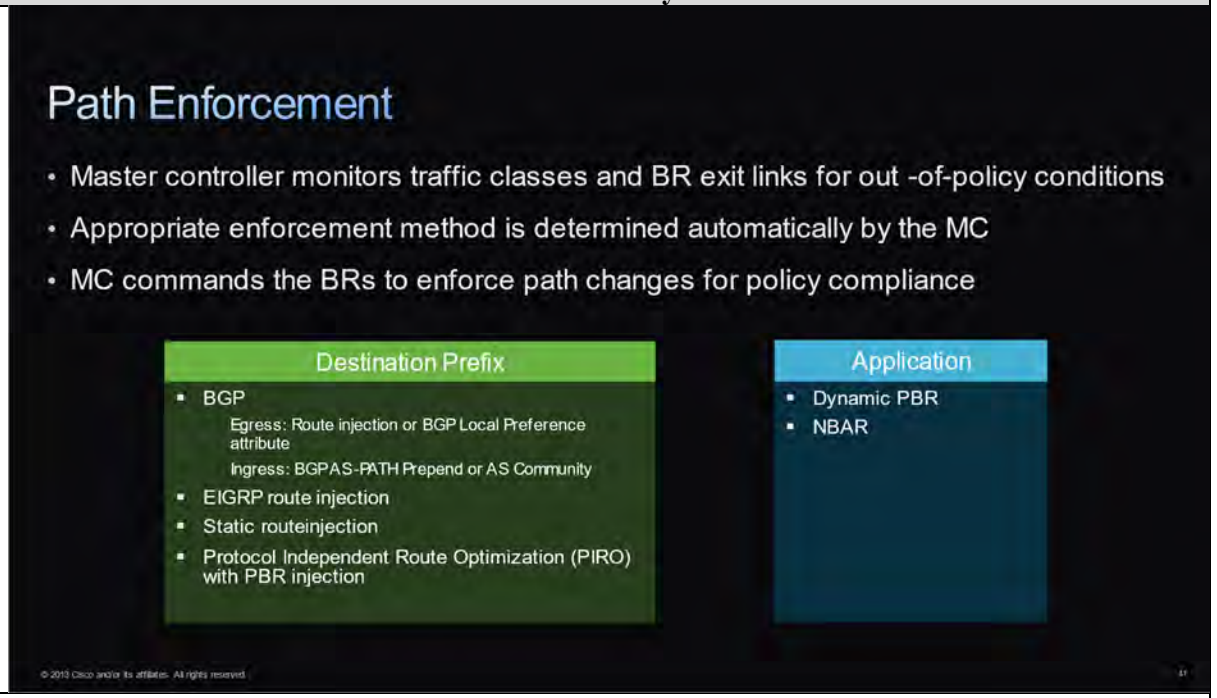
| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | |  |





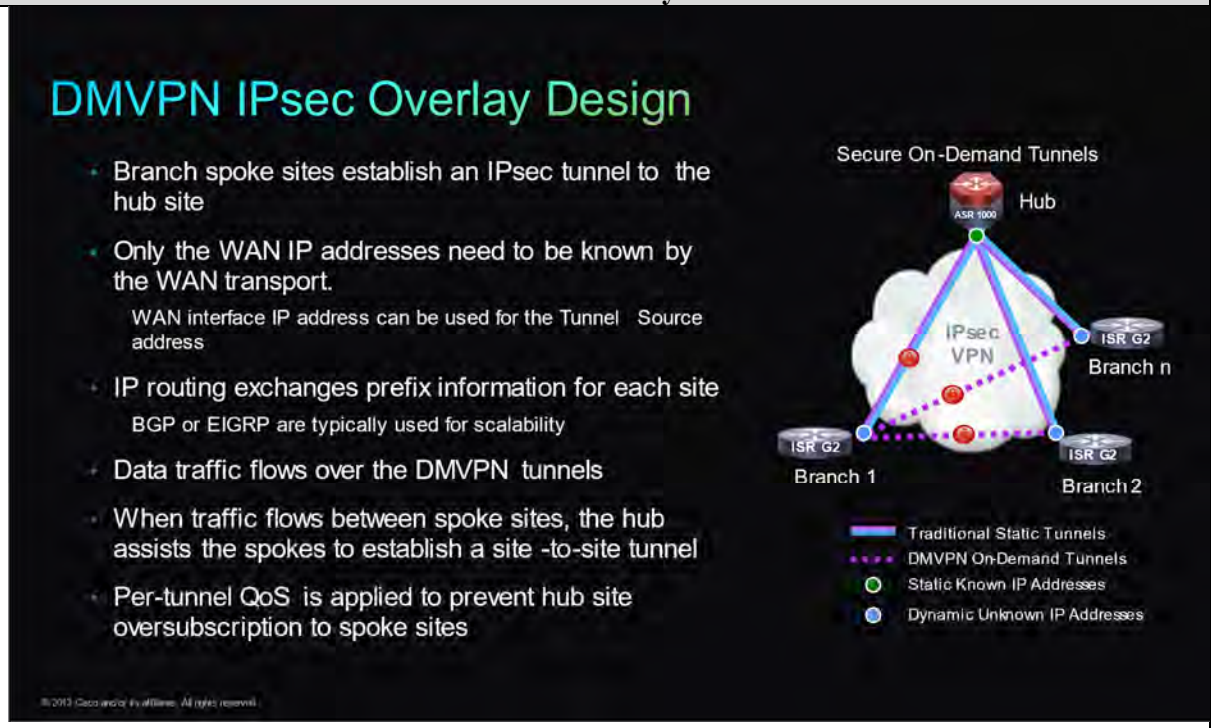
Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 899 of 1100

| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | |  |

| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | |  |
| 1[b] | receiving, by the network node from the controller, the instruction and the criterion; | Cisco IWAN System discloses receiving, by the network node from the controller, the instruction and the criterion.<br><br>*See supra at* 1[a]. |
| 1[c] | receiving, by the network node from the first entity over the packet network, a packet addressed to the second entity; | Cisco IWAN System discloses receiving, by the network node from the first entity over the packet network, a packet addressed to the second entity.<br><br>For example, Cisco IWAN System discloses receiving at a branch device/border router traffic flows between user devices directed to data center devices.<br><br>Cisco IWAN |

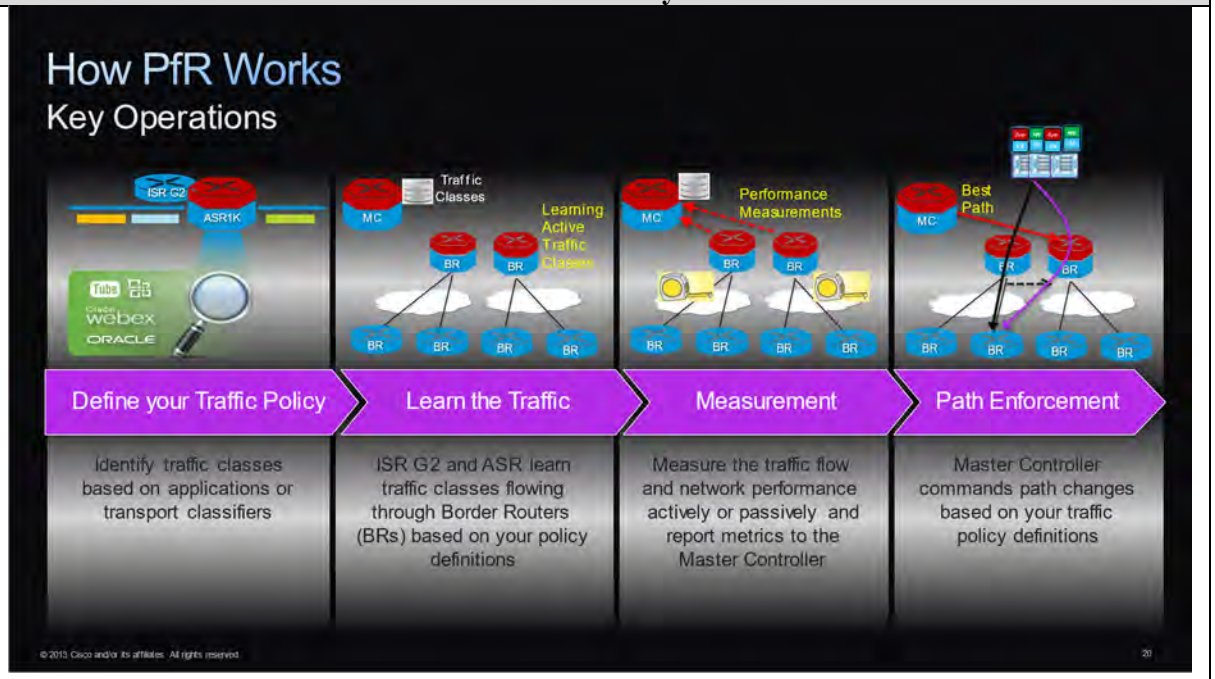


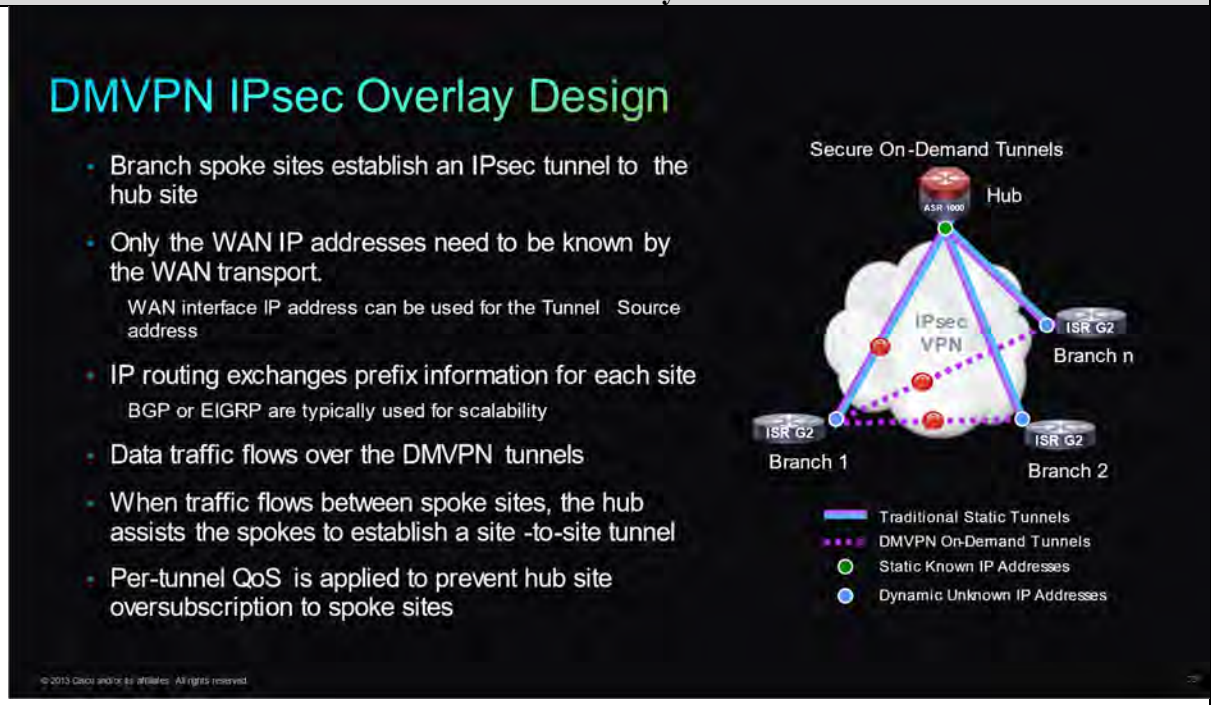
Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 901 of 1100

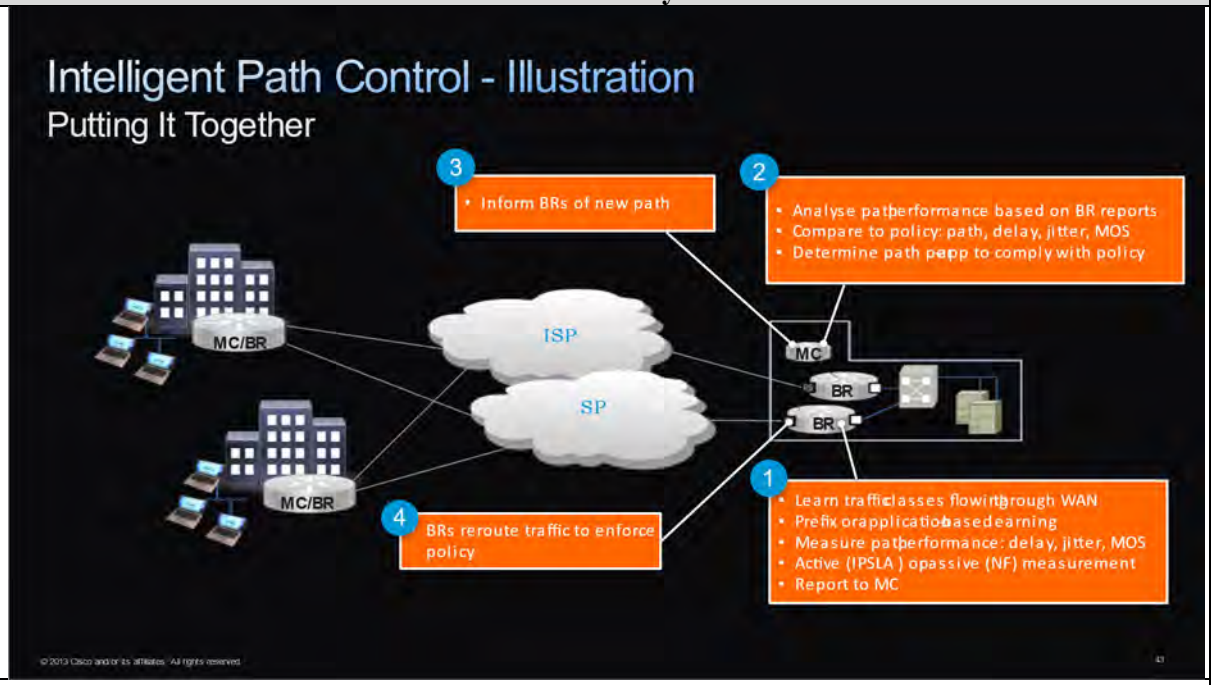
| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | |  <br>Cisco IWAN |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 902 of 1100

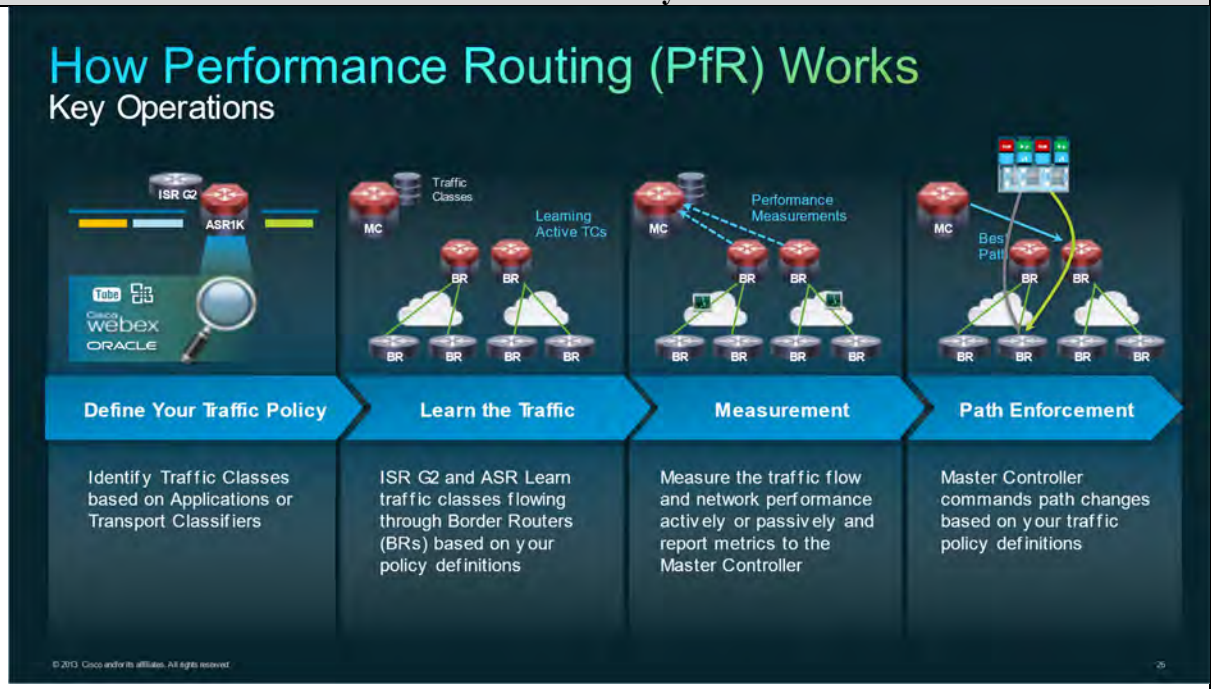
| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | |  <br>Cisco IWAN - Uncompromised Experience |

| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | |  |

| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | |  |
| 1[d] | checking, by the network node, if the packet satisfies the criterion; | Cisco IWAN System discloses checking, by the network node, if the packet satisfies the criterion.<br><br>For example, Cisco IWAN System discloses monitoring, learning, and measuring the traffic flows at the branch device/border router to determine whether the traffic flow meets traffic policy definitions.<br><br>Cisco Next Generation |

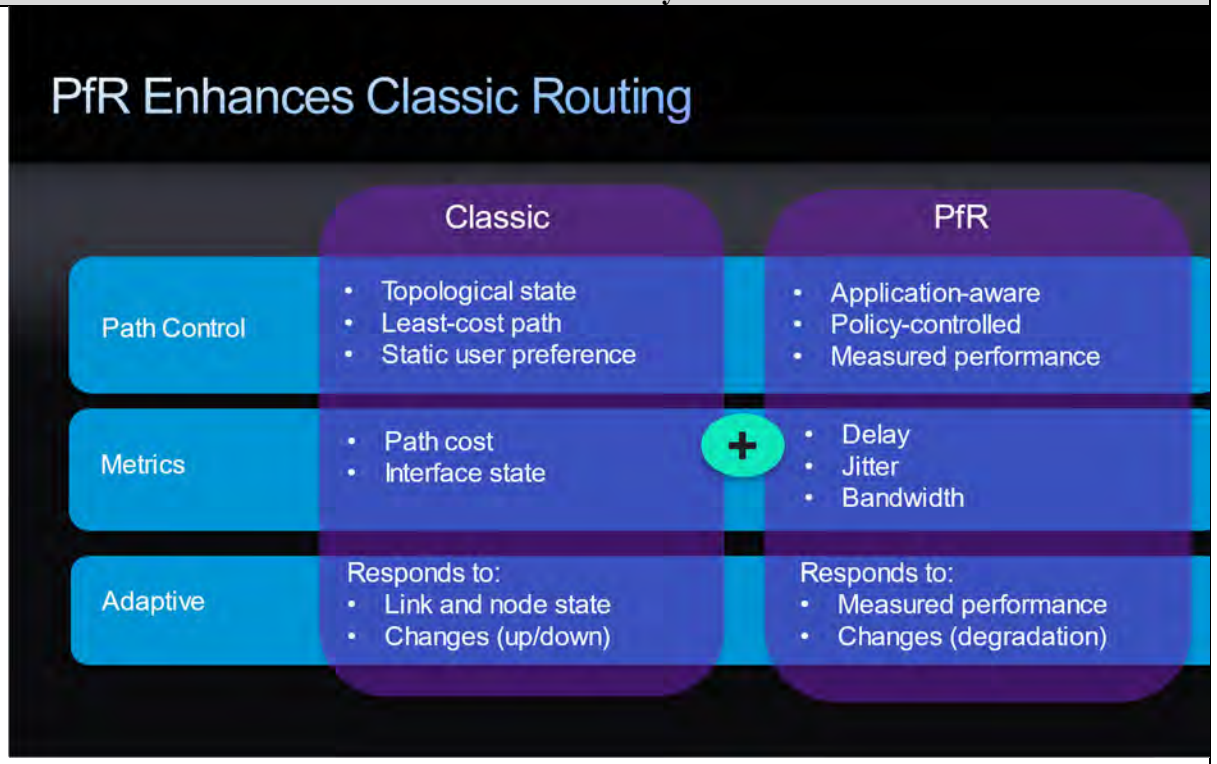


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 905 of 1100

| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | |  <br>Cisco IWAN |

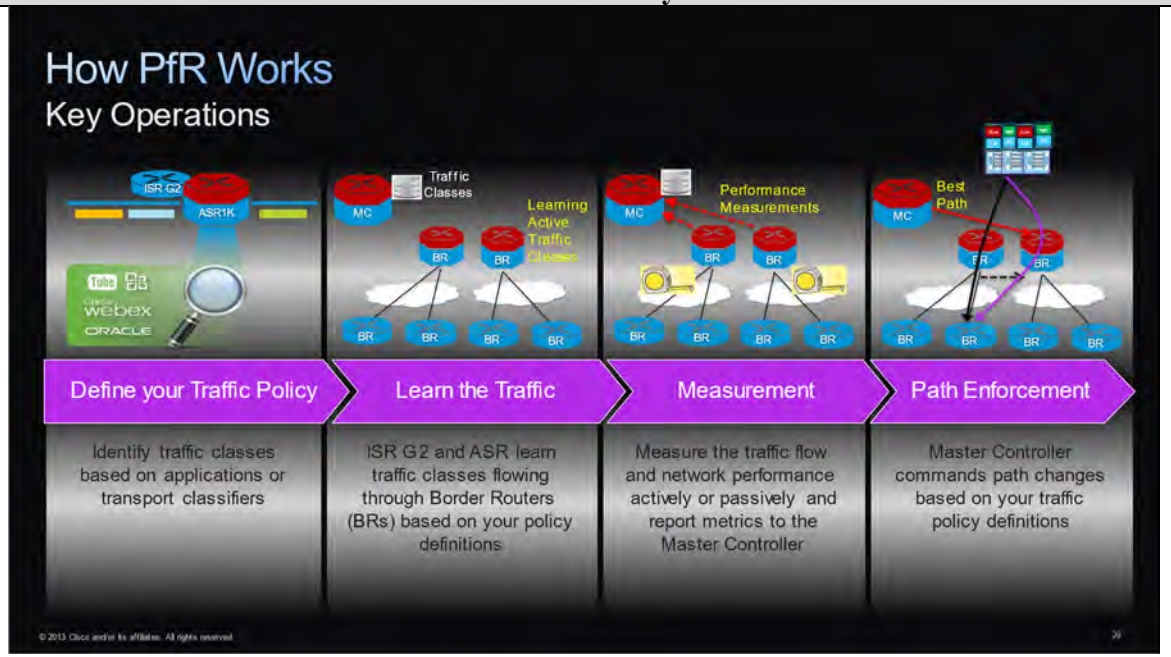


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 906 of 1100

| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | |  <br>Cisco IWAN |

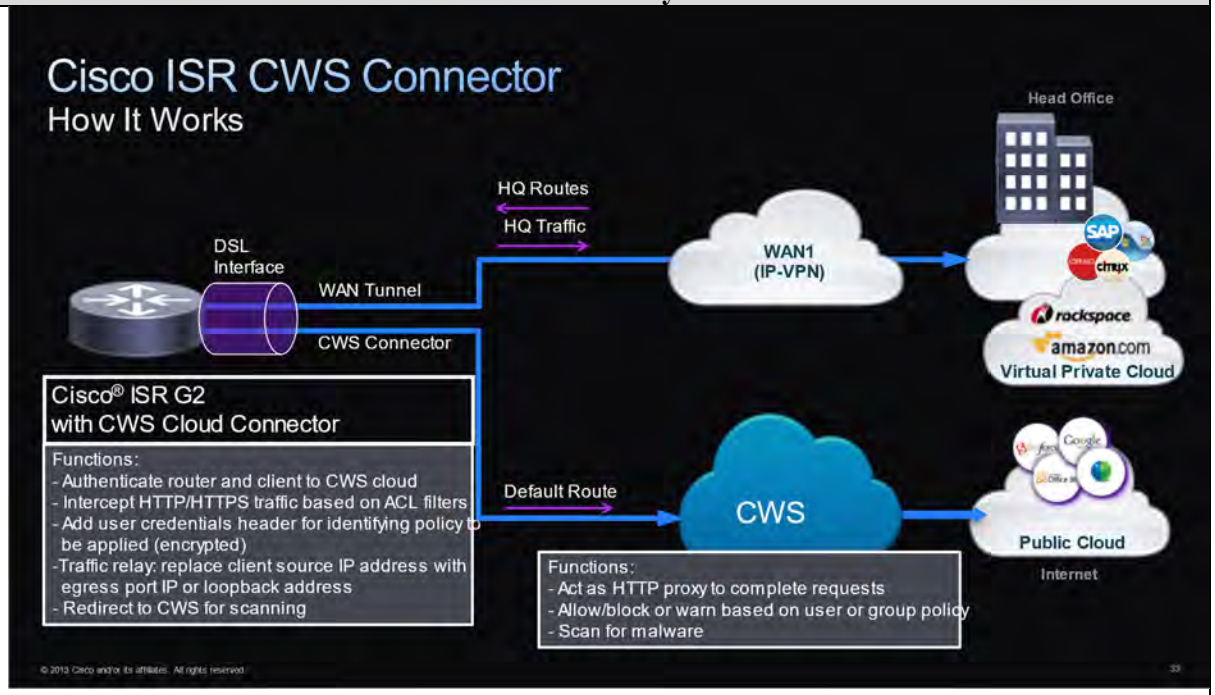


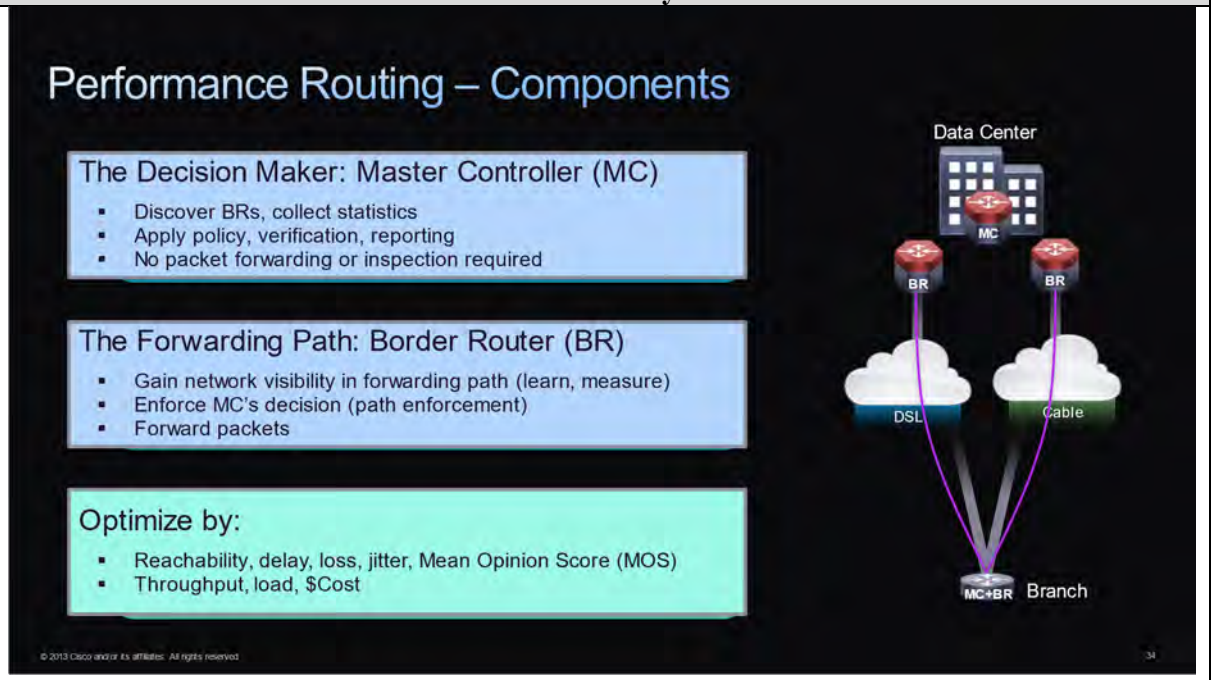
Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 907 of 1100

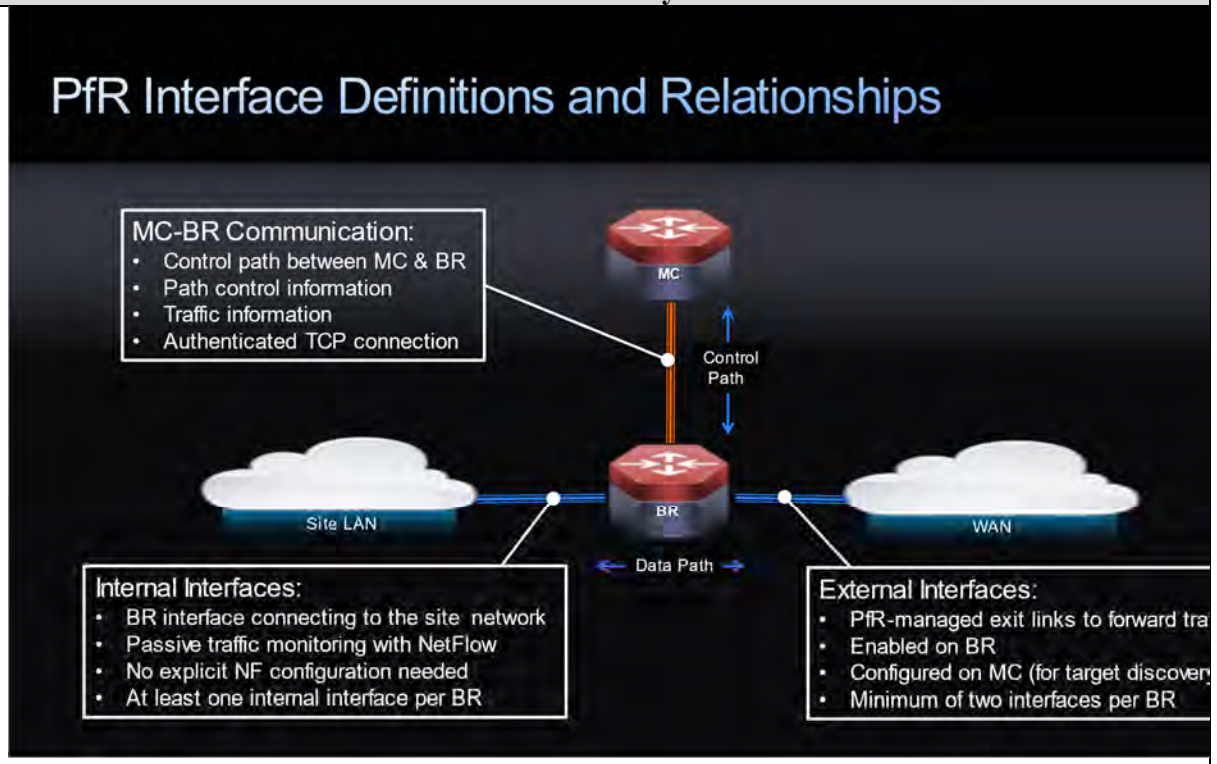
| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | |  <br>Cisco IWAN |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 908 of 1100

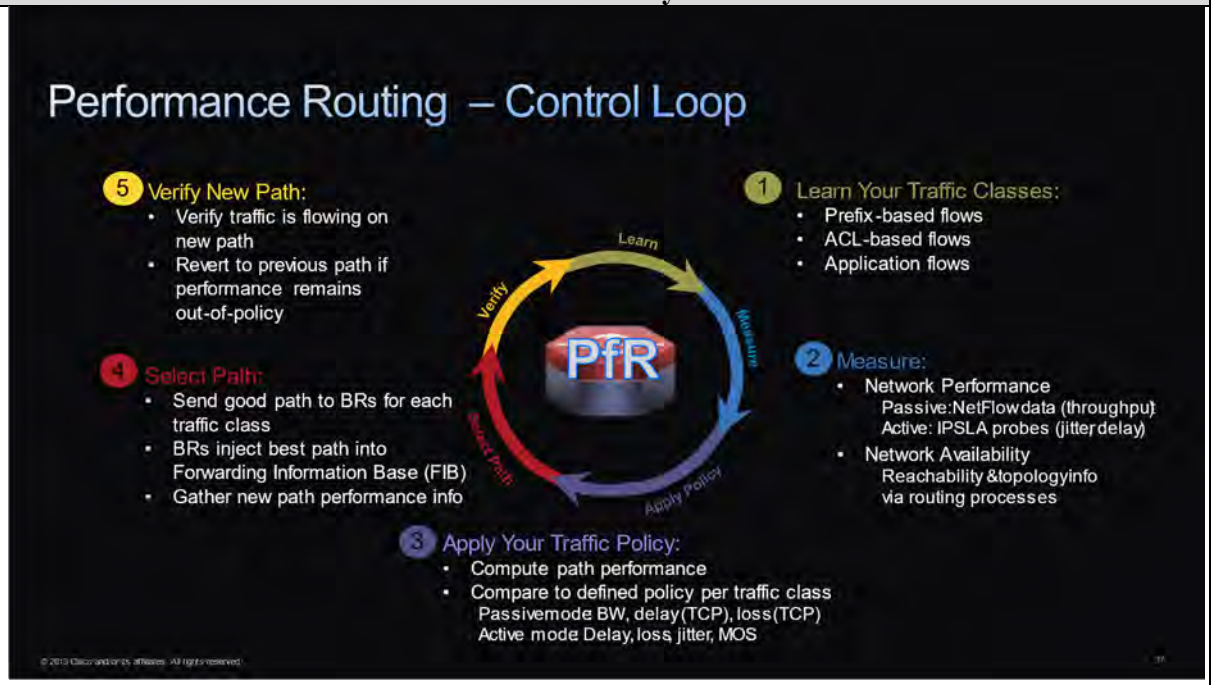
| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | |  <br>Cisco IWAN - Uncompromised Experience |

| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | |  |

| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | |  <br>7 |



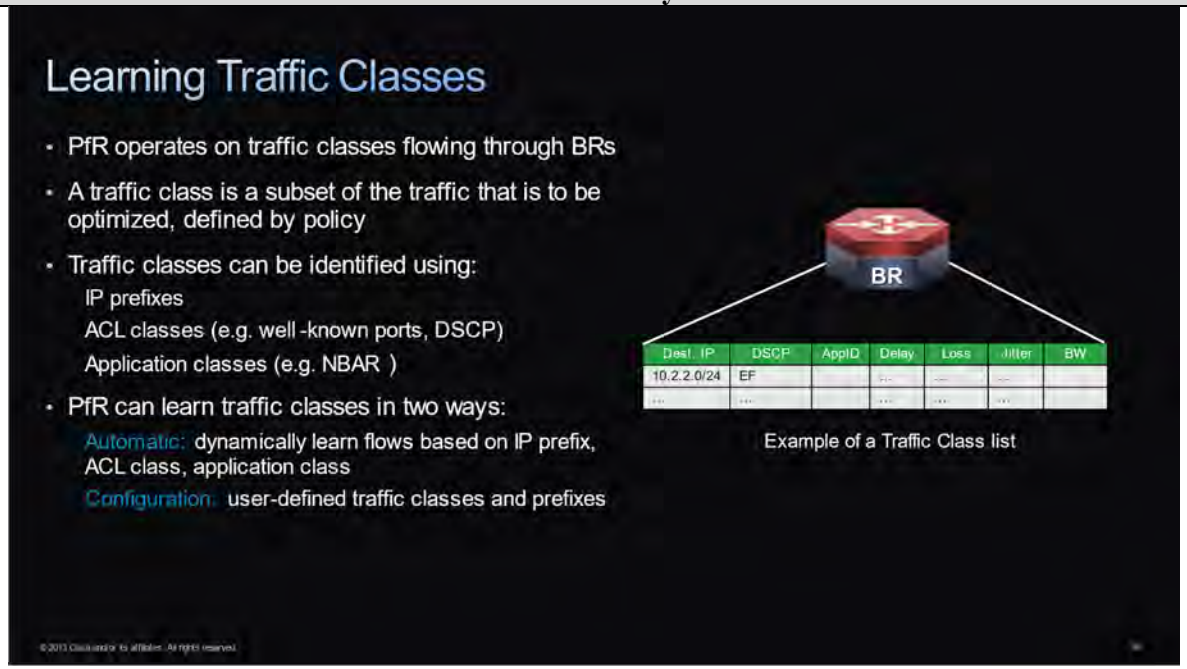
Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 911 of 1100

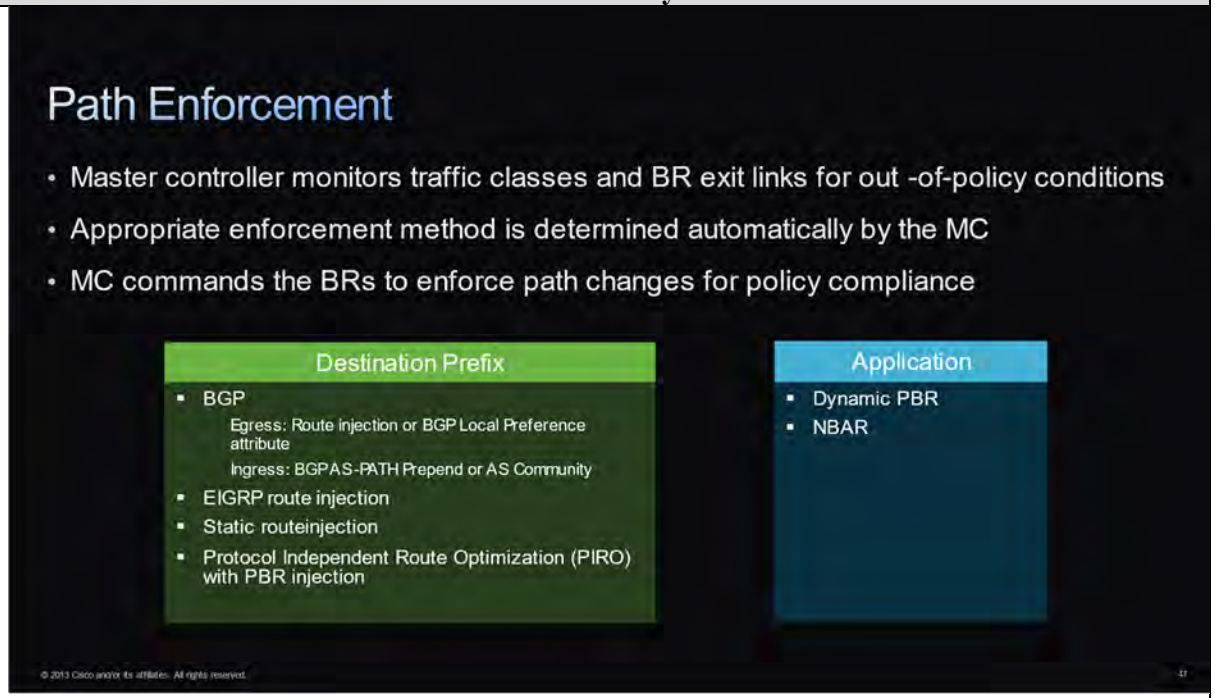
| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | |  |





Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 912 of 1100

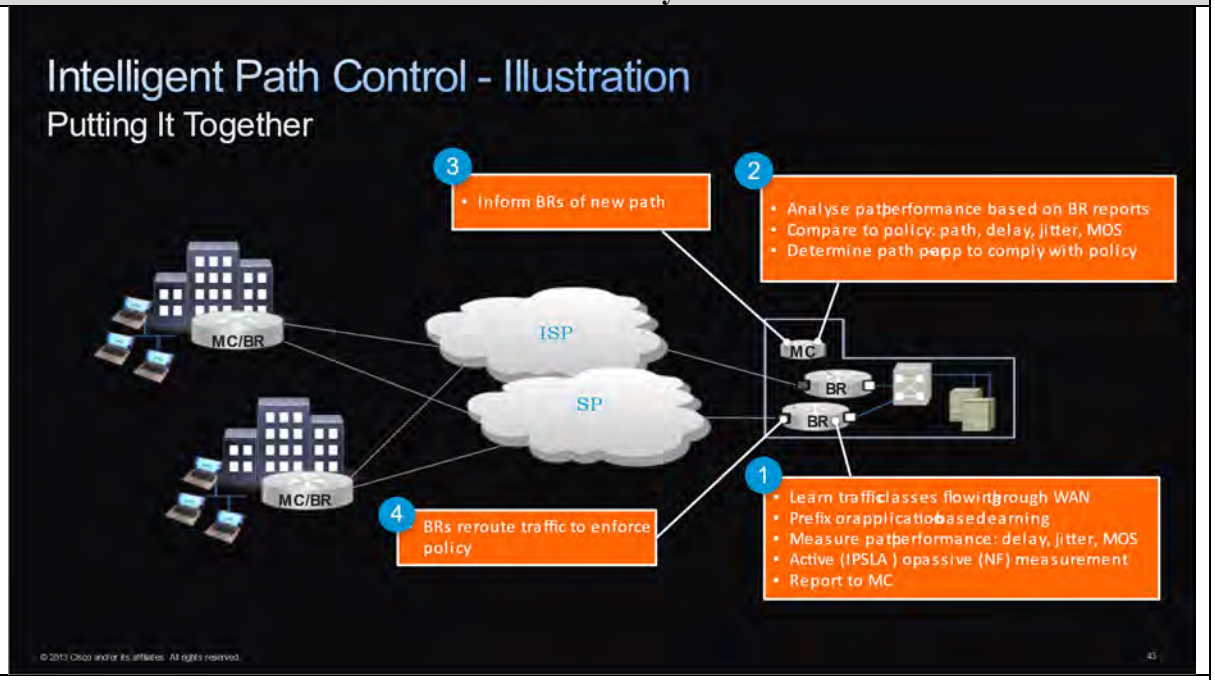
| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | |  |

| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | |  |



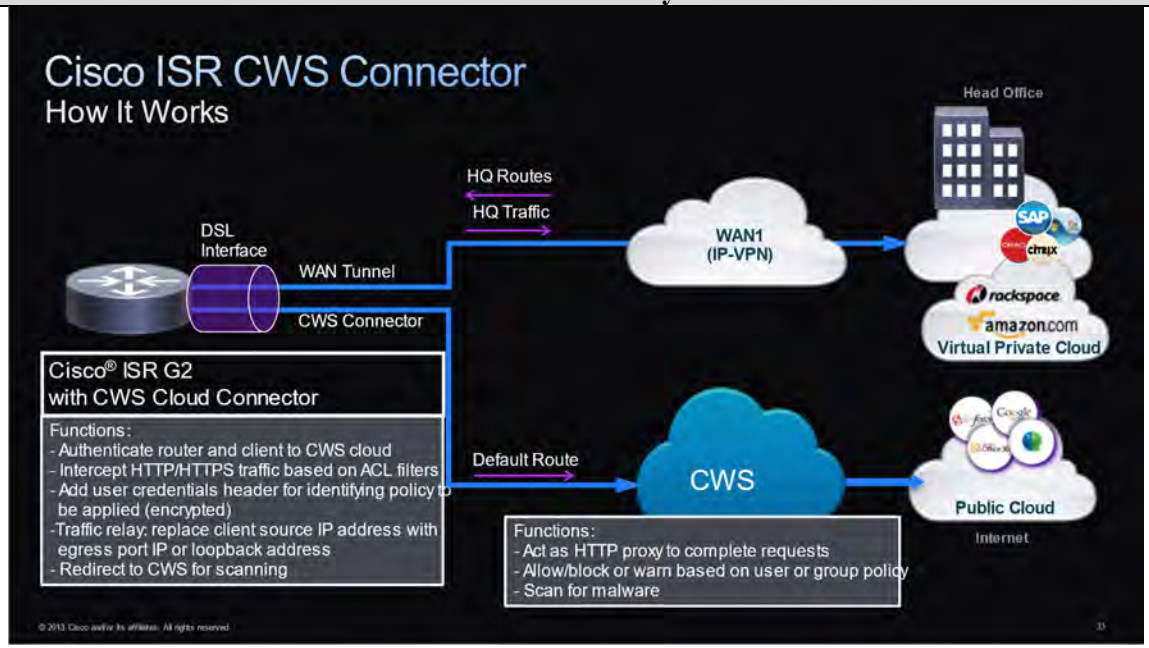


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 914 of 1100

| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | |  |
| 1[e] | responsive to the packet not satisfying the criterion, sending, by the network node over the packet network, the packet to the second entity; and | Cisco IWAN System discloses responsive to the packet not satisfying the criterion, sending, by the network node over the packet network, the packet to the second entity.<br><br>For example, Cisco IWAN System discloses relaying traffic by the branch device/border router to intended data center devices based on a particular traffic class. A person of ordinary skill in the art would understand that a packet not satisfying the criterion depends on the particular traffic policy definition and traffic class of the packet. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Cisco IWAN System is found to not meet this limitation, responsive to the packet not satisfying the criterion, sending, by the network node over the packet network, the packet to the second entity would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>Cisco IWAN |

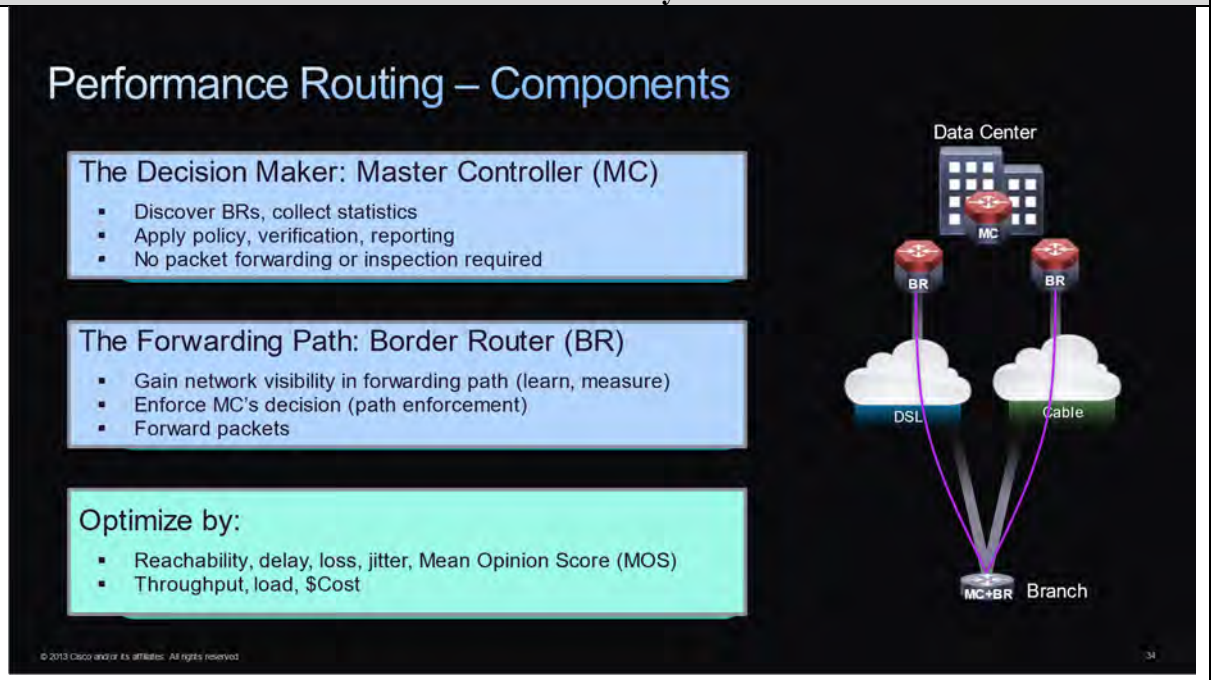


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 915 of 1100

| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | |  <br>Cisco IWAN - Uncompromised Experience |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 916 of 1100

| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | |  |

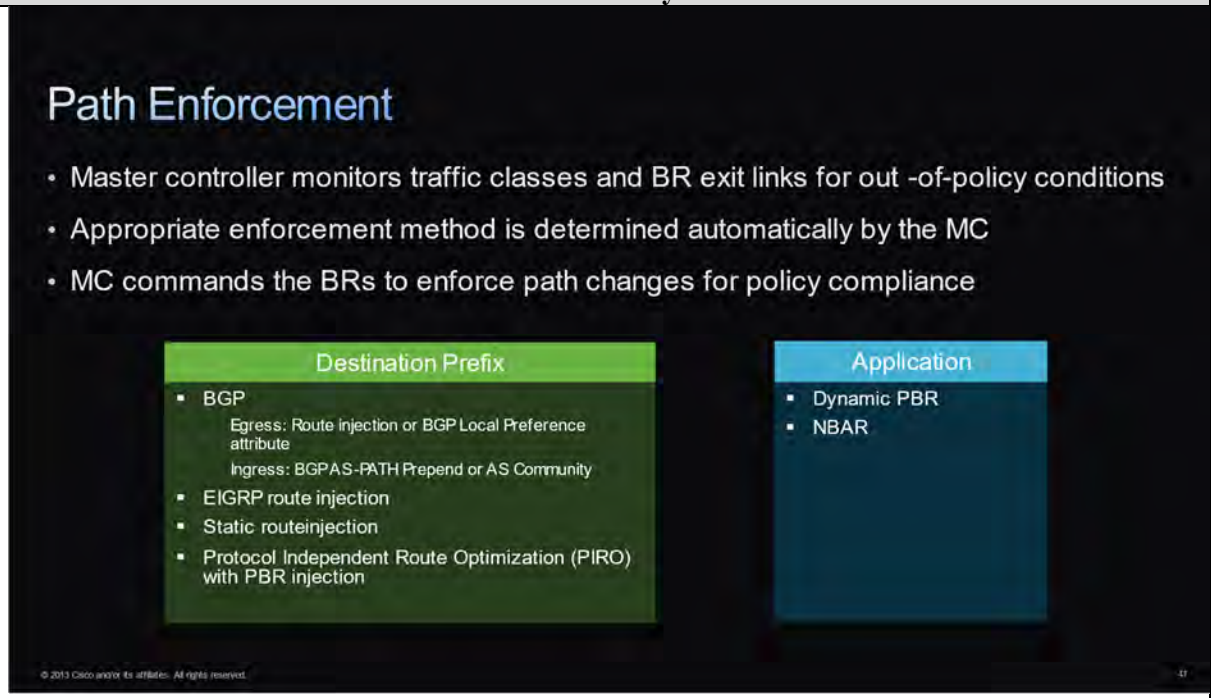
| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | |  |





Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 918 of 1100

| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|




Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Cisco IWAN System in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 1[e] of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references.

For example, Kempf discloses sending the packet from the network element to the destination device in response to the packet not matching the action in the flow table.

Kempf at [0044] ("FIG. 1 is a diagram of one embodiment of an example network with an OpenFlow switch, conforming to the OpenFlow 1.0 specification. The OpenFlow 1.0 protocol enables a controller 101 to connect to an OpenFlow 1.0 enabled switch 109 using a secure channel 103 and control a single forwarding table 107 in the switch 109. The controller 101 is an external software component executed by a remote computing device



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 919 of 1100

| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | | that enables a user to configure the Open-Flow 1.0 switch 109. The secure channel 103 can be provided by any type of network including a local area network (LAN) or a wide area network (WAN), such as the Internet.")<br><br>Kempf at [0045] ("FIG. 2 is a diagram illustrating one embodiment of the contents of a flow table entry. The forwarding table 107 is populated with entries consisting of a rule 201 defining matches for fields in packet headers; an action 203 associated to the flow match; and a collection of statistics 205 on the flow. When an incoming packet is received a lookup for a matching rule is made in the flow table 107. If the incoming packet matches a particular rule, the associated action defined in that flow table entry is performed on the packet.")<br><br>Kempf at [0046] ("A rule 201 contains key fields from several headers in the protocol stack, for example source and destination Ethernet MAC addresses, source and destination IP addresses, IP protocol type number, incoming and outgoing TCP or UDP port numbers. To define a flow, all the available matching fields may be used. But it is also possible to restrict the matching rule to a subset of the available fields by using wildcards for the unwanted fields.")<br><br>Kempf at [0047] ("The actions that are defined by the specification of OpenFlow 1.0 are Drop, which drops the matching packets; Forward, which forwards the packet to one or all outgoing ports, the incoming physical port itself, the controller via the secure channel, or the local networking stack (if it exists). OpenFlow 1.0 protocol data units (PDU s) are defined with a set of structures specified using the C programming language. Some of the more commonly used messages are: report switch configuration message; modify state messages (in-cluding a modify flow entry message and port modification message); read state messages, where while the system is running, the datapath may be queried about its current state using this message; and send packet message, which is used when the controller wishes to send a packet out through the datapath.")<br><br>Kempf at [0050] ("FIG. 4 illustrates one embodiment of the processing of packets through an OpenFlow 1.1 switched packet pro-cessing pipeline. A received packet is compared against each of the flow tables 401. After each flow table match, the actions are accumulated into an action set. If processing requires matching against another flow table |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 920 of 1100

| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | | the actions in the matched rule include an action directing processing to the next table in the pipeline. Absent the inclusion of an action in the set to execute all accumulated actions immediately, the actions are executed at the end 403 of the packet processing pipeline. An action allows the writing of data to a metadata register, which is carried along in the packet processing pipe-line like the packet header.")<br><br>Kempf at [0051] ("FIG. 5 is a flowchart of one embodiment of the OpenFlow 1.1 rule matching process. OpenFlow 1.1 contains support for packet tagging. OpenFlow 1.1 allows matching based on header fields and multi-protocol label switching (MPLS) labels. One virtual LAN (VLAN) label and one MPLS label can be matched per table. The rule matching process is initiated with the arrival of a packet to be processed (Block 501 ). Starting at the first table 0 a lookup is performed to determine a match with the received packet (Block 503). If there is no match in this table, then one of a set of default actions is taken (i.e., send packet to controller, drop the packet or continue to next table) (Block 509). If there is a match, then an update to the action set is made along with counters, packet or match set fields and meta data (Block 505). A check is made to determine the next table to process, which can be the next table sequentially or one specified by an action of a matching rule (Block 507). Once all of the tables have been processed, then the resulting action set is executed (Block 511). FIG. 6 is a diagram of the fields, which a matching process can utilize for identifying rules to apply to a packet.")<br><br>Kempf at [0053] ("In one embodiment, a group table can be supported in conjunction with the OpenFlow 1.1 protocol. Group tables enable a method for allowing a single flow match to trigger forwarding on multiple ports. Group table entries consist of four fields: a group identifier, which is a 32 bit unsigned integer identifying the group; a group type that determines the group's semantics; counters that maintain statistics on the group; and an action bucket list, which is an ordered list of action buckets, where each bucket contains a set of actions to execute together with their parameters.")<br><br>Kempf at [0091] ("When a packet header matches a rule associated with the virtual port, the GTP TEID is written into the lower 32 bits of the metadata and the packet is directed to the virtual port. The virtual port calculates the hash of the TEID and looks up the tunnel header information in the tunnel header table. If no such tunnel information is present, the packet is forwarded to the controller with an error indication. Other-wise, the virtual port constructs a |

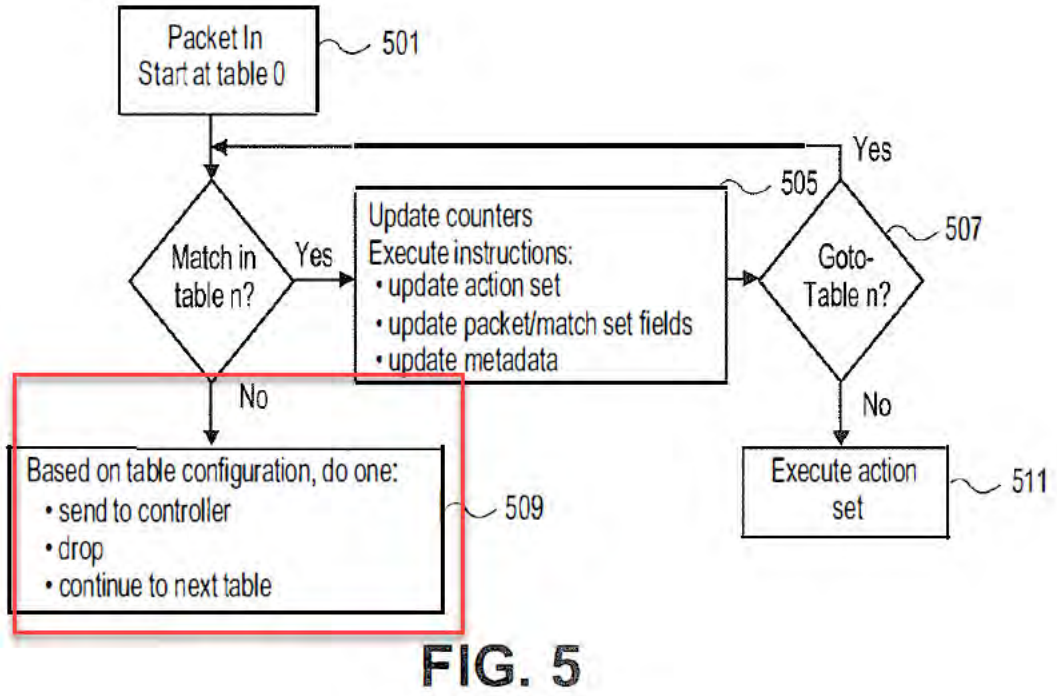


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 921 of 1100

| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | | GTP tunnel header and encapsulates the packet. Any DSCP bits or VLAN priority bits are additionally set in the IP or MAC tunnel headers, and any VLAN tags or MPLS labels are pushed onto the packet. The encapsulated packet is forwarded out the physical port to which the virtual port is bound.")<br><br>Kempf at [0092] ("In one embodiment, the system implements a GTP fast path decapsulation virtual port. When requested by the S-GW and P-GW control plane software running in the cloud computing system, the gateway switch installs rules and actions for routing GTP encapsulated packets out of GTP tunnels. The rules match the GTP header flags and the GTP TEID for the packet, in the modified OpenFlow flow table shown in FIG. 17 as follows: the IP destination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); the UDP destination port is the GTP-U destination port (2152); and the header fields and message type field is wildcarded with the flag 0XFFF0 and the upper two bytes of the field match the G-PDU message type (255) while the lower two bytes match 0x30, i.e. the packet is a GTP packet not a GTP' packet and the version number is 1.")<br><br>Kempf at Figure 5 (annotation added) |

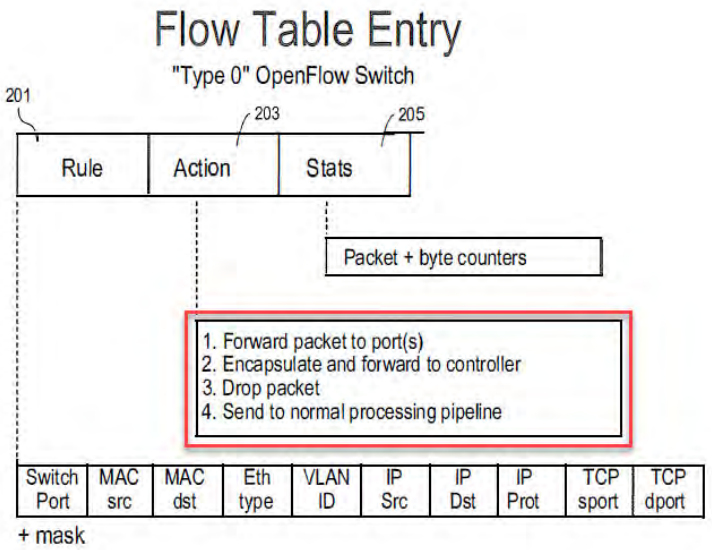


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 922 of 1100

| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | |  <br>FIG. 5<br>Kempf at Figure 2 (annotation added) |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 923 of 1100

| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | |  <br>FIG. 2<br><br>As another example, Swenson discloses monitoring and categorizing network traffic by the steering device based on instructions and desired criteria sent by the network controller to determine if packet flows require further inspection. Based on the instruction and desired criteria, the network controller monitors and optimizes only a subset of network traffic. Packet flows that do not meet the desired criteria from the network controller's instructions at the steering device are not sent for further inspection and are sent to their originally intended destination.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 924 of 1100

| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | | device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0028] ("In contrast to conventional inline TCP throughput monitoring devices that monitor every single data packets transmitted and received, the network controller 140 is an "out-of-band" computer server that interfaces with the steer-ing device 130 to selectively inspect user flows of interest. The network controller 140 may further identify user flows (e.g., among the flows of interest) for optimization. In one embodiment, the network controller 140 may be imple-mented at the steering device 130 to monitor traffic. In other embodiments, the network controller 140 is coupled to and communicates with the steering device 130 for traffic moni-toring and optimization. When queried by the steering device 130, the network controller 140 determines if a given network flow should be ignored, monitored further or optimized. Opti-mization of a flow is often decided at the beginning of the flow because it is rarely possible to switch to optimized content mid-stream once non-optimized content delivery has begun. However, the network controller 140 may determine that existing flows associated with a particular subscriber or other entity should be optimized. In turn, new flows ( e.g., resulting from seek requests in media, new media requests, resume after pause, etc.) determined to be associated with the entity may be optimized. The network controller 140 uses the net-work state as well as historical traffic data in its decision for monitoring and optimization. Knowledge on the current net-work state, such as congestion, deems critical when it comes to data optimization.")<br><br>Swenson at [0038] ("Turning back to FIG. 1, the network controller 140 allows network operators to apply fine granular optimization policies to ensure high quality of experience (QoE) based on cell tower congestion, device types, subscriber profiles and service plans with lower hardware and software costs. The architecture of the network controller 140 provides an excel-lent fit for the net neutrality guideline of"reasonable network management", and better compliance to the copyright law (DMCA) than solutions that rely on long-term caching. Hav-ing the ability of monitoring network traffic on a per sub-scriber, per flow, or per video file basis, the network controller 140 also selectively monitors and |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 925 of 1100

| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | | optimizes only a subset of traffic that benefits from optimization the most, thus achiev-ing both scalability and efficiency for optimization at a com-petitive price-point. The core element of the network control-ler 140 lies in its mechanisms for congestion detection and mitigation, which allows optimization resources to be utilized in the most efficient and surgical manner.")<br><br>Swenson at [0042] ("The network controller 140 collects real-time statis-tical data on the network flows from core network side with-out probes deployed in the RAN network. The statistical data is stored and compared against historical flow data to estimate level of congestion and available network bandwidth. Instead of collecting traffic statistics for every flow and every session, the network controller 140 samples only large flows involving media objects such as videos and images above a certain size ( e.g., above 50 kB). The network controller 140 can choose to be a pass-through device to monitor the large flows as well as to determine whether to optimize the flows. Measuring only larger flows has the advantage to mitigate corruptions caused by origin server latency and network glitches. Furthermore, focusing on the large flows helps the network controller to reduce the background noise and to increase noise-to-signal ratio in bandwidth measuring by removing the impact of millions of tiny or small flows with delivery time in millisec-onds. Therefore the reliability of bandwidth estimation and congestion detection is much higher.")<br><br>Swenson at [0045] ("The steering device interface 316 interacts with an external routing appliance, such as the steering device 130 to divert portions of the network traffic ( e.g., large object net-work flows). Existing routing appliances in most carrier net-works are designed to handle large amounts of network traf-fic. They are not, however, ideal devices to operate for monitoring and analysis individual flows. Through the steer-ing device interface 316, the network controller 140 may communicate with the external routing appliances, such as the steering device 130, to steer a portion of network traffic to the network controller 140 when certain conditions are met. Generally, network flows of interest to the network controller 140 contain larger media objects, such as videos and images. In one embodiment, the smaller flows, such as web page and text information, are not exchanged over the steering device interface 316.")<br><br>Swenson at [0059] ("In one embodiment, as the steering device 130 monitors network responses, it is looking for flows that match one or more signatures for video and images |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 926 of 1100

| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | | When a match-ing flow is detected, the steering device 130 forwards the HTTP request and a portion of the HTTP response to the network controller 140 over the ICAP client interface 404. After receiving the request and the portion of response at the ICAP server interface 406, the flow analyzer 312 of the net-work controller 140 performs a deep flow inspection to deter-mine if the flow is worth bandwidth monitoring and/or user detection. For example, the flow inspection performed by the flow analyzer 312 may determine if the flow indeed contains large or medium object ( e.g., larger than 50 kB), and/or if the source IP address of the flow is from a user or a group of users that are required to be monitored by policies. The flow ana-lyzer 312 may also determine if the flow needs to be opti-mized based on historical flow statistical data.")<br><br>Swenson at [0060] ("If the flow is deemed of interest, the steering device 130 is notified to steer the flow through the network controller 140. This is known as the "continue" working mode for bandwidth monitoring. In the "continue" mode, the network controller 140 interfaces with the steering device 130 to func-tion, on-demand, as a traditional inline network element for flows deemed of interest. Thus, the network controller 140 ingests the network flow for inspection and subsequently forwards the network flow on the network response path. For example, for this particular flow, the origin server 160 responds to the user request by sending video or images over the network link 413 to the steering device 130, which for-wards the video or images to the network controller 140 over a network link 414. After the network controller 140 updates the flow statistics, the video or images are returned to the steering device 130 over a network link 415, which transmits the video or images to the user device 110 over the network link 416.")<br><br>Swenson at [0065] ("FIG. 5 is a block diagram illustrating an example event trace of "continue" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 512 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 512 to the intended origin server 160 and receives a response 514 back from the origin server 160. The steering device 130 then sends an ICAP request message 516 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this |

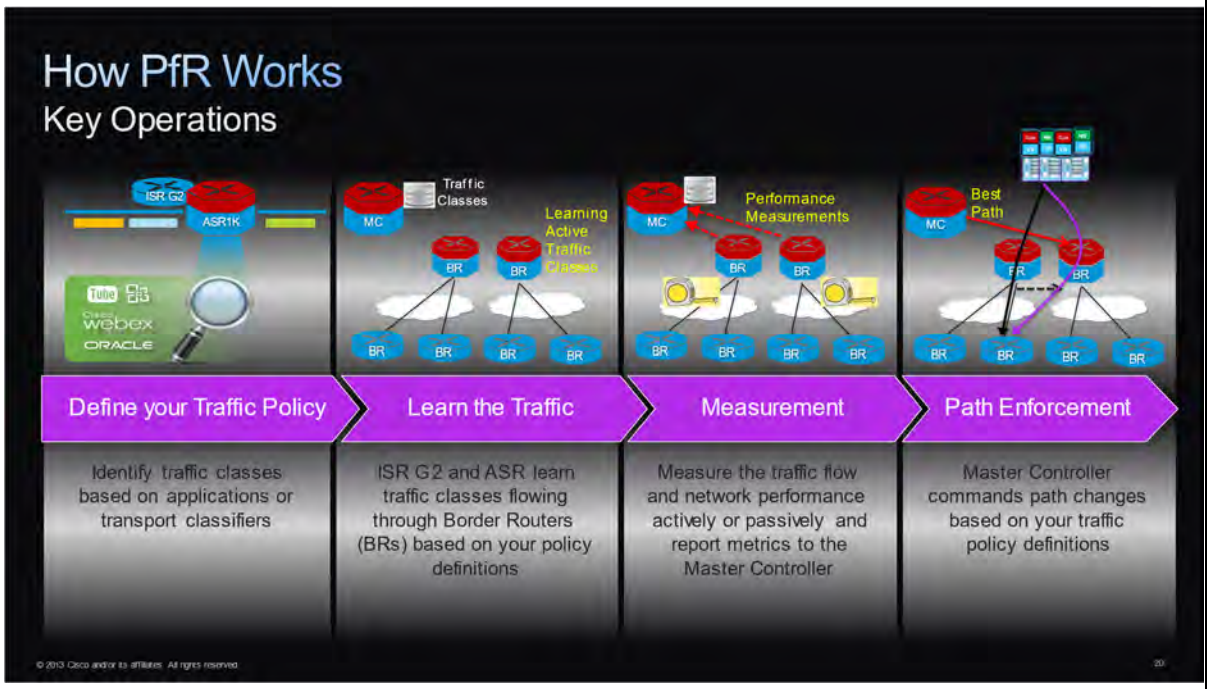


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 927 of 1100

| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | | case, the network controller 140 responds with a redirect to optimize the video in ICAP response 518. Upon receiving the instruc-tion, the steering device 130 re-writes the response 514 to an HTTP redirect response 520, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 520 directly to the user device 110. In case the flow dose not contain video or image objects, or the network controller 140 determines not to monitor the flow, the steering device 13 0 would forward the response to the user device 110.")<br><br>Swenson at [0069] ("FIG. 6 is a block diagram illustrating an example event trace of "counting" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 612 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 612 to the intended origin server 160 and receives a response 614 back from the origin server 160. The steering device 130 then sends an ICAP request message 616 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 618. Upon receiving the instruc-tion, the steering device 130 re-writes the response 614 to an HTTP redirect response 620, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 620 directly to the user device 110. In case the flow dose not contain video or image objects that need to be redirected, the steering device 130 would forward the response to the user device 110.") |
| 1[f] | responsive to the packet satisfying the criterion, sending the packet, by the network node over the packet network, to an entity that is included in the | Cisco IWAN System discloses responsive to the packet satisfying the criterion, sending the packet, by the network node over the packet network, to an entity that is included in the instruction and is other than the second entity.<br><br>For example, Cisco IWAN System discloses sending traffic flow metrics to the hub/master controller by the branch device/border router based on a particular traffic class. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Cisco IWAN System is found to not meet this |

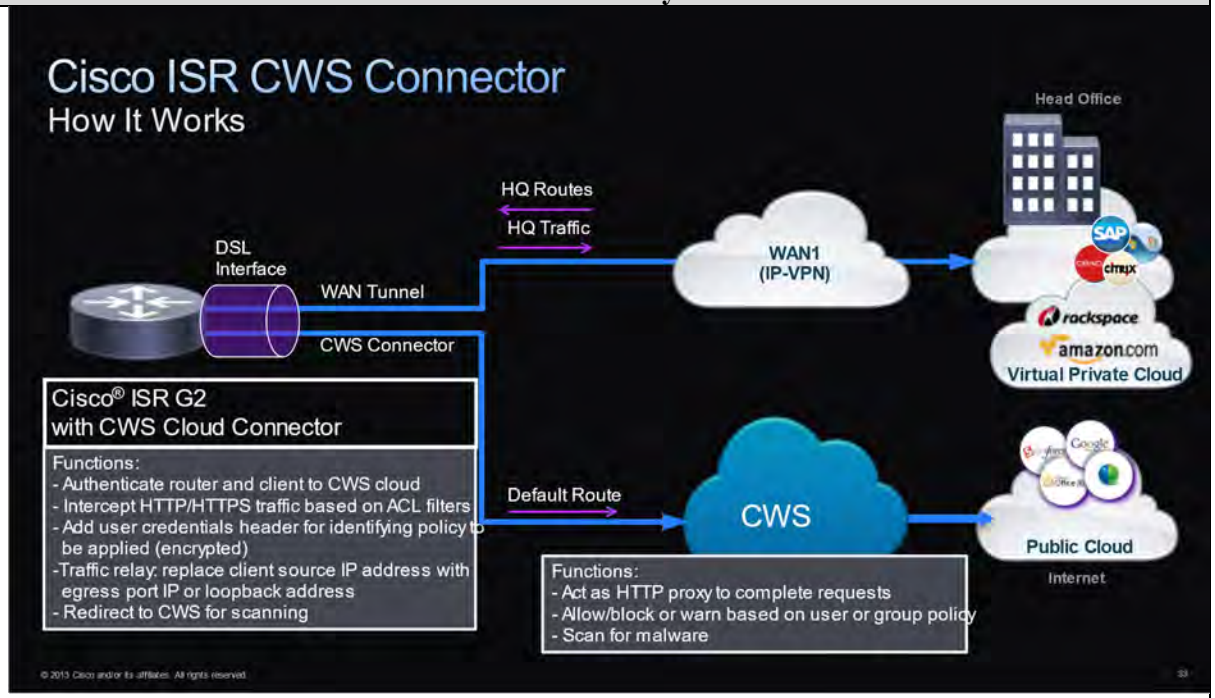


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 928 of 1100

| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | instruction and is other than the second entity. | limitation, responsive to the packet satisfying the criterion, sending the packet, by the network node over the packet network, to an entity that is included in the instruction and is other than the second entity would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>Cisco IWAN<br><br>  <br><br>Cisco IWAN |

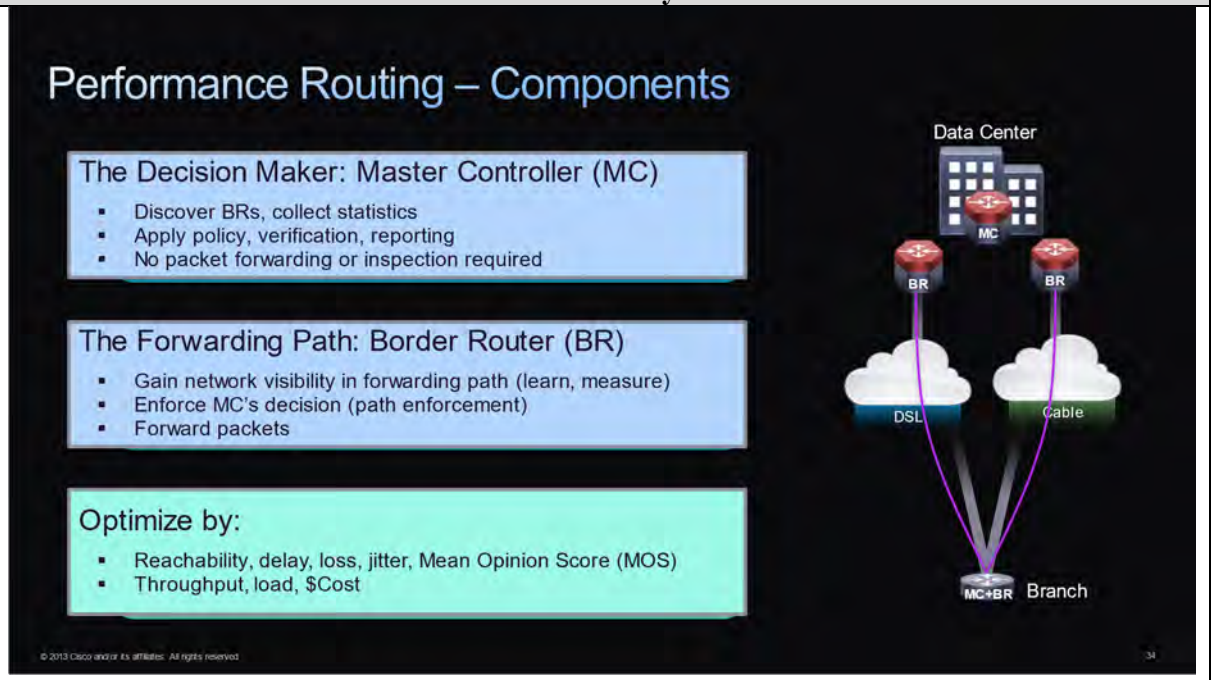


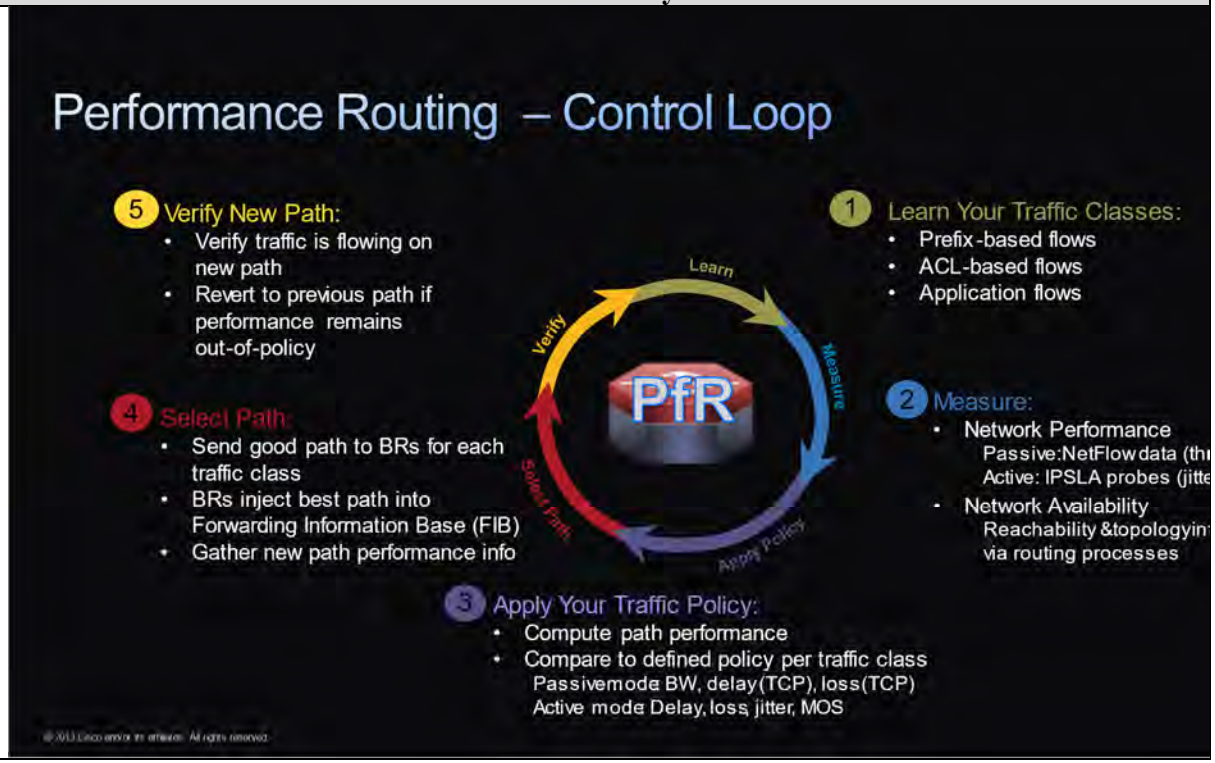
Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 929 of 1100

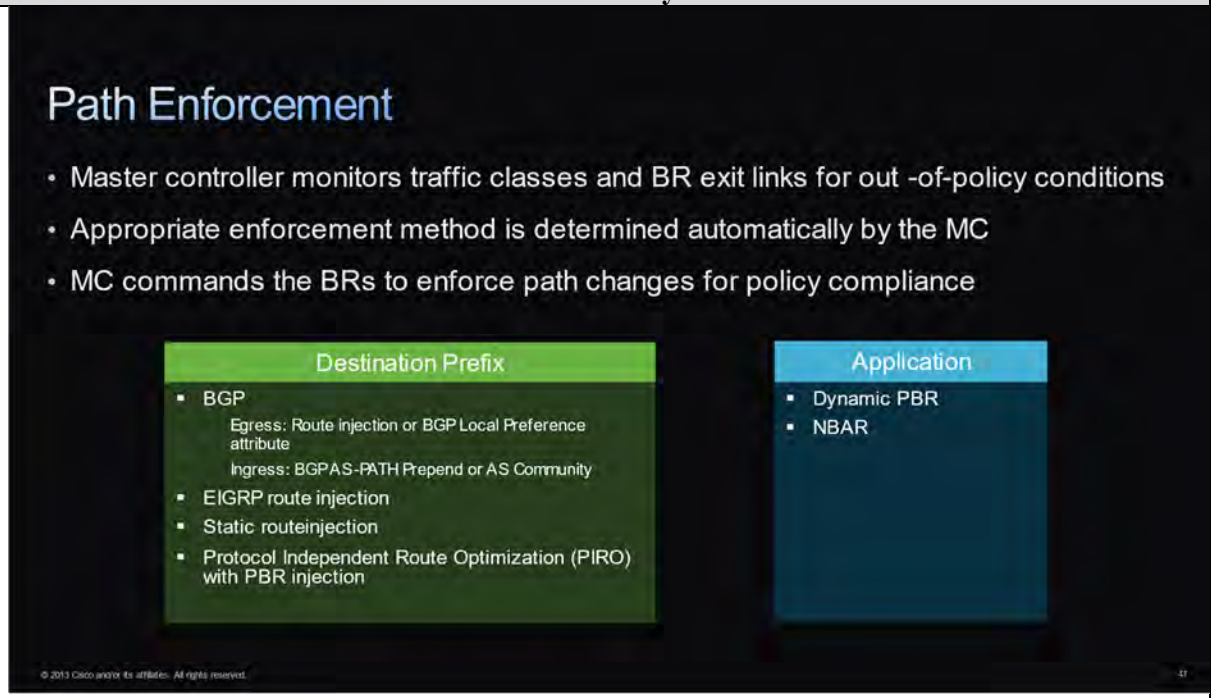
| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | |  <br>Cisco IWAN - Uncompromised Experience |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 930 of 1100

| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | |  |

| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | |  |

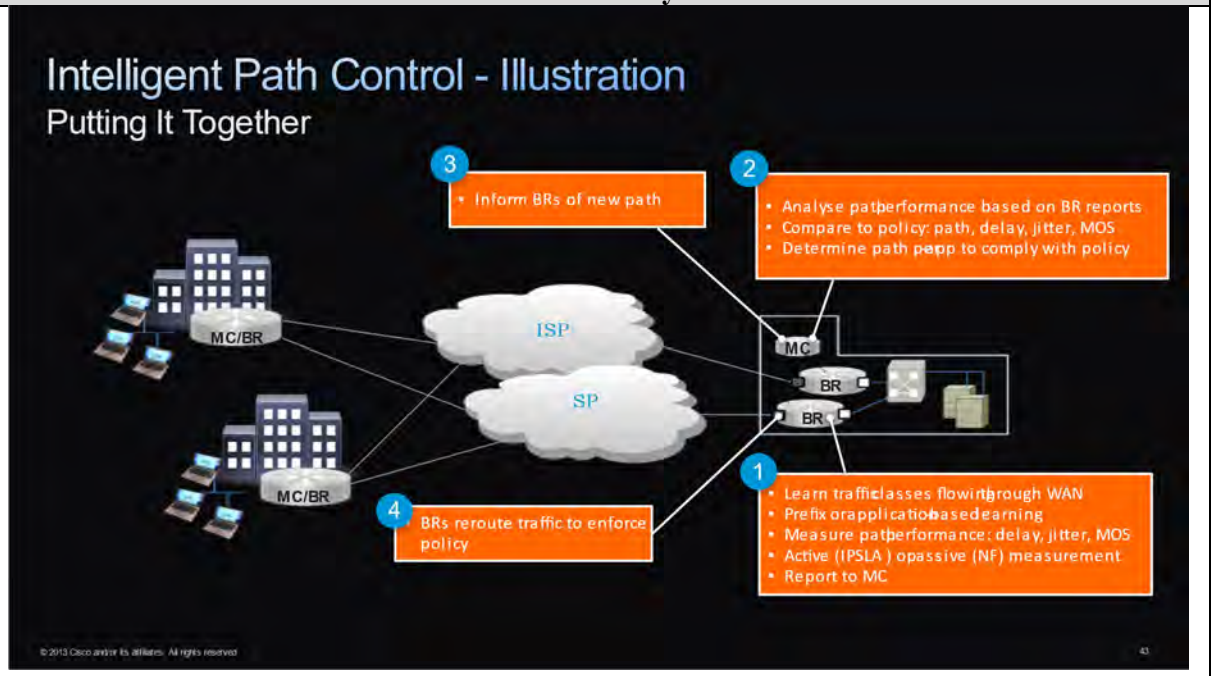
| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | |  |





Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 933 of 1100

| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|




Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Cisco IWAN System in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 1[f] of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references.

For example, Kempf discloses sending the packet from the network element to the controller or another table, in response to the packet matching the action in the flow table.

Kempf at [0044] ("FIG. 1 is a diagram of one embodiment of an example network with an OpenFlow switch, conforming to the OpenFlow 1.0 specification. The OpenFlow 1.0 protocol enables a controller 101 to connect to an OpenFlow 1.0 enabled switch 109 using a secure channel 103 and control a single forwarding table 107 in the switch 109. The controller 101 is an external software component executed by a remote computing device that enables a user to configure the Open-Flow 1.0 switch 109. The secure channel 103 can



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 934 of 1100

| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | | be provided by any type of network including a local area network (LAN) or a wide area network (WAN), such as the Internet.")<br><br>Kempf at [0045] ("FIG. 2 is a diagram illustrating one embodiment of the contents of a flow table entry. The forwarding table 107 is populated with entries consisting of a rule 201 defining matches for fields in packet headers; an action 203 associated to the flow match; and a collection of statistics 205 on the flow. When an incoming packet is received a lookup for a matching rule is made in the flow table 107. If the incoming packet matches a particular rule, the associated action defined in that flow table entry is performed on the packet.")<br><br>Kempf at [0046] ("A rule 201 contains key fields from several headers in the protocol stack, for example source and destination Ethernet MAC addresses, source and destination IP addresses, IP protocol type number, incoming and outgoing TCP or UDP port numbers. To define a flow, all the available matching fields may be used. But it is also possible to restrict the matching rule to a subset of the available fields by using wildcards for the unwanted fields.")<br><br>Kempf at [0047] ("The actions that are defined by the specification of OpenFlow 1.0 are Drop, which drops the matching packets; Forward, which forwards the packet to one or all outgoing ports, the incoming physical port itself, the controller via the secure channel, or the local networking stack (if it exists). OpenFlow 1.0 protocol data units (PDU s) are defined with a set of structures specified using the C programming language. Some of the more commonly used messages are: report switch configuration message; modify state messages (in-cluding a modify flow entry message and port modification message); read state messages, where while the system is running, the datapath may be queried about its current state using this message; and send packet message, which is used when the controller wishes to send a packet out through the datapath.")<br><br>Kempf at [0050] ("FIG. 4 illustrates one embodiment of the processing of packets through an OpenFlow 1.1 switched packet pro-cessing pipeline. A received packet is compared against each of the flow tables 401. After each flow table match, the actions are accumulated into an action set. If processing requires matching against another flow table, the actions in the matched rule include an action directing processing to the next table in the |

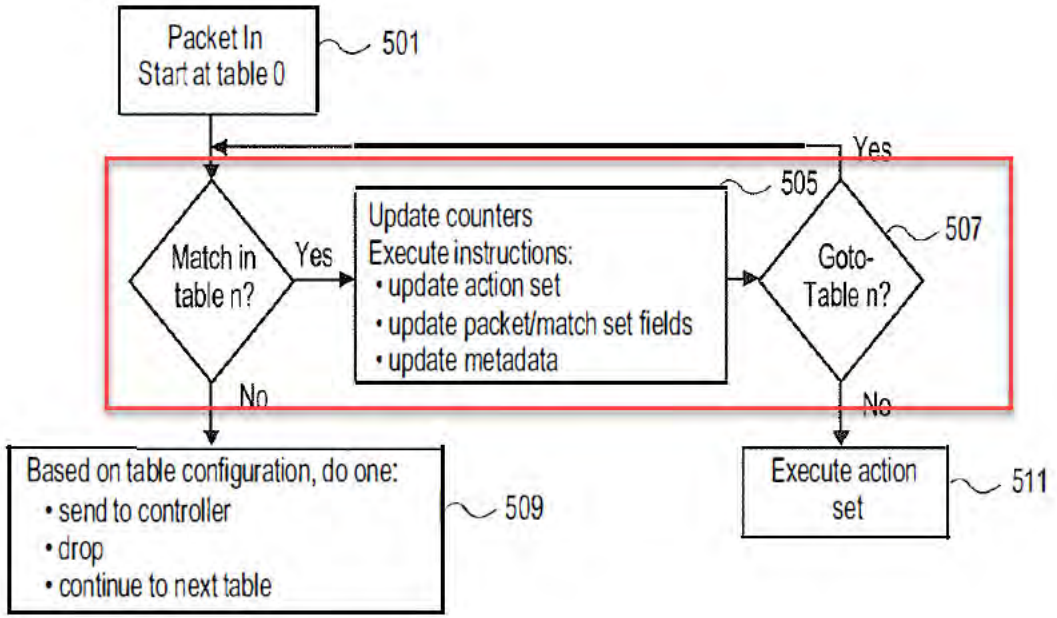


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 935 of 1100

| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | | pipeline. Absent the inclusion of an action in the set to execute all accumulated actions immediately, the actions are executed at the end 403 of the packet processing pipeline. An action allows the writing of data to a metadata register, which is carried along in the packet processing pipe-line like the packet header.")<br><br>Kempf at [0091] ("When a packet header matches a rule associated with the virtual port, the GTP TEID is written into the lower 32 bits of the metadata and the packet is directed to the virtual port. The virtual port calculates the hash of the TEID and looks up the tunnel header information in the tunnel header table. If no such tunnel information is present, the packet is forwarded to the controller with an error indication. Other-wise, the virtual port constructs a GTP tunnel header and encapsulates the packet. Any DSCP bits or VLAN priority bits are additionally set in the IP or MAC tunnel headers, and any VLAN tags or MPLS labels are pushed onto the packet. The encapsulated packet is forwarded out the physical port to which the virtual port is bound.")<br><br>Kempf at [0106] ("This encapsulates the packet and sends it to the OpenFlow controller.")<br><br>Kempf at Figure 5 (annotation added) |

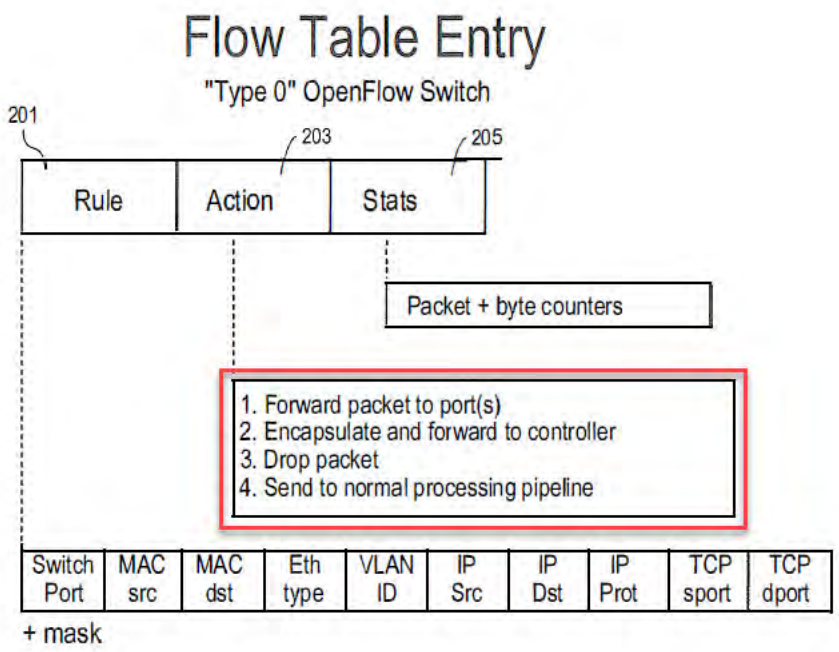


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 936 of 1100

| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | |  <br>FIG. 5<br>Kempf at Figure 2 (annotation added) |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 937 of 1100

| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | |  <br>FIG. 2<br><br>For example, Swenson discloses determining by the steering device monitors flows that match one or more signatures or criteria of the packet. Swenson further discloses that when a matching flow is detected the steering device forwards the packet to the network controller.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 938 of 1100

| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | | inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0028] ("In contrast to conventional inline TCP throughput monitoring devices that monitor every single data packets transmitted and received, the network controller 140 is an "out-of-band" computer server that interfaces with the steer-ing device 130 to selectively inspect user flows of interest. The network controller 140 may further identify user flows (e.g., among the flows of interest) for optimization. In one embodiment, the network controller 140 may be imple-mented at the steering device 130 to monitor traffic. In other embodiments, the network controller 140 is coupled to and communicates with the steering device 130 for traffic moni-toring and optimization. When queried by the steering device 130, the network controller 140 determines if a given network flow should be ignored, monitored further or optimized. Opti-mization of a flow is often decided at the beginning of the flow because it is rarely possible to switch to optimized content mid-stream once non-optimized content delivery has begun. However, the network controller 140 may determine that existing flows associated with a particular subscriber or other entity should be optimized. In turn, new flows ( e.g., resulting from seek requests in media, new media requests, resume after pause, etc.) determined to be associated with the entity may be optimized. The network controller 140 uses the net-work state as well as historical traffic data in its decision for monitoring and optimization. Knowledge on the current net-work state, such as congestion, deems critical when it comes to data optimization.")<br><br>Swenson at [0029] ("As a flow is sent to the network controller 140 for inspection, historical network traffic data stored at the net-work controller 140 may be searched. The historical network traffic data includes information such as subscriber informa-tion, the cell towers to which the user devices attached, rout-ers through which the traffic is passing, geography regions, the backhaul segments, and time-of-day of the flows. For example, in a mobile network, the cell tower to which a user device is attached can be most useful, since it is the location where most congestion occurs due to limited bandwidth and high cost of the radio access network infrastructure. The network controller 140 looks into the historical traffic data for the average of the bandwidth per user at the particular cell tower. The |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 939 of 1100

| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | | network controller 140 can then estimate the amount ofbandwidth or degree of congestion for the new flow based on the historical record.")<br><br>Swenson at [0038] ("Turning back to FIG. 1, the network controller 140 allows network operators to apply fine granular optimization policies to ensure high quality of experience (QoE) based on cell tower congestion, device types, subscriber profiles and service plans with lower hardware and software costs. The architecture of the network controller 140 provides an excel-lent fit for the net neutrality guideline of"reasonable network management", and better compliance to the copyright law (DMCA) than solutions that rely on long-term caching. Hav-ing the ability of monitoring network traffic on a per sub-scriber, per flow, or per video file basis, the network controller 140 also selectively monitors and optimizes only a subset of traffic that benefits from optimization the most, thus achiev-ing both scalability and efficiency for optimization at a com-petitive price-point. The core element of the network control-ler 140 lies in its mechanisms for congestion detection and mitigation, which allows optimization resources to be utilized in the most efficient and surgical manner.")<br><br>Swenson at [0039] ("Referring now to FIG. 3, it illustrates one embodi-ment of an example architecture of the network controller 140 for providing selective real-time network monitoring and subscriber identification. The network controller 140 com-prises a flow analyzer 312, a policy engine 314, a steering device interface 316, a video optimizer redirector 318, a flow cache 322, and a subscriber log 324. In other embodiments, the network controller 140 may include additional, fewer, or different components for various applications. Conventional components such as network interfaces, security functions, failover servers, management and network operations con-soles, and the like are not shown so as to not obscure the details of the system architecture.")<br><br>Swenson at [0045] ("The steering device interface 316 interacts with an external routing appliance, such as the steering device 130 to divert portions of the network traffic ( e.g., large object net-work flows). Existing routing appliances in most carrier net-works are designed to handle large amounts of network traf-fic. They are not, however, ideal devices to operate for monitoring and analysis individual flows. Through the steer-ing device interface 316, the network controller 140 may communicate with the external routing appliances, such as the steering device 130, to steer a portion of network traffic to the |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 940 of 1100

| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | | network controller 140 when certain conditions are met. Generally, network flows of interest to the network controller 140 contain larger media objects, such as videos and images. In one embodiment, the smaller flows, such as web page and text information, are not exchanged over the steering device interface 316.")<br><br>Swenson at [0059] ("In one embodiment, as the steering device 130 monitors network responses, it is looking for flows that match one or more signatures for video and images. When a match-ing flow is detected, the steering device 130 forwards the HTTP request and a portion of the HTTP response to the network controller 140 over the ICAP client interface 404. After receiving the request and the portion of response at the ICAP server interface 406, the flow analyzer 312 of the net-work controller 140 performs a deep flow inspection to deter-mine if the flow is worth bandwidth monitoring and/or user detection. For example, the flow inspection performed by the flow analyzer 312 may determine if the flow indeed contains large or medium object ( e.g., larger than 50 kB), and/or if the source IP address of the flow is from a user or a group of users that are required to be monitored by policies. The flow ana-lyzer 312 may also determine if the flow needs to be opti-mized based on historical flow statistical data.")<br><br>Swenson at [0060] ("If the flow is deemed of interest, the steering device 130 is notified to steer the flow through the network controller 140. This is known as the "continue" working mode for bandwidth monitoring. In the "continue" mode, the network controller 140 interfaces with the steering device 130 to func-tion, on-demand, as a traditional inline network element for flows deemed of interest. Thus, the network controller 140 ingests the network flow for inspection and subsequently forwards the network flow on the network response path. For example, for this particular flow, the origin server 160 responds to the user request by sending video or images over the network link 413 to the steering device 130, which for-wards the video or images to the network controller 140 over a network link 414. After the network controller 140 updates the flow statistics, the video or images are returned to the steering device 130 over a network link 415, which transmits the video or images to the user device 110 over the network link 416.")<br><br>Swenson at [0065] ("FIG. 5 is a block diagram illustrating an example event trace of "continue" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user |

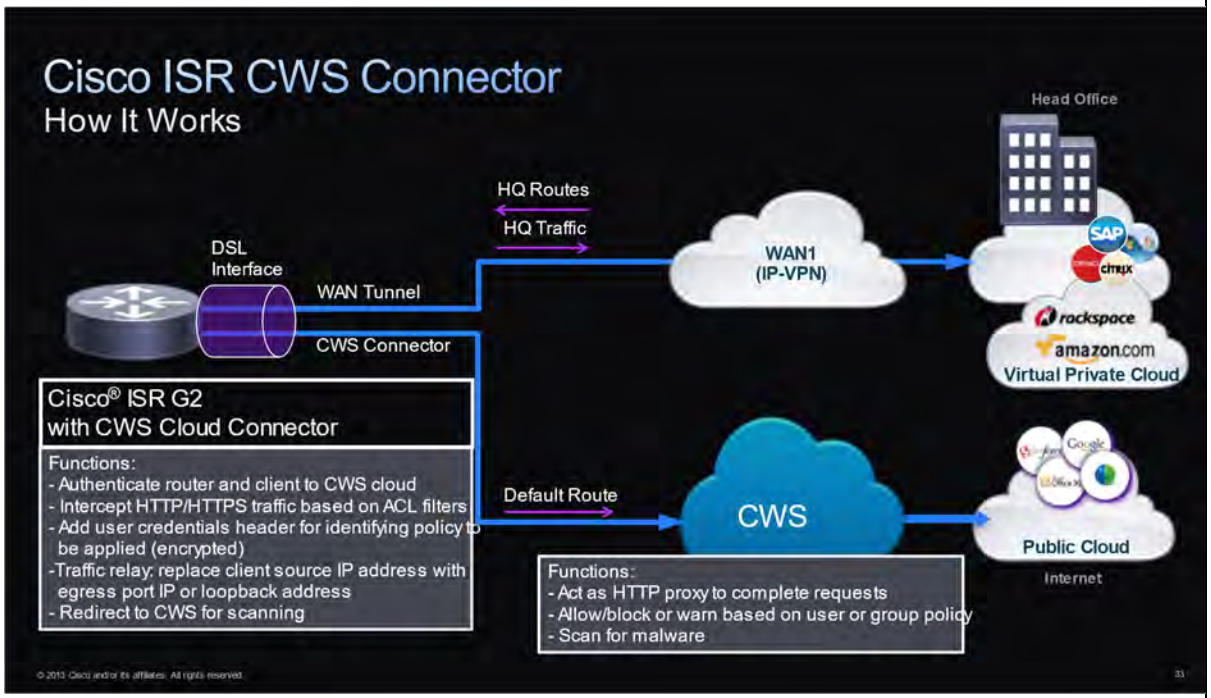


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 941 of 1100

| No. | '111 Patent Claim 1 | Cisco IWAN System |
|---|---|---|
| | | device 110 initiates an HTTP GET request 512 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 512 to the intended origin server 160 and receives a response 514 back from the origin server 160. The steering device 130 then sends an ICAP request message 516 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 518. Upon receiving the instruc-tion, the steering device 130 re-writes the response 514 to an HTTP redirect response 520, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 520 directly to the user device 110. In case the flow dose not contain video or image objects, or the network controller 140 determines not to monitor the flow, the steering device 13 0 would forward the response to the user device 110.")<br><br>Swenson at [0069] ("FIG. 6 is a block diagram illustrating an example event trace of "counting" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 612 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 612 to the intended origin server 160 and receives a response 614 back from the origin server 160. The steering device 130 then sends an ICAP request message 616 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 618. Upon receiving the instruc-tion, the steering device 130 re-writes the response 614 to an HTTP redirect response 620, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 620 directly to the user device 110. In case the flow dose not contain video or image objects that need to be redirected, the steering device 130 would forward the response to the user device 110.") |

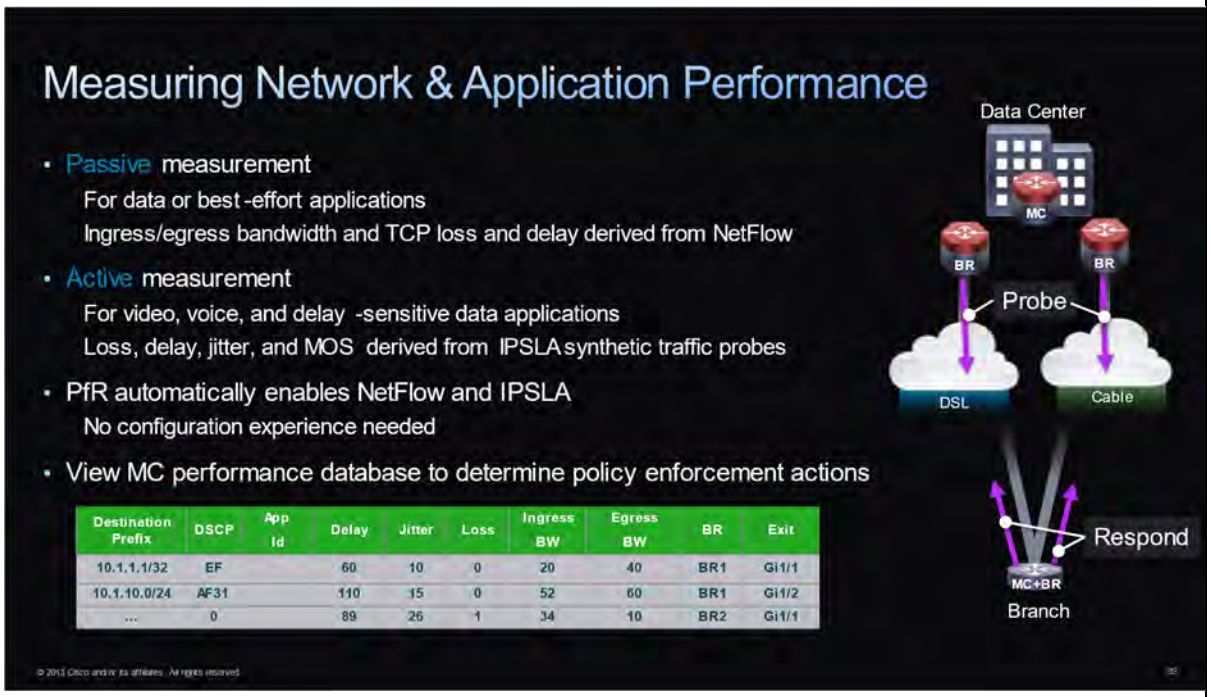


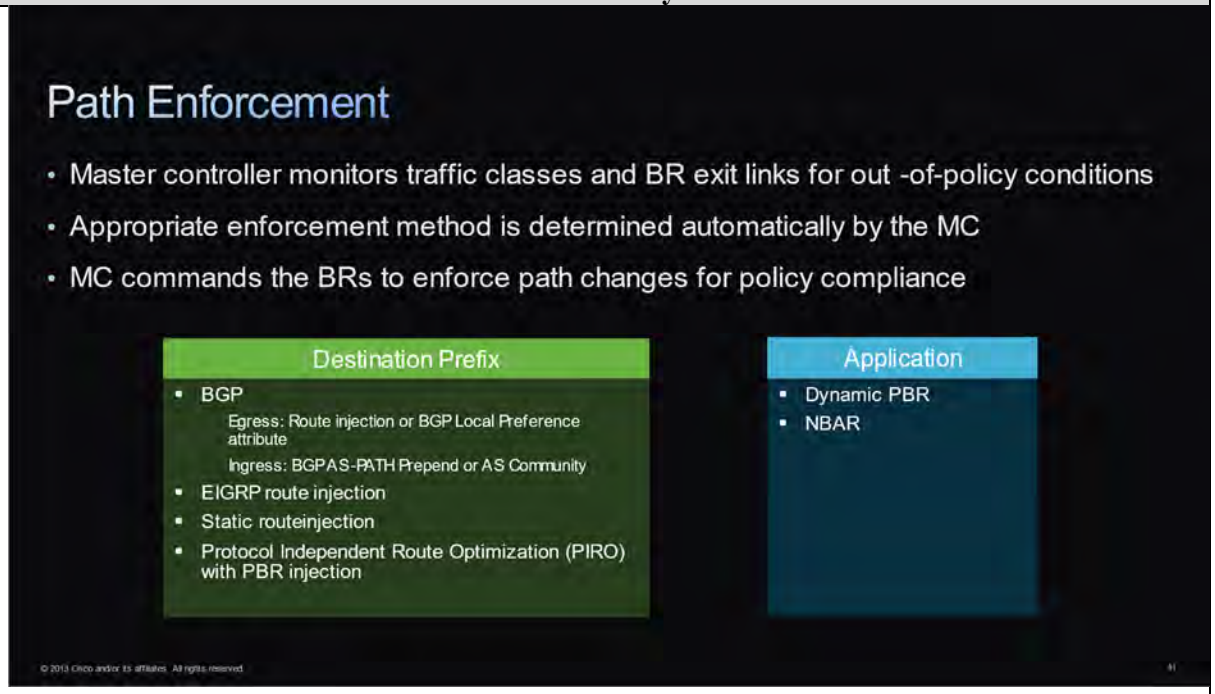
Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 942 of 1100

| No. | '111 Patent Claim 2 | Cisco IWAN System |
|---|---|---|
| 2[a] | The method according to claim 1, wherein the instruction is 'probe', 'mirror', or 'terminate' instruction, and | Cisco IWAN System discloses the method according to claim 1, wherein the instruction is 'probe', 'mirror', or 'terminate' instruction.<br><br>For example, Cisco IWAN System discloses traffic policies sent by the hub/master controller to the branch devices/border routers, including probing, forwarding, and blocking policies. A person of ordinary skill in the art would understand that such traffic policies could include any number of definitions including 'probe', 'mirror', and 'terminate'. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Cisco IWAN System is found to not meet this limitation, wherein the instruction is 'probe', 'mirror', or 'terminate' instruction would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>Cisco IWAN<br>  <br><br><br>Cisco IWAN - Uncompromised Experience |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 943 of 1100

| No. | '111 Patent Claim 2 | Cisco IWAN System |
|---|---|---|
| | |  |

| No. | '111 Patent Claim 2 | Cisco IWAN System |
|---|---|---|
| | |  <br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Cisco IWAN System in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 2[a] of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references.<br><br>For example, Chua discloses programming network nodes with redirecting, mirroring, and blocking programmed actions.<br><br>Chua at 7:28-54 ("SDN controller 112 may receive data as input from service devices 116. For example, SDN controller 112 may be con-figured to receive data from an intrusion detection system (IDS) device, a Denial of Service (DoS) device, a Distributed Denial of Service (DDoS) device, an intrusion prevention system (IPS) device, or the like. Based on this information, SDN controller 112 may make network enforcement decisions for specific |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 945 of 1100

| No. | '111 Patent Claim 2 | Cisco IWAN System |
|---|---|---|
| | | traffic flows. That is, SDN controller 112 may program network devices of SDN 106 to perform pro-grammed actions on packets of a packet flow based on this data. Such programmed actions may include:<br><br>Allow-explicitly allow a certain network flow to proceed to its destination<br>Block-explicitly block a certain flow from traversing SDN 106<br>Mirror-allow the traffic, but send a copy of the traffic for deeper inspection or recording to, e.g., one of service devices 116<br>Redirect-redirect the traffic to another network (such as a honeypot device or other device of service devices 116) for either inspection or to keep a potential hacker 'busy' to determine if there is a real security threat.<br>Transform-modify or translate values of headers of packets in the network flow<br>Encapsulate-encapsulate packets in the network flow with a particular header")<br><br>Chua at 28:7-32 ("In addition, SDN controller 112 may configure the service device to send service-related data to one or more network devices (334). The service-related data may cause the net-work devices to change a path along which the packet is forwarded. For example, when the service device is a security device (e.g., a firewall or an IDS), if the security device determines that one or more packets of a packet flow are malicious, the security device may send service data indicat-ing that the packet flow includes malicious data. SDN con-troller 112 may program the network devices of the SDN to perform a programmed action based on the service-related data (336). For example, SDN controller 112 may program network devices to, in response to an indication that packets of a packet flow include malicious data, forward packets of the packet flow to a destination of the packet flow, forward packets of malicious packet flows to a collection device for further analysis, cause network devices to drop packets of the malicious packet flows, send a close session message to devices from which packets of the malicious packet flows were received, block the packets of the packet flow, mirror copies of the packets of the packet flow to a second service device while forwarding the packets of the packet flow to the destination of the packet flow, redirect the packets of the packet flow to a third service device, transform one or more values of headers of the packets, and/or encapsulate the pack-ets with a particular header, or other such actions.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 946 of 1100

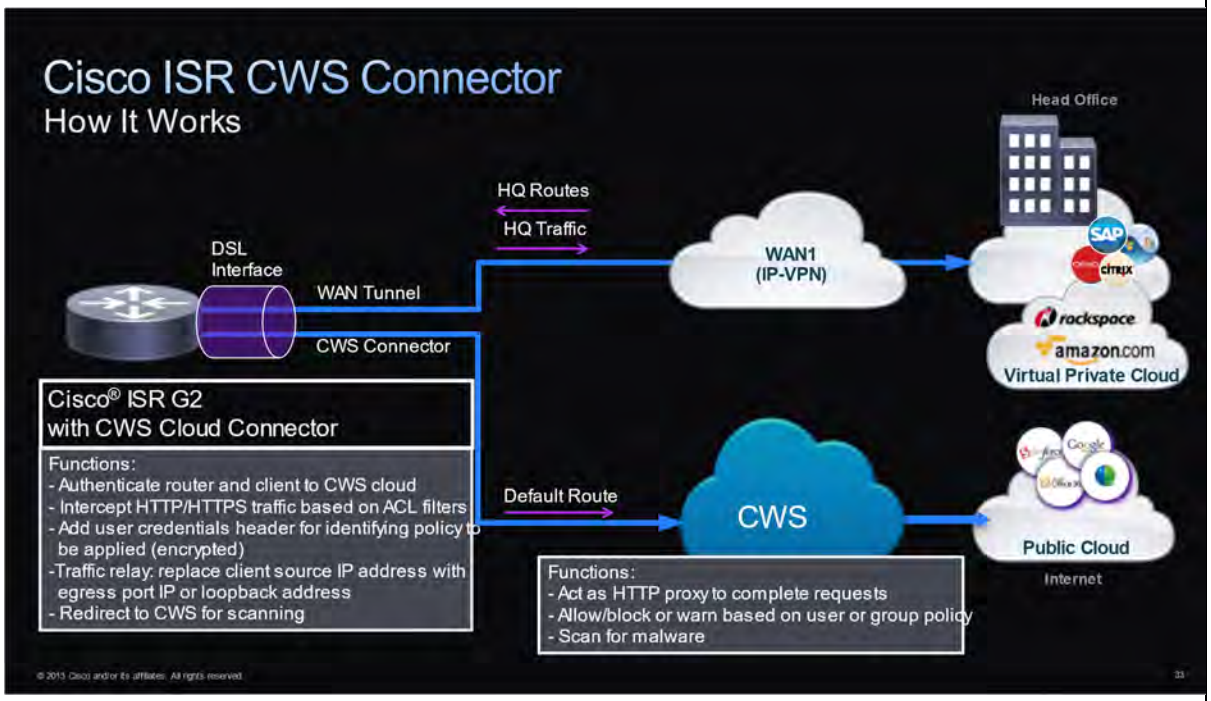
| No. | '111 Patent Claim 2 | Cisco IWAN System |
|---|---|---|
| | | As another example, Copeland discloses probing, copying, and terminating rules configured on the network device.<br><br>Copeland at [0057] ("In accordance with an aspect of the invention, a flow is considered terminated after a predetermined period of time has elapsed on a particular connection or port. For example, if HTTP Web traffic on port 80 ceases for a predetermined period of time, but other traffic begins to occur on port 80 after the expiration of that predetermined time period, it is considered that a new flow has begun, and the system responds accordingly to assign a new flow number and track the statistics and characteristics thereof. In the disclosed embodiment, the predetermined time period is 330 seconds, but those skilled in the art will understand that this time is arbitrary and may be heuristically adjusted.")<br><br>Copeland at [0082] ("Following the reserved field, the next 6 bits are a series of one-bit flags, shown in FIG. 2 as flags U, A, P, R, S, F. The first flag is the urgent flag (U). If the U flag is set, it indicates that the urgent pointer is valid and points to urgent data that should be acted upon as soon as possible. The next flag is the A ( or ACK or "acknowledgment") flag. The ACK flag indicates that an acknowledgment number is valid, and acknowledges that data has been received. The next flag, the push (P) flag, tells the receiving end to push all buffered data to the receiving application. The reset (R) flag is the following flag, which terminates both ends of the TCP connection. Next, the S (or SYN for "synchronize") flag is set in the initial packet of a TCP connection where both ends have to synchronize their TCP buffers. Following the SYN flag is the F (for FIN or "finish") flag. This flag signifies that the sending end of the communication and the host will not send any more data but still may acknowledge data that is received.")<br><br>Copeland at [0093] ("As illustrated, when Hostl terminates its end of the session, it sends a packet with the FIN and ACK flags set. The FIN flag informs Host2 that Hostl will send no more data. The ACK flag acknowledges the last data received by Hostl by informing Host2 of the next sequence number it expects to receive.")<br><br>Copeland at [0095] ("When Host 2 is ready to terminate the session, it sends its own packet with the FIN and ACK flags set. Hostl responds that it has received the final packet with an ACK packet providing to Host2 an acknowledgment number one greater than the sequence number provided in the FIN-ACK packet of Host2.") |


~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 947 of 1100

| No. | '111 Patent Claim 2 | Cisco IWAN System |
|---|---|---|
| | | Copeland at [0099] ("As another example, if a particular host sends a large number of SYN packets to a target host and in response receives numerous R packets from the targeted host, a potential TCP probe is indicated. Likewise, numerous UDP packets sent from one host to a targeted host and numerous ICMP "port unavailable" packets received from the targeted host indicate a potential UDP probe. A stealth probe is indicated by multiple packets from the same source port number sent to different port numbers on a targeted host.")<br><br>Copeland at [0107] ("A flow is terminated if no communications occur between the two IP addresses and the one low port ( e.g. port 80) for 330 seconds. Most Web browsers or a TCP connec-tion send a reset packet (i.e. a packet with the R flag set) if no communications are sent or received for 5 minutes. An analysis can determine if the flow is abnormal or not for HTTP communications.")<br><br>Copeland at [0123] ("Flow processing is done for TCP and UDP packets, and the port numbers in the transport layer header are used to identify the flow record to be updated. For ICMP packets that constitute rejections of a packet, the copy of the rejected packet in the ICMP data field is used to identify the IP addresses and port numbers of the corresponding flow.")<br><br>Copeland at [0145] ("A list IP of addresses contacted or probed by each host can be maintained. When this list indicates that more than a threshold number of other hosts (e.g., 8) have been contacted in the same subnet, CI is added to the to the host and a bit in the host record is set to indicate that the host has received CI for "address scanning." Note that the number of hosts to designate a scan is not required to be a fixed value, but could be adjusted based on the sample rate or other means to enhance the accuracy making the number of hosts scanned "statistically significant". These and other values of concern index are shown for non-flow based events in FIG. 7.")<br><br>Copeland at [0158] ("Flow processing is done for TCP and UDP packets, and the port numbers in the transport layer header are used to identify the flow record to be updated. For ICMP packets that constitute rejections of a packet, the copy of the rejected packet in the ICMP data field is used to identify the IP addresses and port numbers of the corresponding flow.") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 948 of 1100

| No. | '111 Patent Claim 2 | Cisco IWAN System |
|---|---|---|
| | | |
| 2[b] | upon receiving by the network node the 'terminate' instruction, the method further comprising blocking, by the network node, the packet from being sent to the second entity and to the controller. | Cisco IWAN System discloses upon receiving by the network node the 'terminate' instruction, the method further comprising blocking, by the network node, the packet from being sent to the second entity and to the controller.<br><br>For example, Cisco IWAN System discloses blocking or preventing the forwarding of traffic flows by the branch device/border router to the intended data center device or hub/master controller based on a particular traffic policy definition.<br><br>Cisco IWAN<br>  |

| No. | '111 Patent Claim 3 | Cisco IWAN System |
|---|---|---|
| 3[a] | The method according to claim 1, wherein the | Cisco IWAN System discloses the method according to claim 1, wherein the instruction is a 'probe', a 'mirror', or a 'terminate' instruction. |

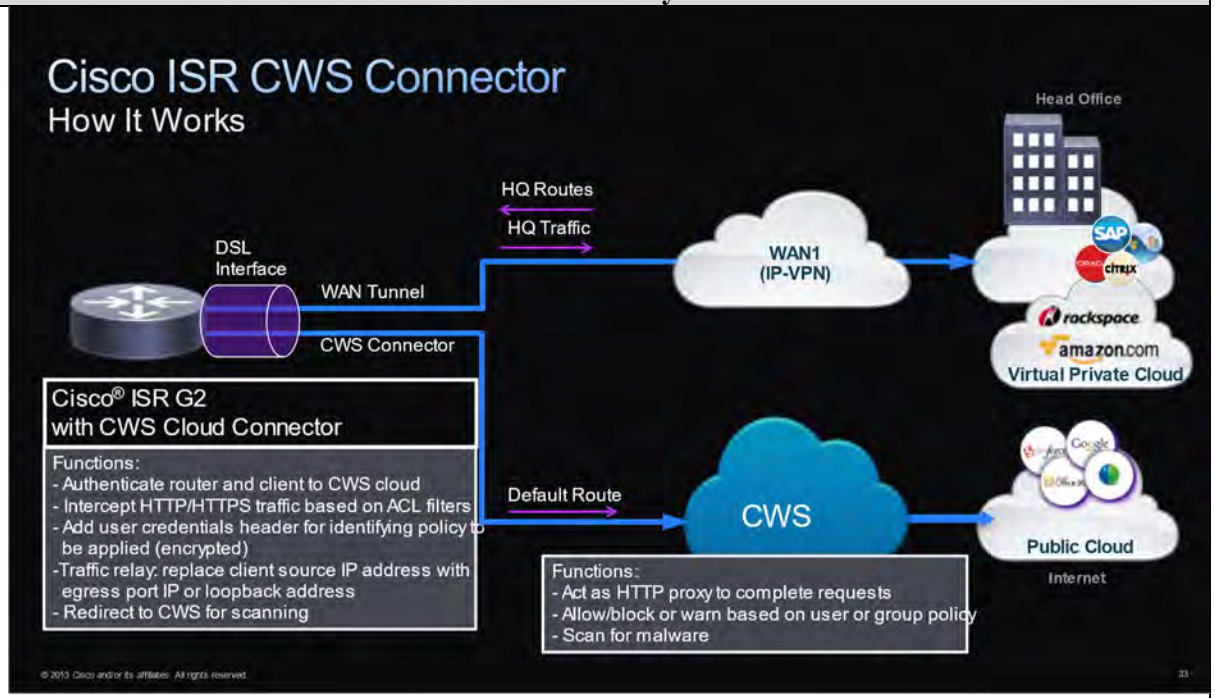


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 949 of 1100

| No. | '111 Patent Claim 3 | Cisco IWAN System |
|---|---|---|
| | instruction is a 'probe', a 'mirror', or a 'terminate' instruction, and | *See supra* at 2(a). |
| 3[b] | upon receiving by the network node the 'mirror' instruction and responsive to the packet satisfying the criterion, the method further comprising sending the packet, by the network node, to the second entity and to the controller. | Cisco IWAN System discloses upon receiving by the network node the 'mirror' instruction and responsive to the packet satisfying the criterion, the 4[b]method further comprising sending the packet, by the network node, to the second entity and to the controller.<br><br>For example, Cisco IWAN System discloses traffic policies sent by the hub/master controller to the branch devices/border routers, including forwarding policies. A person of ordinary skill in the art would understand that such traffic policies could include any number of definitions including a mirror instruction in which, responsive to determining a particular traffic class of the traffic flow, forwarding the traffic to the intended data center device as well as the hub/master controller. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Cisco IWAN System is found to not meet this limitation, upon receiving by the network node the 'mirror' instruction and responsive to the packet satisfying the criterion, method further comprising sending the packet, by the network node, to the second entity and to the controller would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>Cisco IWAN |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 950 of 1100

| No. | '111 Patent Claim 3 | Cisco IWAN System |
|---|---|---|
| | |  <br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Cisco IWAN System in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 3[b] of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references.<br><br>For example, Chua discloses a mirror program in response to an indication based on the packet header in which the network devices mirror copies of the packets of the packet flow to a second service device while forwarding the packets of the packet flow to the destination of the packet flow.<br><br>Chua at 7:28-54 ("SDN controller 112 may receive data as input from service devices 116. For example, SDN controller 112 may be con-figured to receive data from an intrusion detection system (IDS) device, a Denial of Service (DoS) device, a Distributed Denial of |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 951 of 1100

| No. | '111 Patent Claim 3 | Cisco IWAN System |
|---|---|---|
| | | Service (DDoS) device, an intrusion prevention system (IPS) device, or the like. Based on this information, SDN controller 112 may make network enforcement decisions for specific traffic flows. That is, SDN controller 112 may program network devices of SDN 106 to perform pro-grammed actions on packets of a packet flow based on this data. Such programmed actions may include:<br><br>Allow-explicitly allow a certain network flow to proceed to its destination<br>Block-explicitly block a certain flow from traversing SDN 106<br>Mirror-allow the traffic, but send a copy of the traffic for deeper inspection or recording to, e.g., one of service devices 116<br>Redirect-redirect the traffic to another network (such as a honeypot device or other device of service devices 116) for either inspection or to keep a potential hacker 'busy' to determine if there is a real security threat.<br>Transform-modify or translate values of headers of packets in the network flow<br>Encapsulate-encapsulate packets in the network flow with a particular header")<br><br>Chua at 16:23-44 ("More particularly, control unit 130 may configure any of service devices 116 to send data representative of a particular event to SDN controller 112, and control unit 130 may auto-matically reprogram one or more network devices of SDN 106 in response to such data. For example, security monitor-ing applications of service devices 116 may determine that a specific source port, destination port, source IP address, des-tination IP address, or the like should be acted upon. Alter-natively, security monitoring applications may determine that, due to content or deep packet inspection, a specific type of traffic is malicious and should be blocked. In either case, the corresponding one of service devices 116 may send a message to SDN controller 112 representative of these deter-minations. As yet another example, a network performance device may monitor various performance metrics, such as latency, jitter, packet loss, or the like, and provide feedback data to SDN controller 112 based on these metrics. SDN controller 112 may respond by programming network devices of SDN 106 to perform a programmed action, such as allowing corresponding traffic, blocking corresponding traf-fic, mirroring corresponding traffic, redirecting correspond-ing traffic.")<br><br>Chua at 28:7-32 ("In addition, SDN controller 112 may configure the service device to send service-related data to one or more network devices (334). The service-related data may |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 952 of 1100

| No. | '111 Patent Claim 3 | Cisco IWAN System |
|---|---|---|
| | | cause the net-work devices to change a path along which the packet is forwarded. For example, when the service device is a security device (e.g., a firewall or an IDS), if the security device determines that one or more packets of a packet flow are malicious, the security device may send service data indicat-ing that the packet flow includes malicious data. SDN con-troller 112 may program the network devices of the SDN to perform a programmed action based on the service-related data (336). For example, SDN controller 112 may program network devices to, in response to an indication that packets of a packet flow include malicious data, forward packets of the packet flow to a destination of the packet flow, forward packets of malicious packet flows to a collection device for further analysis, cause network devices to drop packets of the malicious packet flows, send a close session message to devices from which packets of the malicious packet flows were received, block the packets of the packet flow, mirror copies of the packets of the packet flow to a second service device while forwarding the packets of the packet flow to the destination of the packet flow, redirect the packets of the packet flow to a third service device, transform one or more values of headers of the packets, and/or encapsulate the pack-ets with a particular header, or other such actions.")<br><br>As another example, Swenson discloses a counting mode instructed by the network controller to the steering device for monitoring and optimizing, in which the steering device forwards the packet flow to the user device/origin server and at the same time, sending the packet flow to the network controller.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 953 of 1100

| No. | '111 Patent Claim 3 | Cisco IWAN System |
|---|---|---|
| | | Swenson at [0028] ("In contrast to conventional inline TCP throughput monitoring devices that monitor every single data packets transmitted and received, the network controller 140 is an "out-of-band" computer server that interfaces with the steer-ing device 130 to selectively inspect user flows of interest. The network controller 140 may further identify user flows (e.g., among the flows of interest) for optimization. In one embodiment, the network controller 140 may be imple-mented at the steering device 130 to monitor traffic. In other embodiments, the network controller 140 is coupled to and communicates with the steering device 130 for traffic moni-toring and optimization. When queried by the steering device 130, the network controller 140 determines if a given network flow should be ignored, monitored further or optimized. Opti-mization of a flow is often decided at the beginning of the flow because it is rarely possible to switch to optimized content mid-stream once non-optimized content delivery has begun. However, the network controller 140 may determine that existing flows associated with a particular subscriber or other entity should be optimized. In turn, new flows ( e.g., resulting from seek requests in media, new media requests, resume after pause, etc.) determined to be associated with the entity may be optimized. The network controller 140 uses the net-work state as well as historical traffic data in its decision for monitoring and optimization. Knowledge on the current net-work state, such as congestion, deems critical when it comes to data optimization.")<br><br>Swenson at [0059] ("In one embodiment, as the steering device 130 monitors network responses, it is looking for flows that match one or more signatures for video and images. When a match-ing flow is detected, the steering device 130 forwards the HTTP request and a portion of the HTTP response to the network controller 140 over the ICAP client interface 404. After receiving the request and the portion of response at the ICAP server interface 406, the flow analyzer 312 of the net-work controller 140 performs a deep flow inspection to deter-mine if the flow is worth bandwidth monitoring and/or user detection. For example, the flow inspection performed by the flow analyzer 312 may determine if the flow indeed contains large or medium object ( e.g., larger than 50 kB), and/or if the source IP address of the flow is from a user or a group of users that are required to be monitored by policies. The flow ana-lyzer 312 may also determine if the flow needs to be opti-mized based on historical flow statistical data.")<br><br>Swenson at [0064] ("Similar to the "continue" mode, after receiving the initial HTTP messages of a flow and determining to monitor the flow, the network controller 140 notify |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 954 of 1100

| No. | ’111 Patent Claim 3 | Cisco IWAN System |
|---|---|---|
| | | the steering device 130 to work in a "counting" mode for bandwidth monitoring. In contrast to the "continue" mode, when a matching flow is detected for "counting" mode, the steering device 130 for-wards the HTTP response directly to the user device 110. While at the same time, the steering device 130 send a cus-tomized ICAP message to the network controller 140 over the network link 425. In one embodiment, the customized ICAP message contains the HTTP request and response headers, as well as a count of payload size of the current flow. After updating the flow statistics, the network controller 140 may acknowledge the gateway over the network line 426. In the "counting" mode, the network controller 140 does not join the network response path as an inline network element, but simply listens to the counting of flow size. The benefit of the "counting" mode is to off-load the network controller 140 from ingesting and forwarding the network flow on the net-work response path, while still enabling the detection of con-gestions and estimation of bandwidth associated with the flows of interest.”)<br><br>Swenson at [0071] (“After receiving the request, the video optimizer 150 forwards the video HTTP GET requests 622 to the origin server 160 and in return, receives a video file 624 from the origin server 160. The video optimizer 150 transcodes the video file to a format usable by the client device 110 based on network bandwidth available to the user device 110. The optimized video 626 is then transmitted from the video opti-mizer 150 to the steering device 130. In one embodiment, the steering device 130 intercepts the optimized video 626. The steering device 130 will then send an ICAP request to the network controller 140 for inspection. The network controller 140 deems this flow to be monitored and sends ICAP response 630. The steering device 130 then allows the flow to go through to the user device 110. The steering device 130 next sends periodic ICAP "counting" updates 632 to the network controller 140 until the flow completes. As such, the client receives the optimized video 626 for substantially real-time playback on an application executing on the user device 110.”)<br>Swenson at [0072] (“In one embodiment, if the video optimizer 150 failed to retrieve user requested video file from the origin server 160, the video optimizer 150 appends a "do not transcode" flag to the HTTP redirect request and returned to the user device 110, which re-sends the request out over the network to the origin server 160. The origin server 160 responds appropriately to the request by sending back video 624, which is intercepted by the steering device 130 only. The steering device 130 forwards the video to the user device 110 |



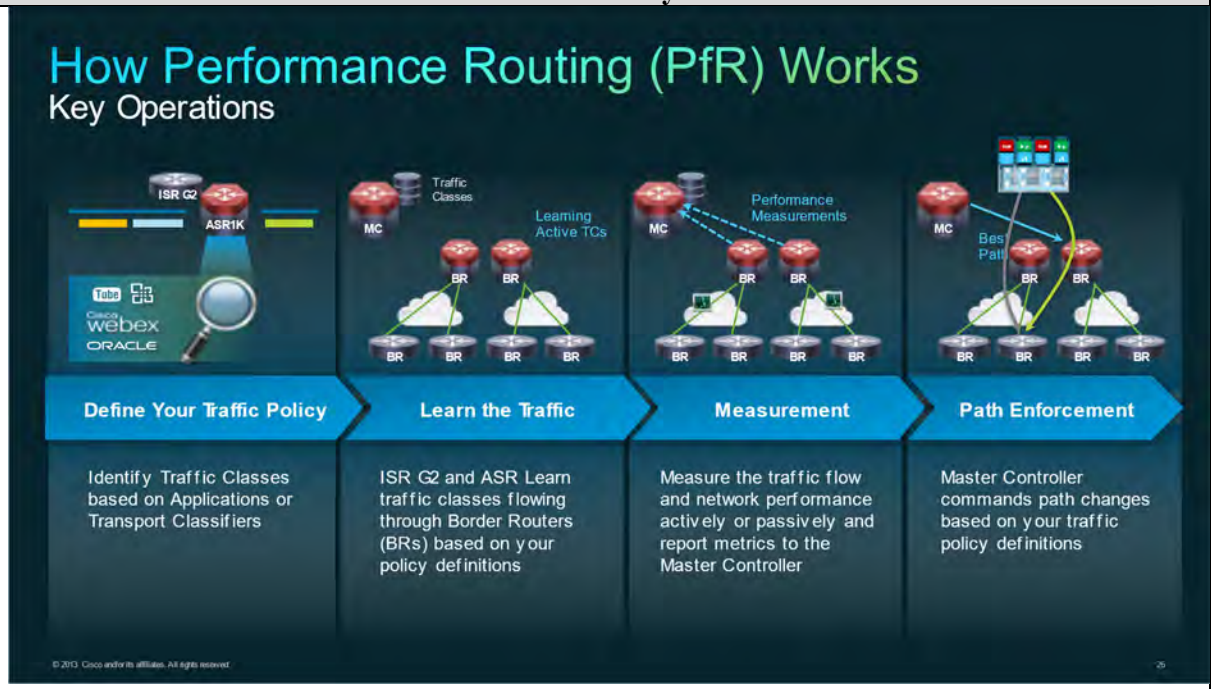
Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 955 of 1100

| No. | '111 Patent Claim 3 | Cisco IWAN System |
|---|---|---|
| | | and at the same time reports the flow size to the network controller 140 for monitoring purpose.") |

| No. | '111 Patent Claim 4 | Cisco IWAN System |
|---|---|---|
| 4[a] | The method according to claim 1, wherein the instruction is 'probe', 'mirror', or 'terminate' instruction, and | Cisco IWAN System discloses the method according to claim 1, wherein the instruction is 'probe', 'mirror', or 'terminate' instruction.<br><br>*See supra* at 2(a). |
| 4[b] | upon receiving by the network node the 'probe' instruction and responsive to the packet satisfying the criterion, the method further comprising: sending the packet, by the network node, to the controller; | Cisco IWAN System discloses upon receiving by the network node the 'probe' instruction and responsive to the packet satisfying the criterion, the method further comprising: sending the packet, by the network node, to the controller.<br><br>For example, Cisco IWAN System discloses a probe policy enforced by the branch device/border router in which traffic flow metrics are sent to the hub/master controller upon determining the traffic belongs to a particular traffic class. A person of ordinary skill in the art would understand that such traffic policies could include any number of definitions in which, responsive to determining a particular traffic class of the traffic flow, forwards the traffic to the hub/master controller. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Cisco IWAN System is found to not meet this limitation, upon receiving by the network node the 'probe' instruction and responsive to the packet satisfying the criterion, the method further comprising: sending the packet, by the network node, to the controller would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>Cisco Next Generation |

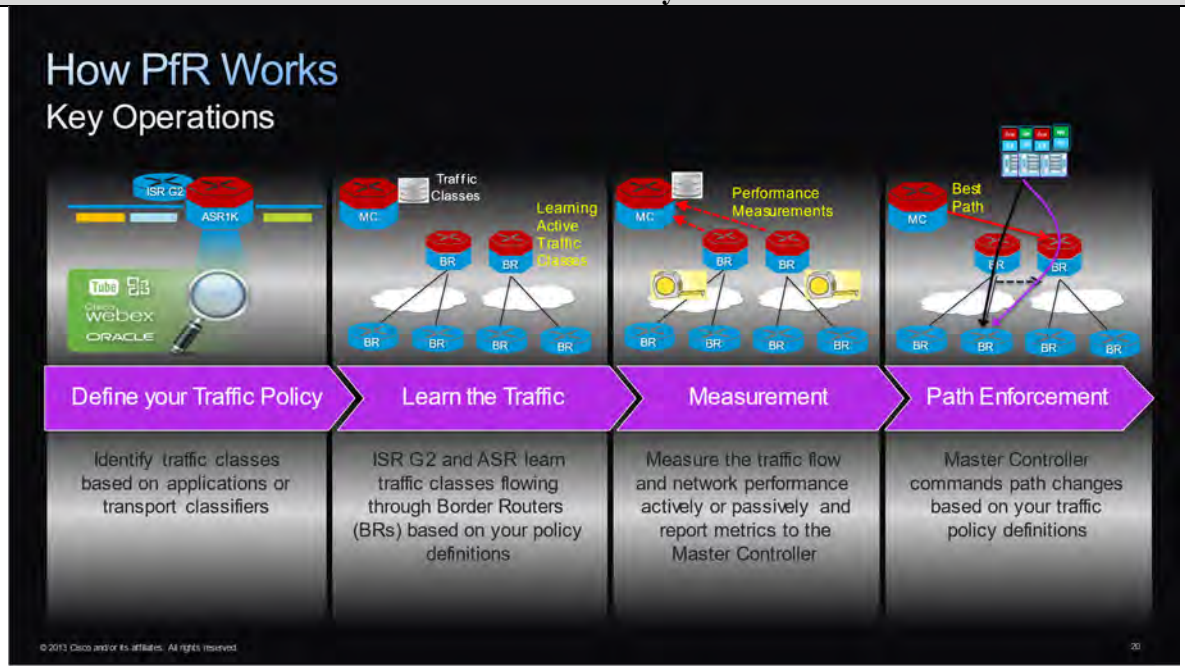


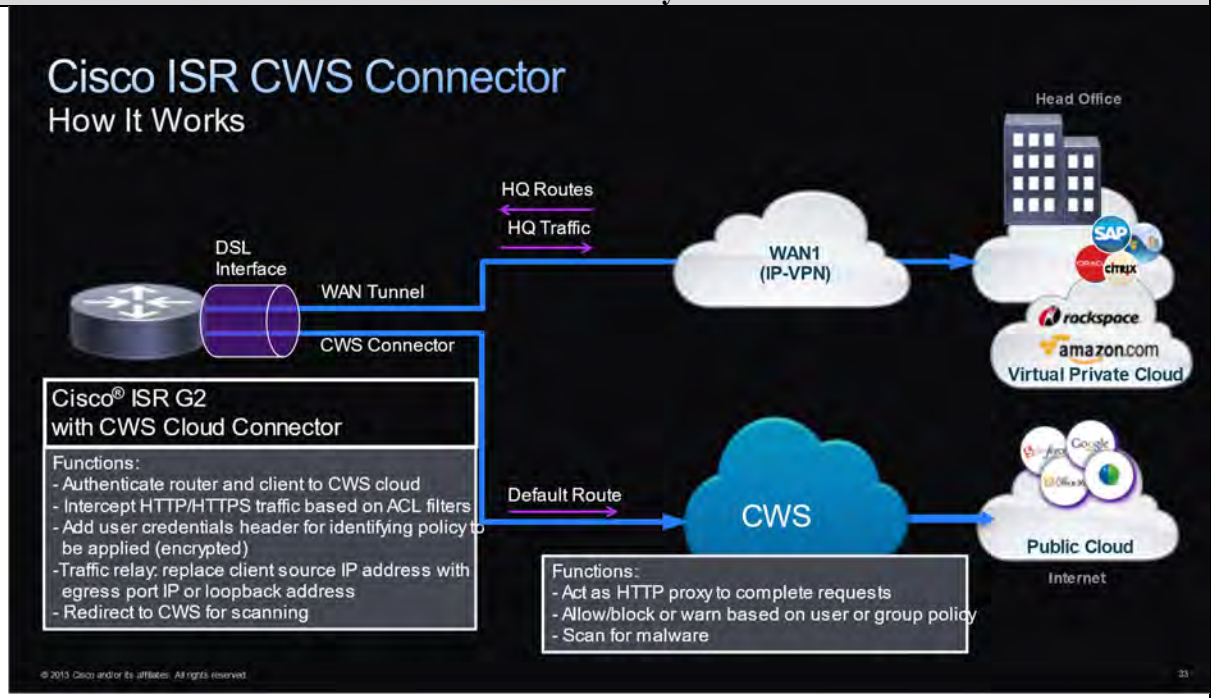
Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 956 of 1100

| No. | '111 Patent Claim 4 | Cisco IWAN System |
|---|---|---|
| | |  <br>Cisco IWAN |



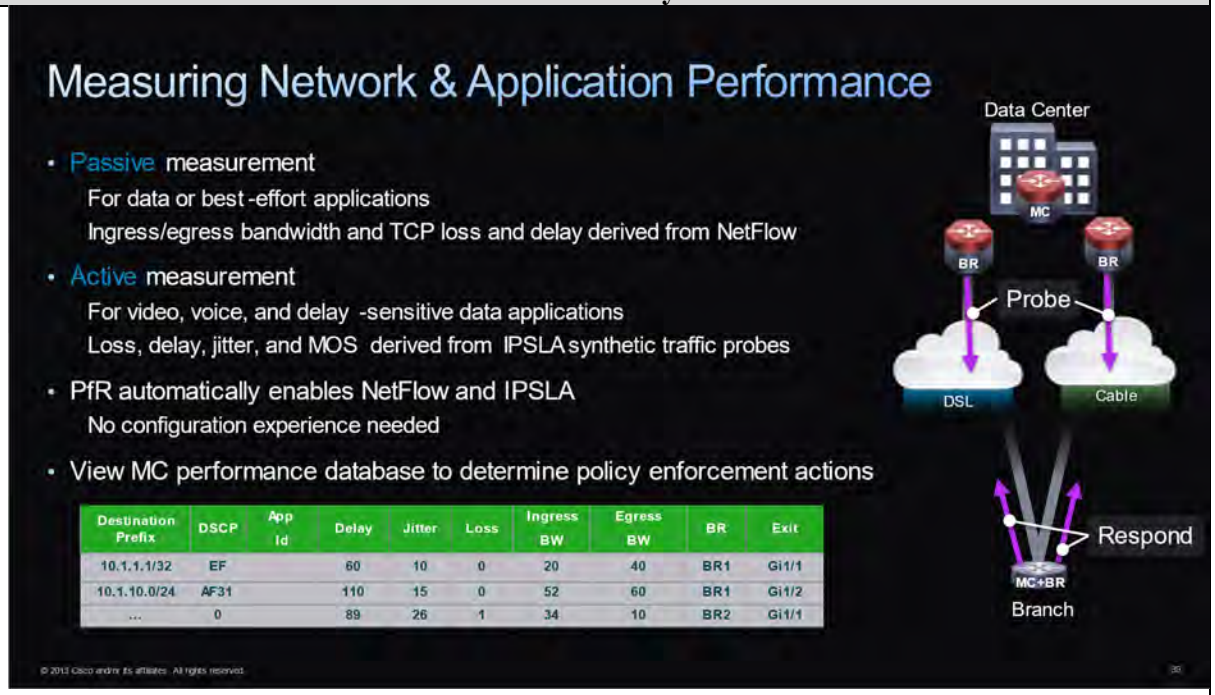
Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 957 of 1100

| No. | '111 Patent Claim 4 | Cisco IWAN System |
|---|---|---|
| | |  <br>Cisco IWAN |

| No. | '111 Patent Claim 4 | Cisco IWAN System |
|---|---|---|
| | |  <br>Cisco IWAN - Uncompromised Experience |



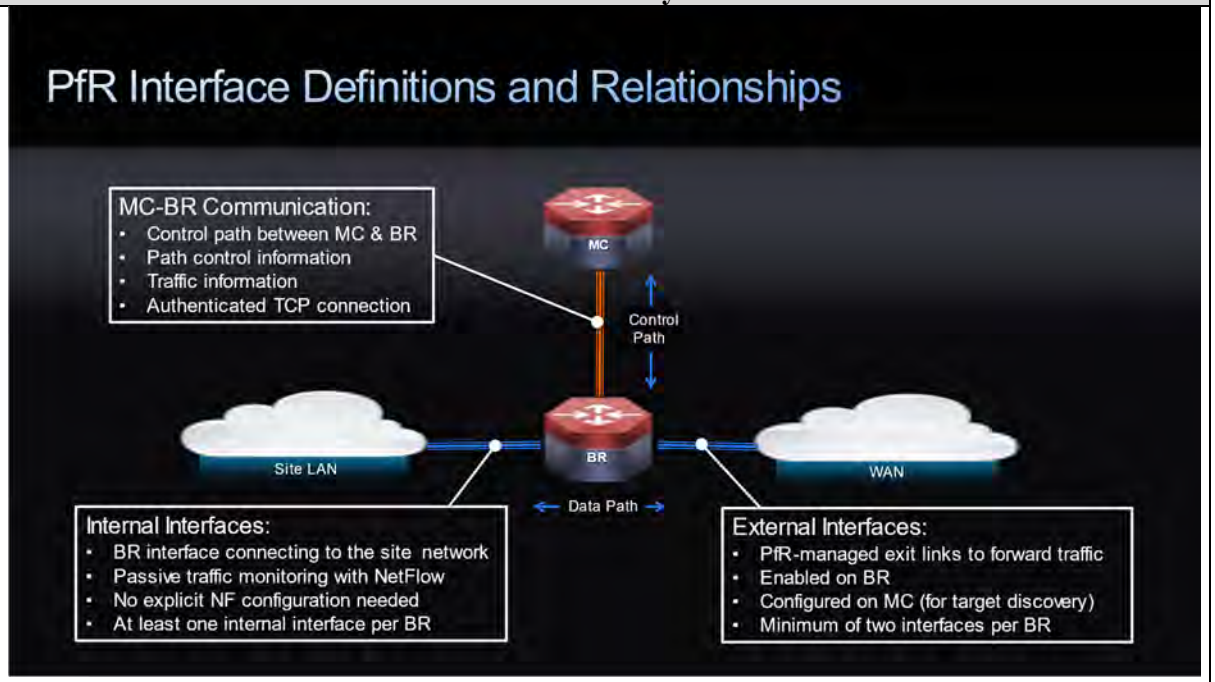
Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 959 of 1100

| No. | '111 Patent Claim 4 | Cisco IWAN System |
|---|---|---|
| | |  |





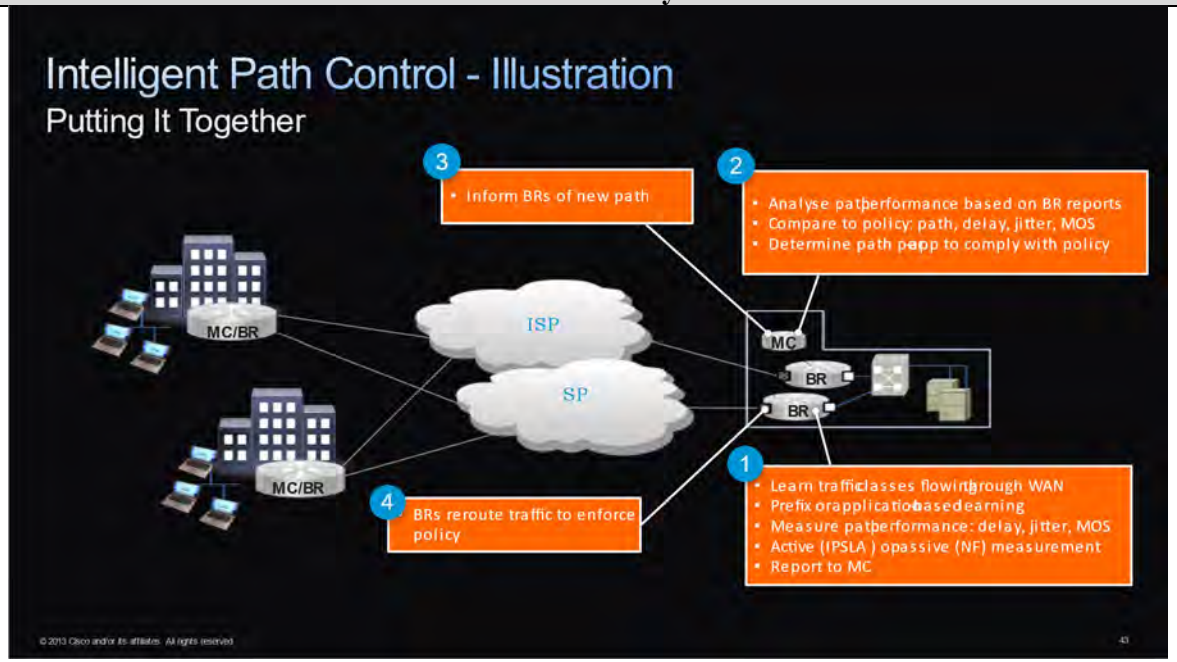
Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 960 of 1100

| No. | '111 Patent Claim 4 | Cisco IWAN System |
|---|---|---|
| | |  |





Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 961 of 1100

| No. | '111 Patent Claim 4 | Cisco IWAN System |
|---|---|---|
| | |  <br><br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Cisco IWAN System in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 4[b] of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references.<br><br>For example, Kempf discloses sending the packet from the network element to the controller or another table, in response to the packet matching the action in the flow table.<br><br>Kempf at [0044] ("FIG. 1 is a diagram of one embodiment of an example network with an OpenFlow switch, conforming to the OpenFlow 1.0 specification. The OpenFlow 1.0 protocol enables a controller 101 to connect to an OpenFlow 1.0 enabled switch 109 using a secure channel 103 and control a single forwarding table 107 in the switch 109. The controller 101 is an external software component executed by a remote computing device that enables a user to configure the Open-Flow 1.0 switch 109. The secure channel 103 can |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 962 of 1100

| No. | '111 Patent Claim 4 | Cisco IWAN System |
|---|---|---|
| | | be provided by any type of network including a local area network (LAN) or a wide area network (WAN), such as the Internet.")<br><br>Kempf at [0045] ("FIG. 2 is a diagram illustrating one embodiment of the contents of a flow table entry. The forwarding table 107 is populated with entries consisting of a rule 201 defining matches for fields in packet headers; an action 203 associated to the flow match; and a collection of statistics 205 on the flow. When an incoming packet is received a lookup for a matching rule is made in the flow table 107. If the incoming packet matches a particular rule, the associated action defined in that flow table entry is performed on the packet.")<br><br>Kempf at [0046] ("A rule 201 contains key fields from several headers in the protocol stack, for example source and destination Ethernet MAC addresses, source and destination IP addresses, IP protocol type number, incoming and outgoing TCP or UDP port numbers. To define a flow, all the available matching fields may be used. But it is also possible to restrict the matching rule to a subset of the available fields by using wildcards for the unwanted fields.")<br><br>Kempf at [0047] ("The actions that are defined by the specification of OpenFlow 1.0 are Drop, which drops the matching packets; Forward, which forwards the packet to one or all outgoing ports, the incoming physical port itself, the controller via the secure channel, or the local networking stack (if it exists). OpenFlow 1.0 protocol data units (PDU s) are defined with a set of structures specified using the C programming language. Some of the more commonly used messages are: report switch configuration message; modify state messages (in-cluding a modify flow entry message and port modification message); read state messages, where while the system is running, the datapath may be queried about its current state using this message; and send packet message, which is used when the controller wishes to send a packet out through the datapath.")<br><br>Kempf at [0050] ("FIG. 4 illustrates one embodiment of the processing of packets through an OpenFlow 1.1 switched packet pro-cessing pipeline. A received packet is compared against each of the flow tables 401. After each flow table match, the actions are accumulated into an action set. If processing requires matching against another flow table, the actions in the matched rule include an action directing processing to the next table in the |

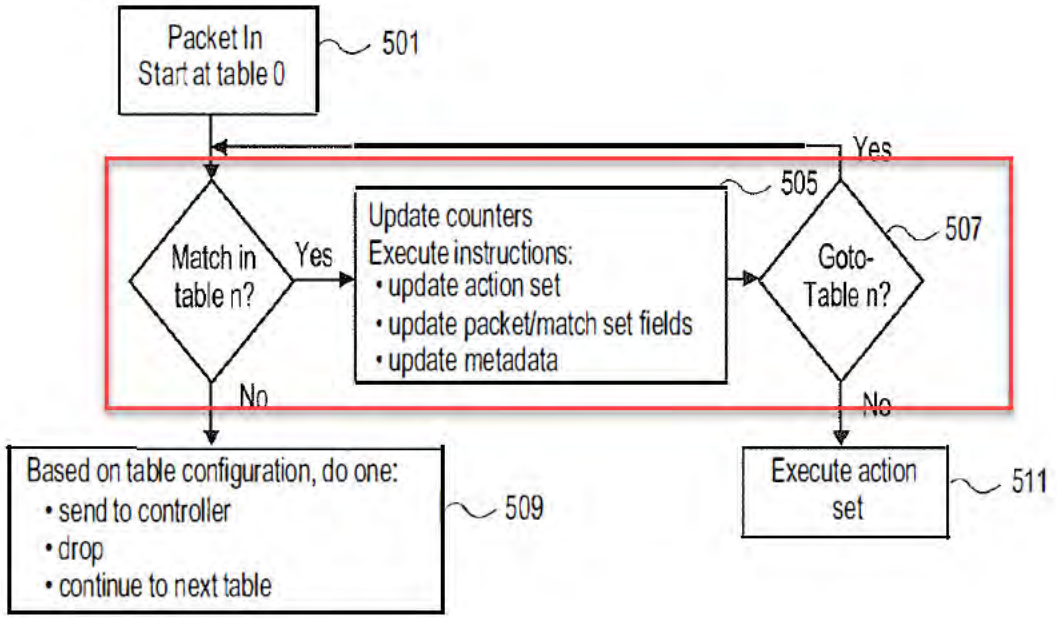


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 963 of 1100

| No. | '111 Patent Claim 4 | Cisco IWAN System |
|---|---|---|
| | | pipeline. Absent the inclusion of an action in the set to execute all accumulated actions immediately, the actions are executed at the end 403 of the packet processing pipeline. An action allows the writing of data to a metadata register, which is carried along in the packet processing pipe-line like the packet header.")<br><br>Kempf at [0091] ("When a packet header matches a rule associated with the virtual port, the GTP TEID is written into the lower 32 bits of the metadata and the packet is directed to the virtual port. The virtual port calculates the hash of the TEID and looks up the tunnel header information in the tunnel header table. If no such tunnel information is present, the packet is forwarded to the controller with an error indication. Other-wise, the virtual port constructs a GTP tunnel header and encapsulates the packet. Any DSCP bits or VLAN priority bits are additionally set in the IP or MAC tunnel headers, and any VLAN tags or MPLS labels are pushed onto the packet. The encapsulated packet is forwarded out the physical port to which the virtual port is bound.")<br><br>Kempf at [0106] ("This encapsulates the packet and sends it to the OpenFlow controller.")<br><br>Kempf at Figure 5 (annotation added) |

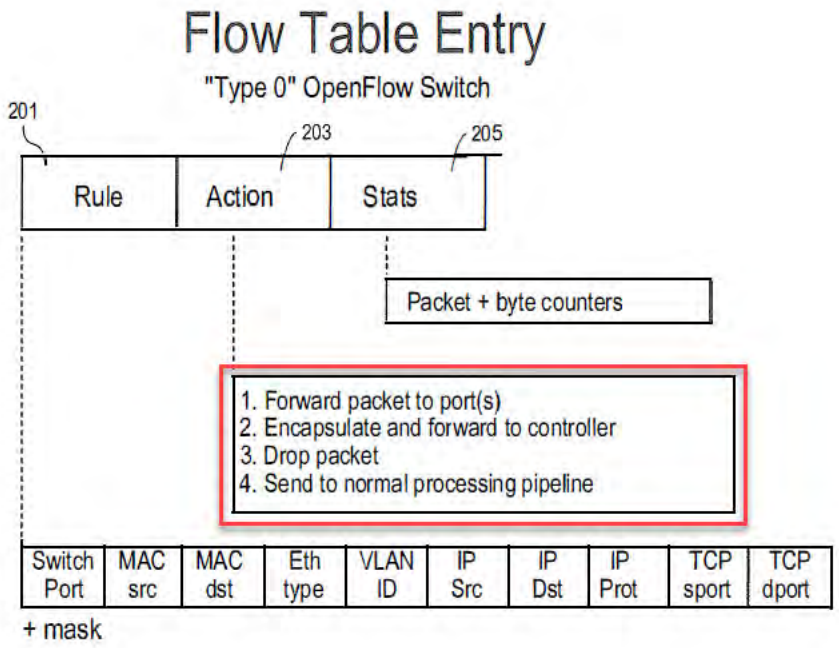


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 964 of 1100

| No. | '111 Patent Claim 4 | Cisco IWAN System |
|---|---|---|
| | |  <br>FIG. 5<br>Kempf at Figure 2 (annotation added) |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 965 of 1100

| No. | '111 Patent Claim 4 | Cisco IWAN System |
|---|---|---|
| | |  <br>FIG. 2<br><br>For example, Swenson discloses determining by the steering device monitors flows that match one or more signatures or criteria of the packet. Swenson further discloses that when a matching flow is detected the steering device forwards the packet to the network controller.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 966 of 1100

| No. | ’111 Patent Claim 4 | Cisco IWAN System |
|---|---|---|
| | | inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.”)<br><br>Swenson at [0028] (“In contrast to conventional inline TCP throughput monitoring devices that monitor every single data packets transmitted and received, the network controller 140 is an "out-of-band" computer server that interfaces with the steer-ing device 130 to selectively inspect user flows of interest. The network controller 140 may further identify user flows (e.g., among the flows of interest) for optimization. In one embodiment, the network controller 140 may be imple-mented at the steering device 130 to monitor traffic. In other embodiments, the network controller 140 is coupled to and communicates with the steering device 130 for traffic moni-toring and optimization. When queried by the steering device 130, the network controller 140 determines if a given network flow should be ignored, monitored further or optimized. Opti-mization of a flow is often decided at the beginning of the flow because it is rarely possible to switch to optimized content mid-stream once non-optimized content delivery has begun. However, the network controller 140 may determine that existing flows associated with a particular subscriber or other entity should be optimized. In turn, new flows ( e.g., resulting from seek requests in media, new media requests, resume after pause, etc.) determined to be associated with the entity may be optimized. The network controller 140 uses the net-work state as well as historical traffic data in its decision for monitoring and optimization. Knowledge on the current net-work state, such as congestion, deems critical when it comes to data optimization.”)<br><br>Swenson at [0029] (“As a flow is sent to the network controller 140 for inspection, historical network traffic data stored at the net-work controller 140 may be searched. The historical network traffic data includes information such as subscriber informa-tion, the cell towers to which the user devices attached, rout-ers through which the traffic is passing, geography regions, the backhaul segments, and time-of-day of the flows. For example, in a mobile network, the cell tower to which a user device is attached can be most useful, since it is the location where most congestion occurs due to limited bandwidth and high cost of the radio access network infrastructure. The network controller 140 looks into the historical traffic data for the average of the bandwidth per user at the particular cell tower. The |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 967 of 1100

| No. | '111 Patent Claim 4 | Cisco IWAN System |
|---|---|---|
| | | network controller 140 can then estimate the amount ofbandwidth or degree of congestion for the new flow based on the historical record.")<br><br>Swenson at [0038] ("Turning back to FIG. 1, the network controller 140 allows network operators to apply fine granular optimization policies to ensure high quality of experience (QoE) based on cell tower congestion, device types, subscriber profiles and service plans with lower hardware and software costs. The architecture of the network controller 140 provides an excel-lent fit for the net neutrality guideline of"reasonable network management", and better compliance to the copyright law (DMCA) than solutions that rely on long-term caching. Hav-ing the ability of monitoring network traffic on a per sub-scriber, per flow, or per video file basis, the network controller 140 also selectively monitors and optimizes only a subset of traffic that benefits from optimization the most, thus achiev-ing both scalability and efficiency for optimization at a com-petitive price-point. The core element of the network control-ler 140 lies in its mechanisms for congestion detection and mitigation, which allows optimization resources to be utilized in the most efficient and surgical manner.")<br><br>Swenson at [0039] ("Referring now to FIG. 3, it illustrates one embodi-ment of an example architecture of the network controller 140 for providing selective real-time network monitoring and subscriber identification. The network controller 140 com-prises a flow analyzer 312, a policy engine 314, a steering device interface 316, a video optimizer redirector 318, a flow cache 322, and a subscriber log 324. In other embodiments, the network controller 140 may include additional, fewer, or different components for various applications. Conventional components such as network interfaces, security functions, failover servers, management and network operations con-soles, and the like are not shown so as to not obscure the details of the system architecture.")<br><br>Swenson at [0045] ("The steering device interface 316 interacts with an external routing appliance, such as the steering device 130 to divert portions of the network traffic ( e.g., large object net-work flows). Existing routing appliances in most carrier net-works are designed to handle large amounts of network traf-fic. They are not, however, ideal devices to operate for monitoring and analysis individual flows. Through the steer-ing device interface 316, the network controller 140 may communicate with the external routing appliances, such as the steering device 130, to steer a portion of network traffic to the |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 968 of 1100

| No. | '111 Patent Claim 4 | Cisco IWAN System |
|---|---|---|
| | | network controller 140 when certain conditions are met. Generally, network flows of interest to the network controller 140 contain larger media objects, such as videos and images. In one embodiment, the smaller flows, such as web page and text information, are not exchanged over the steering device interface 316.")<br><br>Swenson at [0059] ("In one embodiment, as the steering device 130 monitors network responses, it is looking for flows that match one or more signatures for video and images. When a match-ing flow is detected, the steering device 130 forwards the HTTP request and a portion of the HTTP response to the network controller 140 over the ICAP client interface 404. After receiving the request and the portion of response at the ICAP server interface 406, the flow analyzer 312 of the net-work controller 140 performs a deep flow inspection to deter-mine if the flow is worth bandwidth monitoring and/or user detection. For example, the flow inspection performed by the flow analyzer 312 may determine if the flow indeed contains large or medium object ( e.g., larger than 50 kB), and/or if the source IP address of the flow is from a user or a group of users that are required to be monitored by policies. The flow ana-lyzer 312 may also determine if the flow needs to be opti-mized based on historical flow statistical data.")<br><br>Swenson at [0060] ("If the flow is deemed of interest, the steering device 130 is notified to steer the flow through the network controller 140. This is known as the "continue" working mode for bandwidth monitoring. In the "continue" mode, the network controller 140 interfaces with the steering device 130 to func-tion, on-demand, as a traditional inline network element for flows deemed of interest. Thus, the network controller 140 ingests the network flow for inspection and subsequently forwards the network flow on the network response path. For example, for this particular flow, the origin server 160 responds to the user request by sending video or images over the network link 413 to the steering device 130, which for-wards the video or images to the network controller 140 over a network link 414. After the network controller 140 updates the flow statistics, the video or images are returned to the steering device 130 over a network link 415, which transmits the video or images to the user device 110 over the network link 416.")<br><br>Swenson at [0065] ("FIG. 5 is a block diagram illustrating an example event trace of "continue" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user |

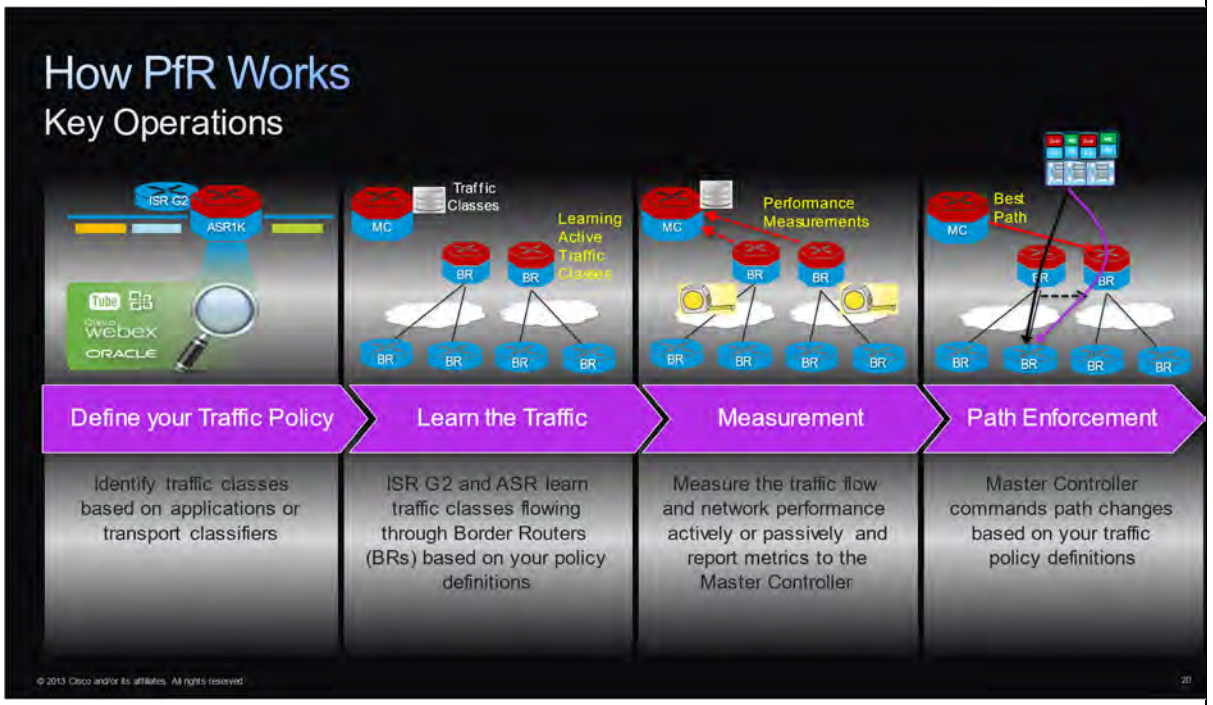


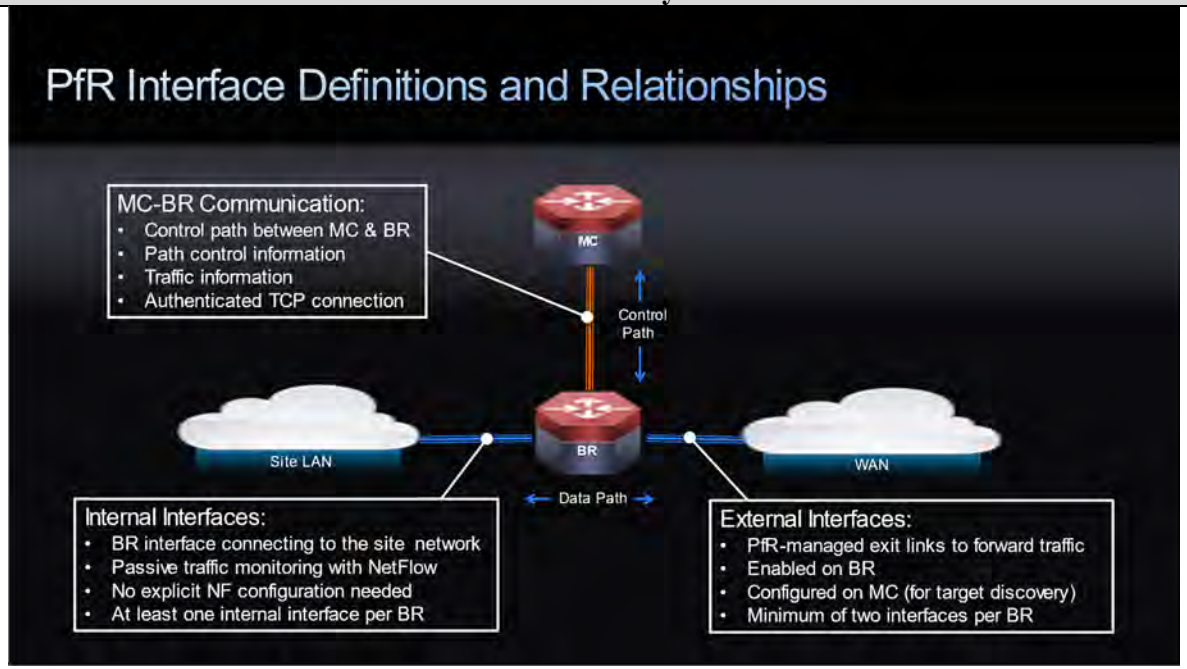
Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 969 of 1100

| No. | '111 Patent Claim 4 | Cisco IWAN System |
|---|---|---|
| | | device 110 initiates an HTTP GET request 512 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 512 to the intended origin server 160 and receives a response 514 back from the origin server 160. The steering device 130 then sends an ICAP request message 516 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 518. Upon receiving the instruc-tion, the steering device 130 re-writes the response 514 to an HTTP redirect response 520, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 520 directly to the user device 110. In case the flow dose not contain video or image objects, or the network controller 140 determines not to monitor the flow, the steering device 13 0 would forward the response to the user device 110.")<br><br>Swenson at [0069] ("FIG. 6 is a block diagram illustrating an example event trace of "counting" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 612 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 612 to the intended origin server 160 and receives a response 614 back from the origin server 160. The steering device 130 then sends an ICAP request message 616 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 618. Upon receiving the instruc-tion, the steering device 130 re-writes the response 614 to an HTTP redirect response 620, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 620 directly to the user device 110. In case the flow dose not contain video or image objects that need to be redirected, the steering device 130 would forward the response to the user device 110.") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 970 of 1100

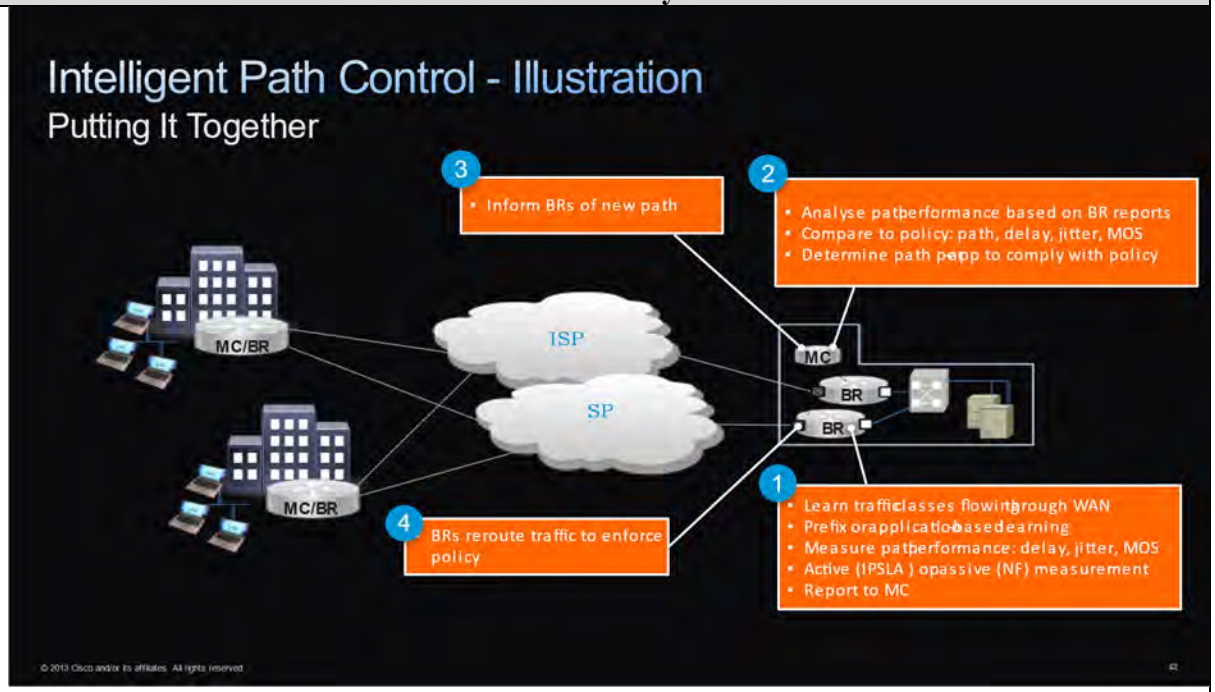
| No. | '111 Patent Claim 4 | Cisco IWAN System |
|---|---|---|
| 4[c] | responsive to receiving the packet, analyzing the packet, by the controller; | Cisco IWAN System discloses responsive to receiving the packet, analyzing the packet, by the controller.<br><br>For example, Cisco IWAN System discloses analyzing traffic flow metrics received by the hub/master controller to update and change traffic policy definitions. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Cisco IWAN System is found to not meet this limitation, responsive to receiving the packet, analyzing the packet, by the controller would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>Cisco IWAN<br>  <br><br><br>Cisco IWAN - Uncompromised Experience |

| No. | '111 Patent Claim 4 | Cisco IWAN System |
|---|---|---|
| | |  |





Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 972 of 1100

| No. | '111 Patent Claim 4 | Cisco IWAN System |
|---|---|---|
| | |  <br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Cisco IWAN System in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 4(c) of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references.<br><br>For example, Swenson discloses the network controller comprising a flow analyzer for analyzing and inspecting the packet.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 973 of 1100

| No. | '111 Patent Claim 4 | Cisco IWAN System |
|---|---|---|
| | | inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0028] ("In contrast to conventional inline TCP throughput monitoring devices that monitor every single data packets transmitted and received, the network controller 140 is an "out-of-band" computer server that interfaces with the steer-ing device 130 to selectively inspect user flows of interest. The network controller 140 may further identify user flows (e.g., among the flows of interest) for optimization. In one embodiment, the network controller 140 may be imple-mented at the steering device 130 to monitor traffic. In other embodiments, the network controller 140 is coupled to and communicates with the steering device 130 for traffic moni-toring and optimization. When queried by the steering device 130, the network controller 140 determines if a given network flow should be ignored, monitored further or optimized. Opti-mization of a flow is often decided at the beginning of the flow because it is rarely possible to switch to optimized content mid-stream once non-optimized content delivery has begun. However, the network controller 140 may determine that existing flows associated with a particular subscriber or other entity should be optimized. In turn, new flows ( e.g., resulting from seek requests in media, new media requests, resume after pause, etc.) determined to be associated with the entity may be optimized. The network controller 140 uses the net-work state as well as historical traffic data in its decision for monitoring and optimization. Knowledge on the current net-work state, such as congestion, deems critical when it comes to data optimization.")<br><br>Swenson at [0029] ("As a flow is sent to the network controller 140 for inspection, historical network traffic data stored at the net-work controller 140 may be searched. The historical network traffic data includes information such as subscriber informa-tion, the cell towers to which the user devices attached, rout-ers through which the traffic is passing, geography regions, the backhaul segments, and time-of-day of the flows. For example, in a mobile network, the cell tower to which a user device is attached can be most useful, since it is the location where most congestion occurs due to limited bandwidth and high cost of the radio access network infrastructure. The network controller 140 looks into the historical traffic data for the average of the bandwidth per user at the particular cell tower. The |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 974 of 1100

| No. | '111 Patent Claim 4 | Cisco IWAN System |
|---|---|---|
| | | network controller 140 can then estimate the amount of bandwidth or degree of congestion for the new flow based on the historical record.")<br><br>Swenson at [0038] ("Turning back to FIG. 1, the network controller 140 allows network operators to apply fine granular optimization policies to ensure high quality of experience (QoE) based on cell tower congestion, device types, subscriber profiles and service plans with lower hardware and software costs. The architecture of the network controller 140 provides an excel-lent fit for the net neutrality guideline of "reasonable network management", and better compliance to the copyright law (DMCA) than solutions that rely on long-term caching. Hav-ing the ability of monitoring network traffic on a per sub-scriber, per flow, or per video file basis, the network controller 140 also selectively monitors and optimizes only a subset of traffic that benefits from optimization the most, thus achiev-ing both scalability and efficiency for optimization at a com-petitive price-point. The core element of the network control-ler 140 lies in its mechanisms for congestion detection and mitigation, which allows optimization resources to be utilized in the most efficient and surgical manner.")<br><br>Swenson at [0039] ("Referring now to FIG. 3, it illustrates one embodi-ment of an example architecture of the network controller 140 for providing selective real-time network monitoring and subscriber identification. The network controller 140 com-prises a flow analyzer 312, a policy engine 314, a steering device interface 316, a video optimizer redirector 318, a flow cache 322, and a subscriber log 324. In other embodiments, the network controller 140 may include additional, fewer, or different components for various applications. Conventional components such as network interfaces, security functions, failover servers, management and network operations con-soles, and the like are not shown so as to not obscure the details of the system architecture.")<br><br>Swenson at [0059] ("In one embodiment, as the steering device 130 monitors network responses, it is looking for flows that match one or more signatures for video and images. When a match-ing flow is detected, the steering device 130 forwards the HTTP request and a portion of the HTTP response to the network controller 140 over the ICAP client interface 404. After receiving the request and the portion of response at the ICAP server interface 406, the flow analyzer 312 of the net-work controller 140 performs a deep flow inspection to deter-mine if the flow is worth bandwidth monitoring and/or user detection. For example, |


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 975 of 1100

| No. | '111 Patent Claim 4 | Cisco IWAN System |
|---|---|---|
| | | the flow inspection performed by the flow analyzer 312 may determine if the flow indeed contains large or medium object ( e.g., larger than 50 kB), and/or if the source IP address of the flow is from a user or a group of users that are required to be monitored by policies. The flow ana-lyzer 312 may also determine if the flow needs to be opti-mized based on historical flow statistical data.")<br><br>Swenson at [0060] ("If the flow is deemed of interest, the steering device 130 is notified to steer the flow through the network controller 140. This is known as the "continue" working mode for bandwidth monitoring. In the "continue" mode, the network controller 140 interfaces with the steering device 130 to func-tion, on-demand, as a traditional inline network element for flows deemed of interest. Thus, the network controller 140 ingests the network flow for inspection and subsequently forwards the network flow on the network response path. For example, for this particular flow, the origin server 160 responds to the user request by sending video or images over the network link 413 to the steering device 130, which for-wards the video or images to the network controller 140 over a network link 414. After the network controller 140 updates the flow statistics, the video or images are returned to the steering device 130 over a network link 415, which transmits the video or images to the user device 110 over the network link 416.")<br><br>Swenson at [0061] ("Once a flow is reported to the network controller 140, a flow cache entry is created for the flow in the flow cache 322. The flow cache entry keeps track of the flow and its associated bandwidth. For a flow that is marked in "continue" mode, each time the steering device 130 forwards a next portion of the flow payload to the network controller 140, the flow cache 3 22 updates the number of bytes for transmitted in the flow. By monitoring the number of bytes per flow over time, the flow analyzer 312 is capable of determining an estimate value of bandwidth associated with flow. Further-more, since the steering device 130 does not have infinite packet buffers, if congestion happens on the network link 416 from the steering device 130 to the user device 110, the TCP congestion control mechanism kicks in at the steering device 130, which may slows down and/or eventually stop receiving data over the network link 413 from origin server 160. During the congestion, the steering device 130 would not forward any data to the network controller 140, since the link 416 is congested and the network controller 140 would not be able to transmit data to the user device 110. Therefore, as an inline element, the network controller 140 can detect network con-gestions and estimate bandwidth associated with any flows of |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 976 of 1100

| No. | '111 Patent Claim 4 | Cisco IWAN System |
|---|---|---|
| | | interest selected by the network controller 140. However, in the "continue" mode, the network controller 140 does not modify and transform the HTTP messaged it receives over the ICAP interface. The network controller 140 simply updates the flow statistics and returns the video or images to the steering device 130 for transmission to the user device 110.")<br><br>Swenson at [0065] ("FIG. 5 is a block diagram illustrating an example event trace of "continue" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 512 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 512 to the intended origin server 160 and receives a response 514 back from the origin server 160. The steering device 130 then sends an ICAP request message 516 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 518. Upon receiving the instruc-tion, the steering device 130 re-writes the response 514 to an HTTP redirect response 520, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 520 directly to the user device 110. In case the flow dose not contain video or image objects, or the network controller 140 determines not to monitor the flow, the steering device 13 0 would forward the response to the user device 110.")<br><br>Swenson at [0069] ("FIG. 6 is a block diagram illustrating an example event trace of "counting" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 612 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 612 to the intended origin server 160 and receives a response 614 back from the origin server 160. The steering device 130 then sends an ICAP request message 616 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this |

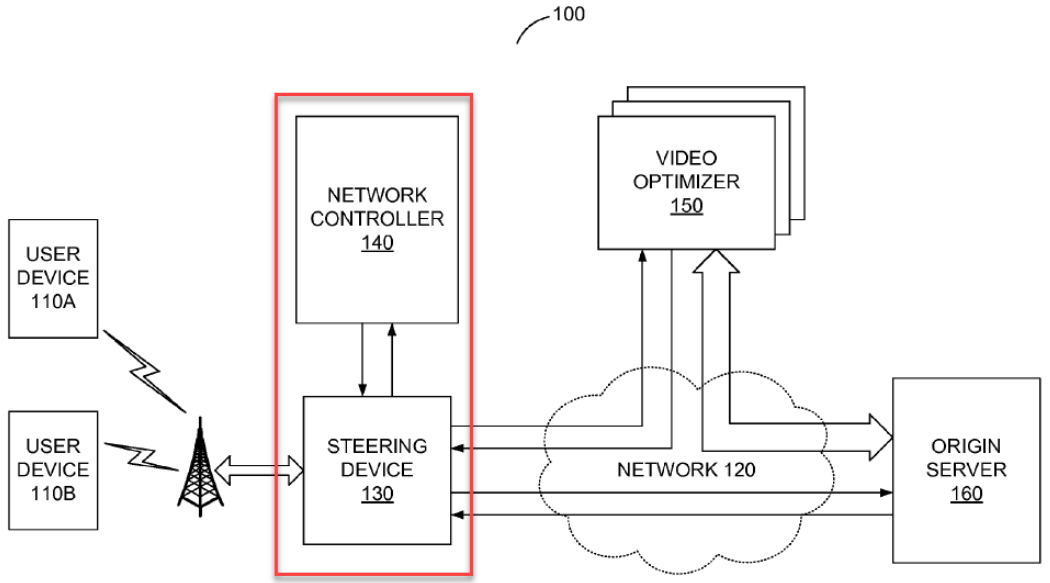


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 977 of 1100

| No. | '111 Patent Claim 4 | Cisco IWAN System |
|---|---|---|
| | | case, the network controller 140 responds with a redirect to optimize the video in ICAP response 618. Upon receiving the instruc-tion, the steering device 130 re-writes the response 614 to an HTTP redirect response 620, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 620 directly to the user device 110. In case the flow dose not contain video or image objects that need to be redirected, the steering device 130 would forward the response to the user device 110.")<br><br>Swenson at [0071] ("After receiving the request, the video optimizer 150 forwards the video HTTP GET requests 622 to the origin server 160 and in return, receives a video file 624 from the origin server 160. The video optimizer 150 transcodes the video file to a format usable by the client device 110 based on network bandwidth available to the user device 110. The optimized video 626 is then transmitted from the video opti-mizer 150 to the steering device 130. In one embodiment, the steering device 130 intercepts the optimized video 626. The steering device 130 will then send an ICAP request to the network controller 140 for inspection. The network controller 140 deems this flow to be monitored and sends ICAP response 630. The steering device 130 then allows the flow to go through to the user device 110. The steering device 130 next sends periodic ICAP "counting" updates 632 to the network controller 140 until the flow completes. As such, the client receives the optimized video 626 for substantially real-time playback on an application executing on the user device 110.")<br><br>Swenson at Figure 1 (annotation added) |

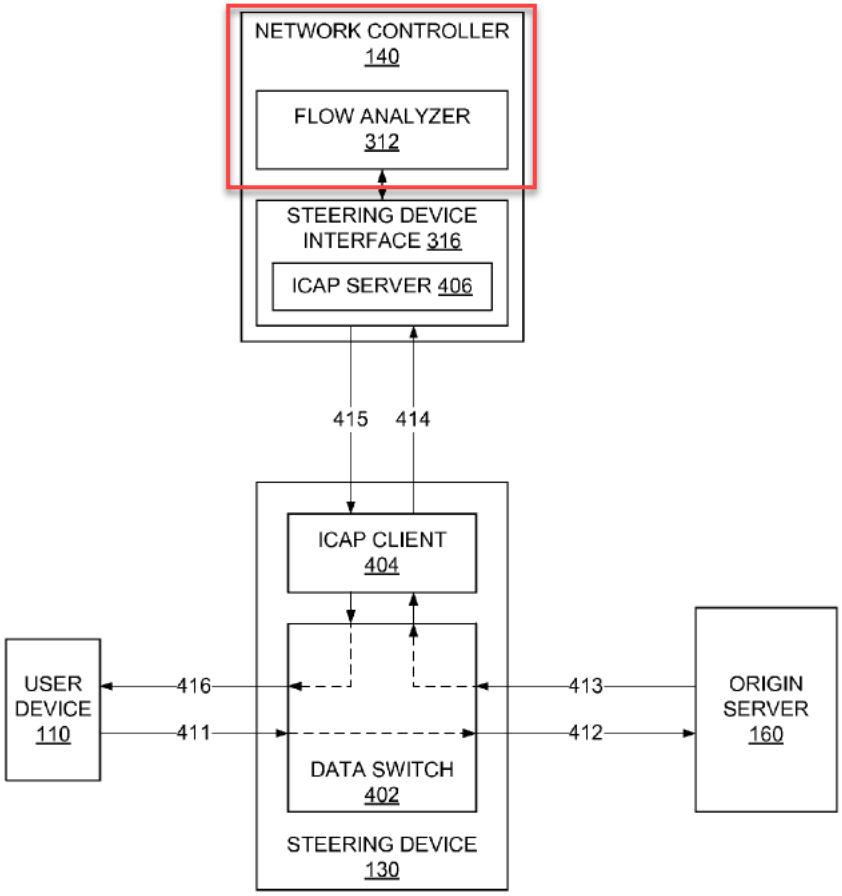


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 978 of 1100

| No. | '111 Patent Claim 4 | Cisco IWAN System |
|---|---|---|
| | |  <br>FIG. 1<br>Swenson at Figure 4A (annotation added) |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 979 of 1100

| No. | '111 Patent Claim 4 | Cisco IWAN System |
|---|---|---|
| | |  <br>FIG. 4A<br><br>For example, Copeland discloses analyzing packets received by the intrusion detection engine on the monitoring appliance. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 980 of 1100

| No. | '111 Patent Claim 4 | Cisco IWAN System |
|---|---|---|
| | | Copeland at [0021] ("The present invention provides an accurate and reliable method for detecting network attacks through the use of sampled packet headers that are provided by a source such as that as defined in sFlow and further based in large part on "flows" as opposed to signatures or anomalies. By utilizing the host and flow information structures that are inherent with flow-based analysis and applying rules of statistical significance and analysis, the intrusion detection system can operate with sampled data such as provided by sFlow in order to provide a hybrid solution that combines many of the benefits of a packet capture implementation with the distributed nature of an IDS that operates on Netflow, thus providing an enhanced wide-area IDS solu-tion.")<br><br>Copeland at [0023] ("According to one aspect of the invention, the detection system works by assigning sampled data packets to various client/server ( C/S) flows. Statistics are collected for each determined flow. Then, the flow statistics are analyzed to determine if the flow appears to be legitimate traffic or possible suspicious activity. A value, referred to as a "concern index," is assigned to each flow that appears suspicious. By assigning a value to each flow that appears suspicious and adding that value to an accumulated concern index associated with the responsible host, it is possible to identify hosts that are engaged in intruder activity without generation of significant unwarranted false alarms. When the concern index value of a host exceeds a preset alarm value, an alert is issued and appropriate action can be taken.")<br><br>Copeland at [0024] ("Generally speaking, the intrusion detection system analyzes network communication traffic for potential detrimental activity. The system collects flow data from sampled packet headers between two hosts or Internet Protocol (IP) addresses. Collecting flow data from packet headers asso-ciated with a single service where at least one port remains constant allows for more efficient analysis of the flow data. The collected flow data is analyzed to assign a concern index value to the flow based upon a probability that the flow was not normal for data communications. A host list is main-tained containing an accumulated concern index derived from the flows associated with the host. Once the accumu-lated concern index has exceeded an alarm threshold value, an alarm signal is generated.")<br><br>Copeland at [0027] ("According to one aspect of the invention, the detection system works by assigning sampled data packets to various client/server ( C/S) flows. Statistics are |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 981 of 1100

| No. | '111 Patent Claim 4 | Cisco IWAN System |
|---|---|---|
| | | collected for each determined flow. Then, the flow statistics are analyzed to determine if the flow appears to be legitimate traffic or possible suspicious activity. A value, referred to as a "concern index," is assigned to each flow that appears suspicious. By assigning a value to each flow that appears suspicious and adding that value to an accumulated concern index associated with the responsible host, it is possible to identify hosts that are engaged in intruder activity without generation of significant unwarranted false alarms. When the concern index value of a host exceeds a preset alarm value, an alert is issued and appropriate action can be taken.")<br><br>Copeland at [0028] ("Generally speaking, the intrusion detection system analyzes network communication traffic for potential detri-mental activity. The system collects flow data from sampled packet headers between two hosts or Internet Protocol (IP) addresses. Collecting flow data from packet headers asso-ciated with a single service where at least one port remains constant allows for more efficient analysis of the flow data. The collected flow data is analyzed to assign a concern index value to the flow based upon a probability that the flow was not normal for data communications. A host list is main-tained containing an accumulated concern index derived from the flows associated with the host. Once the accumu-lated concern index has exceeded an alarm threshold value, an alarm signal is generated.")<br><br>Copeland at [0063] ("Consequently, abnormal flows and/or events iden-tified by the intrusion detection engine 155 will raise the concern index (CI) for the associated host. The intrusion detection engine 155 analyzes the data flow between IP devices. However, different types of services have different flow characteristics associated with that service. Therefore, a C/S flow can be determined by the packets exchanged between the two hosts dealing with the same service.")<br><br>Copeland at [0065] ("The intrusion detection engine 155 analyzes the flow data 160 to determine if the flow appears to be legitimate traffic or possible suspicious activity. Flows with suspicious activity are assigned a predetermined concern index (CI) value based upon a heuristically predetermined assessment of the significance of the threat of the particular traffic or flow or suspicious activity. The flow concern index values have been derived heuristically from extensive net-work traffic analysis. Concern index values are associated with particular hosts and stored in the host data structure 166 (FIG. 1). Exemplary concern |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 982 of 1100

| No. | '111 Patent Claim 4 | Cisco IWAN System |
|---|---|---|
| | | index values for various exemplary flow-based events and other types of events are illustrated in connection with FIGS. 6 and 7.)<br><br>Copeland at [0069] ("It will now be appreciated that the disclosed meth-odology of intrusion detection is accomplished at least in part by analyzing communication flows to determine if such communications have the flow characteristics of probes or attacks. By analyzing communications for abnormal flow characteristics, attacks can be determined without the need for resource-intensive packet data analysis. A flow can be determined from the packets 101 that are transmitted between two hosts utilizing a single service. The addresses and port numbers of communications are easily discerned by analysis of the header information in a datagram.")<br><br>Copeland at [0087] ("As previously stated, the flow-based engine 155 does not analyze the data segments of packets for signature identification. Instead, the engine 155 associates all packets with a flow. It analyzes certain statistical data and assigns a concern index value to abnormal activity. The engine 155 builds a concern index for suspicious hosts by detecting suspicious activities on the network. An alarm is generated when those hosts build enough concern (in the form of a cumulated CI value) to cross the network administrator's predetermined threshold.")<br><br>Copeland at [0097] ("The described TCP session 300 of FIG. 3 is a generic TCP session in which a network might engage. In accordance with the invention, flow data is collected about the session to help determine if the communication is abnormal. In the preferred embodiment, information such as the total number of packets sent, the total amount of data sent, the session start time and duration, and the TCP flags set in all of the packets, are collected, stored in the database 160, and analyzed to determine if the communication was suspicious. If a communication is deemed suspicious, i.e. it meets predetermined criteria, a predetermined concern index value associated with a determined category of suspicious activity is added to the cumulated CI value associated with the host that made the communication.")<br><br>Copeland at [0111] ("As shown, the packets exchanged between two hosts associated with a single service can determine a flow. A port number designates a service application that is associated with the particular port. Communications utiliz-ing differing protocols or services |

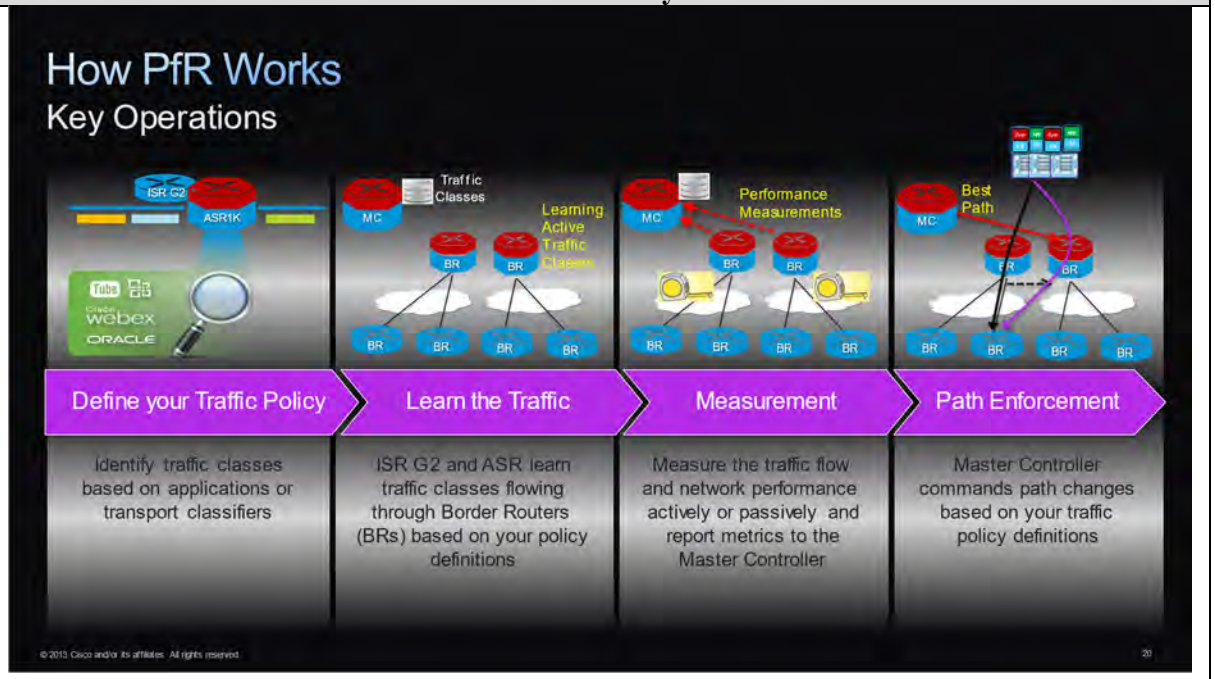


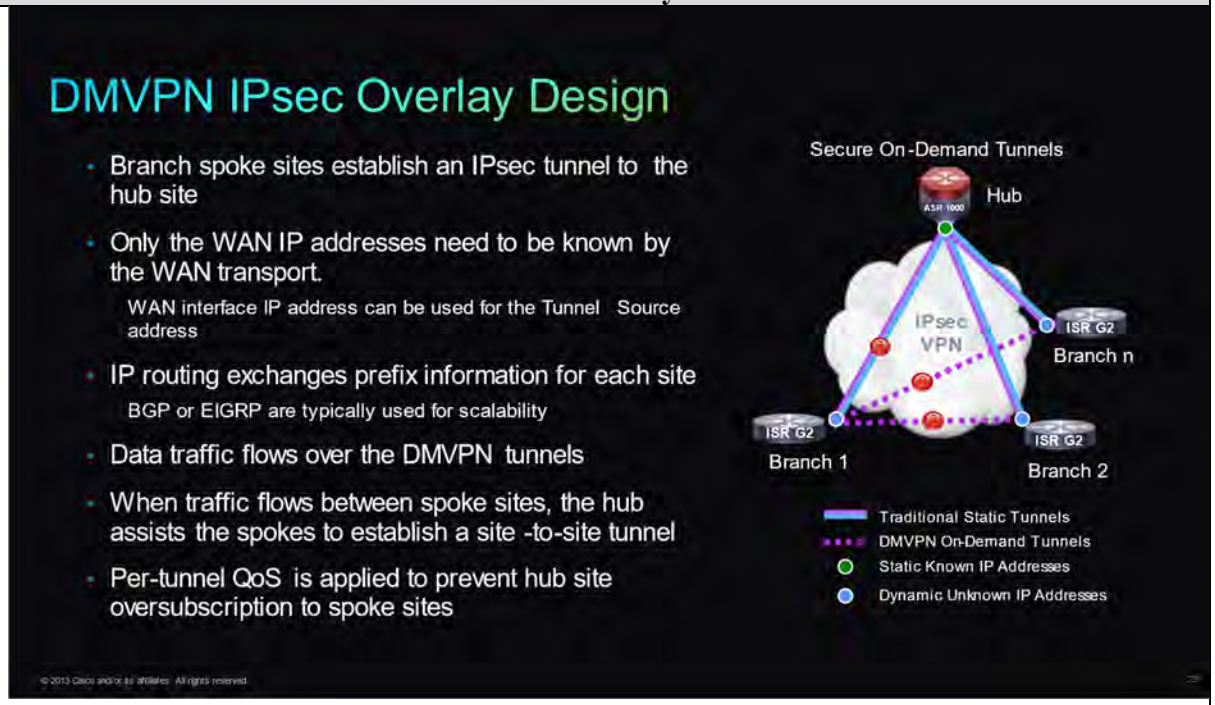
Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 983 of 1100

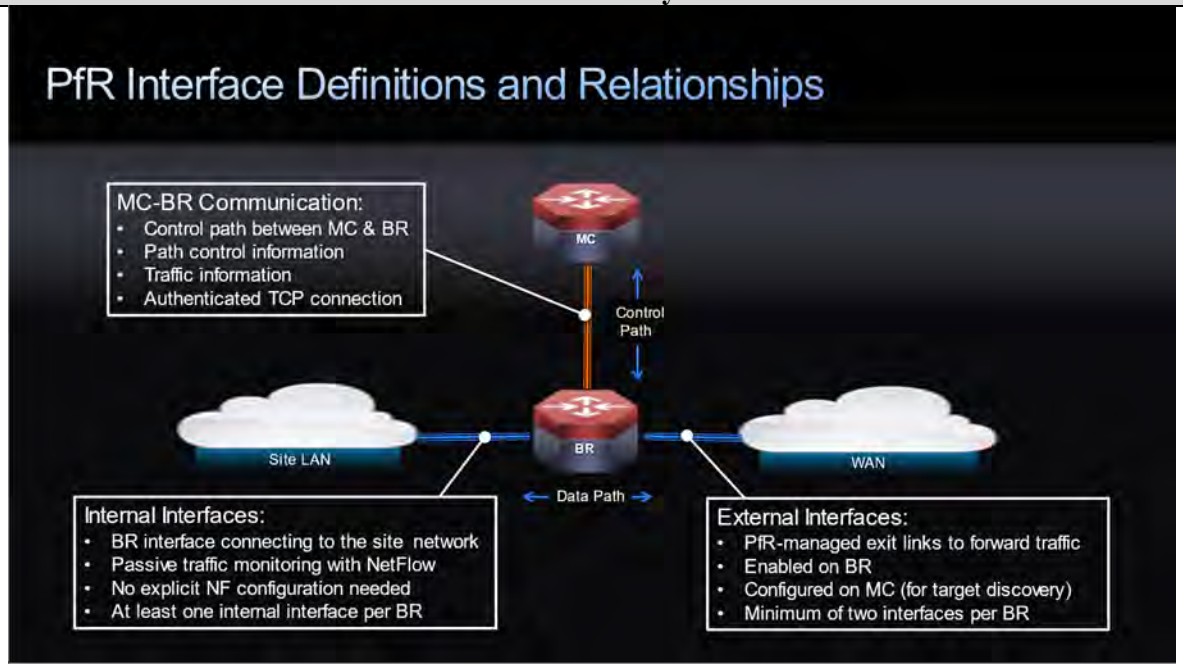
| No. | '111 Patent Claim 4 | Cisco IWAN System |
|---|---|---|
| | | provide differing flow characteristics. Consequently, the flow engine 155 analyzes each of the services separately.")<br><br>Copeland at [0150] ("A preferred hardware configuration 800 of an embodiment that executes the functions of the above-described flow-based engine is described in reference to FIG. 8. FIG. 8 illustrates a typically hardware configuration 800 for a network intrusion detection system. A monitoring appliance 150 serves as a pass-by filter of network traffic. A network device 135, such as a router or switch supporting sFlow provides the location for connecting the monitoring appliance 150 to the network 899 for monitoring the network traffic.") |
| 4[d] | sending the packet, by the controller, to the network node; and | Cisco IWAN System discloses sending the packet, by the controller, to the network node.<br><br>For example, Cisco IWAN System discloses sending updated traffic policies with traffic flow metrics to the branch devices/border routers. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Cisco IWAN System is found to not meet this limitation, sending the packet, by the controller, to the network node would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>Cisco IWAN |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 984 of 1100

| No. | '111 Patent Claim 4 | Cisco IWAN System |
|---|---|---|
| | |  <br>Cisco IWAN - Uncompromised Experience |

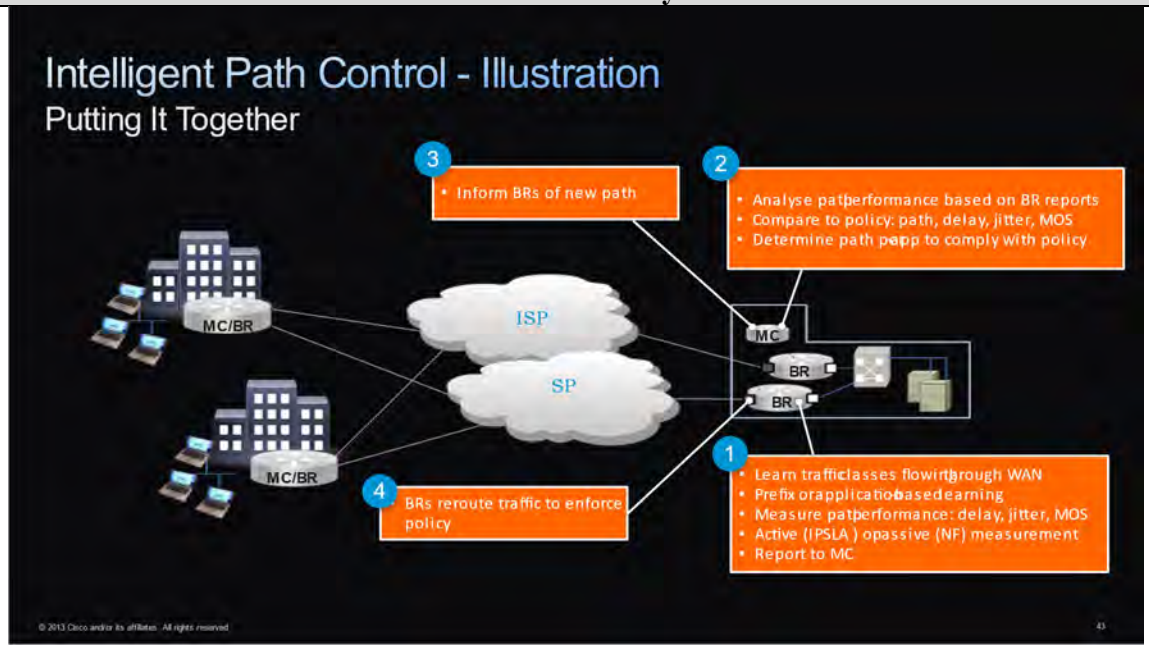
| No. | '111 Patent Claim 4 | Cisco IWAN System |
|---|---|---|
| | |  <br>Cisco IWAN - Uncompromised Experience |

| No. | '111 Patent Claim 4 | Cisco IWAN System |
|---|---|---|
| | |  |





Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 987 of 1100

| No. | '111 Patent Claim 4 | Cisco IWAN System |
|---|---|---|
| | |  <br><br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Cisco IWAN System in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 4(d) of Exhibit E-4 renders the claim, including the present limitation, obvious. Below is an example.<br><br>For example, Swenson discloses sending the packet, for example a video or image, back to the steering device after the network controller analyzes the packet and updates flow statistics.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 988 of 1100

| No. | '111 Patent Claim 4 | Cisco IWAN System |
|---|---|---|
| | | interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0028] ("In contrast to conventional inline TCP throughput monitoring devices that monitor every single data packets transmitted and received, the network controller 140 is an "out-of-band" computer server that interfaces with the steer-ing device 130 to selectively inspect user flows of interest. The network controller 140 may further identify user flows (e.g., among the flows of interest) for optimization. In one embodiment, the network controller 140 may be imple-mented at the steering device 130 to monitor traffic. In other embodiments, the network controller 140 is coupled to and communicates with the steering device 130 for traffic moni-toring and optimization. When queried by the steering device 130, the network controller 140 determines if a given network flow should be ignored, monitored further or optimized. Opti-mization of a flow is often decided at the beginning of the flow because it is rarely possible to switch to optimized content mid-stream once non-optimized content delivery has begun. However, the network controller 140 may determine that existing flows associated with a particular subscriber or other entity should be optimized. In turn, new flows ( e.g., resulting from seek requests in media, new media requests, resume after pause, etc.) determined to be associated with the entity may be optimized. The network controller 140 uses the net-work state as well as historical traffic data in its decision for monitoring and optimization. Knowledge on the current net-work state, such as congestion, deems critical when it comes to data optimization.")<br><br>Swenson at [0029] ("As a flow is sent to the network controller 140 for inspection, historical network traffic data stored at the net-work controller 140 may be searched. The historical network traffic data includes information such as subscriber informa-tion, the cell towers to which the user devices attached, rout-ers through which the traffic is passing, geography regions, the backhaul segments, and time-of-day of the flows. For example, in a mobile network, the cell tower to which a user device is attached can be most useful, since it is the location where most congestion occurs due to limited bandwidth and high cost of the radio access network infrastructure. The network controller 140 looks into the historical traffic data for the average of the bandwidth per user at the particular cell tower. The |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 989 of 1100

| No. | '111 Patent Claim 4 | Cisco IWAN System |
|---|---|---|
| | | network controller 140 can then estimate the amount ofbandwidth or degree of congestion for the new flow based on the historical record.")<br><br>Swenson at [0057] ("The Internet content adaption protocol is a light-weight protocol aimed at executing a simple remote proce-dure call on HTTP messages. ICAP leverages edge-based devices to help deliver value-added services using transparent HTTP proxy caches. Content adaptation refers to performing the particular value added service, such as content manipula-tion or other processing, for the associated HTTP client request/response. ICAP clients pass HTTP messages to ICAP servers for transformation or other processing. In tum, the ICAP server executes its transformation service on the HTTP messages and sends back responses to the ICAP client. At the core of this process is a cache that can proxy all client trans-actions and process them through ICAP servers, which may focus on specific functions, such as ad insertion, virus scan-ning, content translation, language translation, or content fil-tering. ICAP servers, such as those utilized by the network controller 140, handle these tasks to off-load value-added services from network devices including an ICAP client, such as the steering device 130. By offloading value added services from the steering device 130, processing infrastructure (e.g., optimization services and network controllers) may be scaled independent from the steering devices handling raw HTTP throughput.")<br><br>Swenson at [0059] ("In one embodiment, as the steering device 130 monitors network responses, it is looking for flows that match one or more signatures for video and images. When a match-ing flow is detected, the steering device 130 forwards the HTTP request and a portion of the HTTP response to the network controller 140 over the ICAP client interface 404. After receiving the request and the portion of response at the ICAP server interface 406, the flow analyzer 312 of the net-work controller 140 performs a deep flow inspection to deter-mine if the flow is worth bandwidth monitoring and/or user detection. For example, the flow inspection performed by the flow analyzer 312 may determine if the flow indeed contains large or medium object ( e.g., larger than 50 kB), and/or if the source IP address of the flow is from a user or a group of users that are required to be monitored by policies. The flow ana-lyzer 312 may also determine if the flow needs to be opti-mized based on historical flow statistical data.")<br><br>Swenson at [0060] ("If the flow is deemed of interest, the steering device 130 is notified to steer the flow through the network controller 140. This is known as the "continue" working |

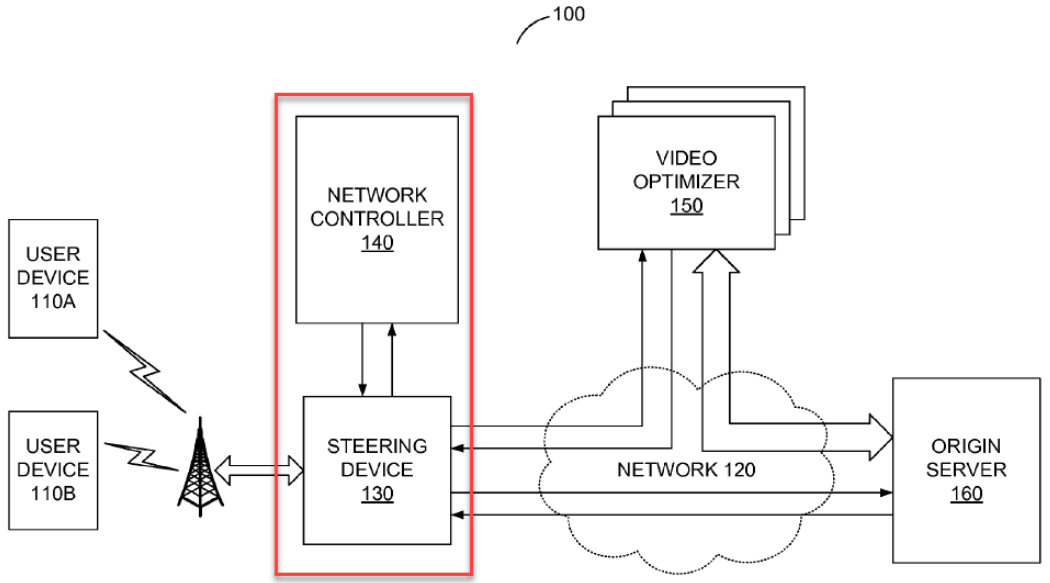


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 990 of 1100

| No. | ’111 Patent Claim 4 | Cisco IWAN System |
|---|---|---|
| | | mode for bandwidth monitoring. In the "continue" mode, the network controller 140 interfaces with the steering device 130 to func-tion, on-demand, as a traditional inline network element for flows deemed of interest. Thus, the network controller 140 ingests the network flow for inspection and subsequently forwards the network flow on the network response path. For example, for this particular flow, the origin server 160 responds to the user request by sending video or images over the network link 413 to the steering device 130, which for-wards the video or images to the network controller 140 over a network link 414. After the network controller 140 updates the flow statistics, the video or images are returned to the steering device 130 over a network link 415, which transmits the video or images to the user device 110 over the network link 416.”)<br><br>Swenson at [0071] (“After receiving the request, the video optimizer 150 forwards the video HTTP GET requests 622 to the origin server 160 and in return, receives a video file 624 from the origin server 160. The video optimizer 150 transcodes the video file to a format usable by the client device 110 based on network bandwidth available to the user device 110. The optimized video 626 is then transmitted from the video opti-mizer 150 to the steering device 130. In one embodiment, the steering device 130 intercepts the optimized video 626. The steering device 130 will then send an ICAP request to the network controller 140 for inspection. The network controller 140 deems this flow to be monitored and sends ICAP response 630. The steering device 130 then allows the flow to go through to the user device 110. The steering device 130 next sends periodic ICAP "counting" updates 632 to the network controller 140 until the flow completes. As such, the client receives the optimized video 626 for substantially real-time playback on an application executing on the user device 110.”)<br><br>Swenson at Figure 1 (annotation added) |

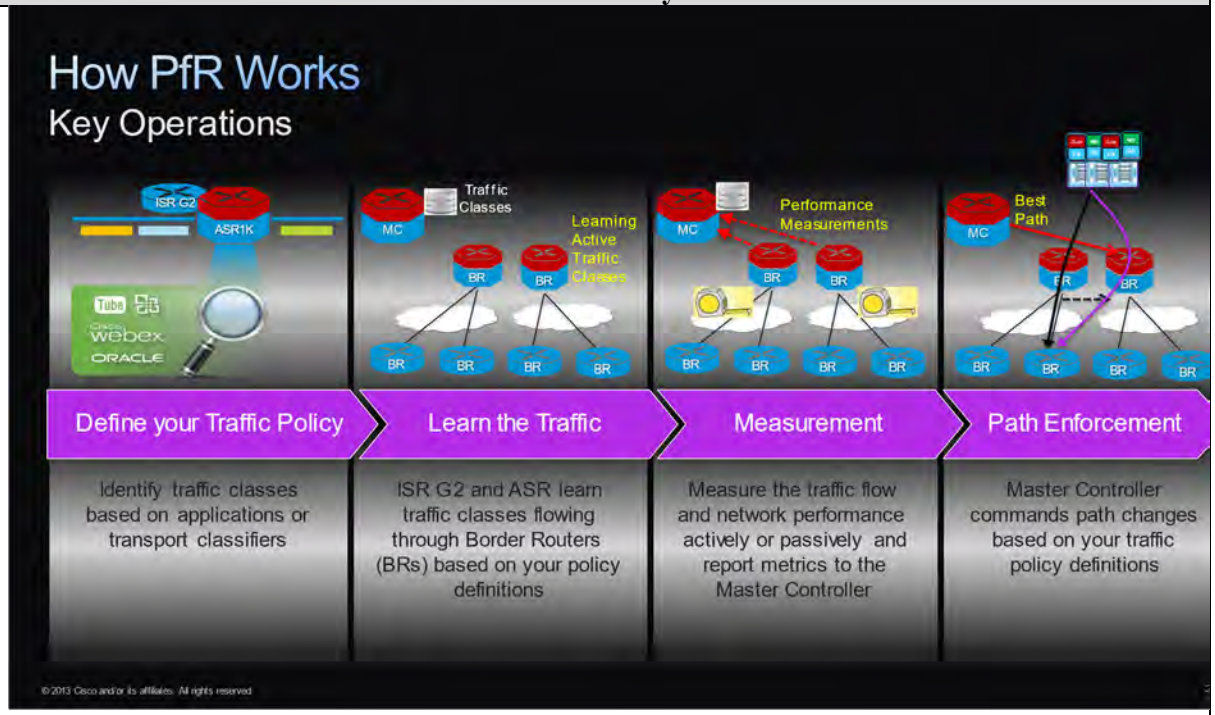


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 991 of 1100

| No. | '111 Patent Claim 4 | Cisco IWAN System |
|---|---|---|
| | |  <br>FIG. 1 |
| 4[e] | responsive to receiving the packet, sending the packet, by the network node, to the second entity. | Cisco IWAN System discloses responsive to receiving the packet, sending the packet, by the network node, to the second entity.<br><br>For example, Cisco IWAN System discloses upon receiving updated traffic policies, sending the traffic to the intended data center device.<br><br>Cisco IWAN |

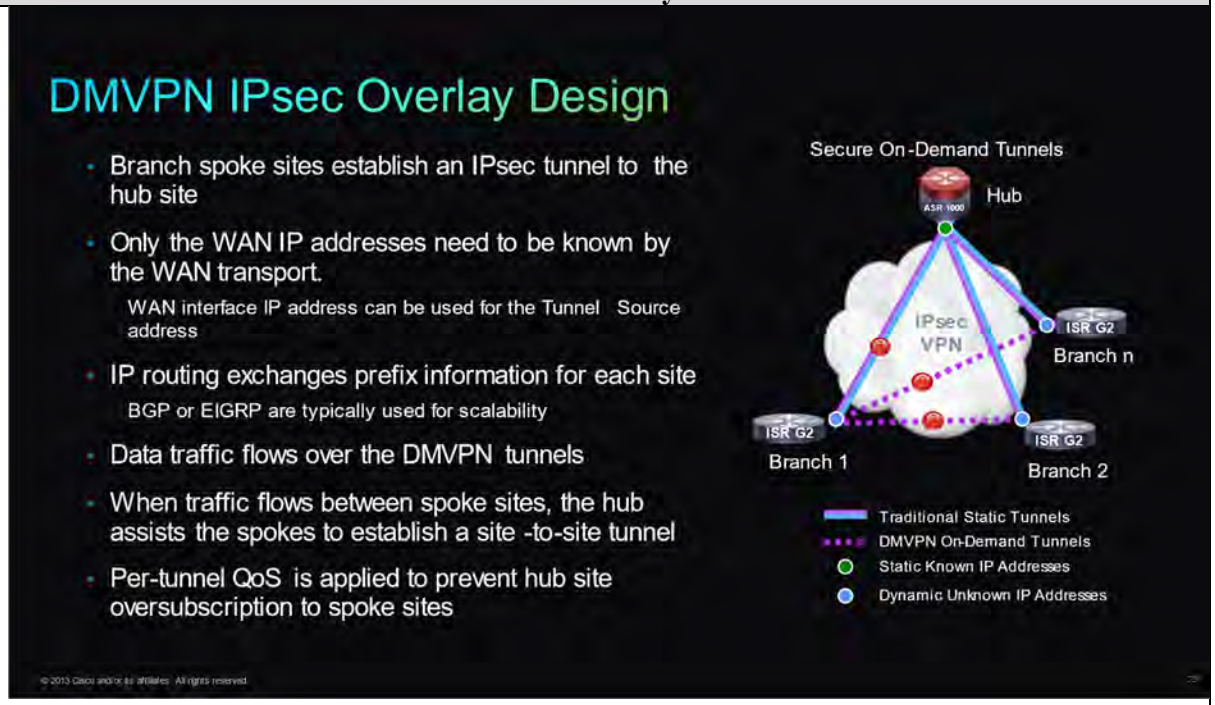


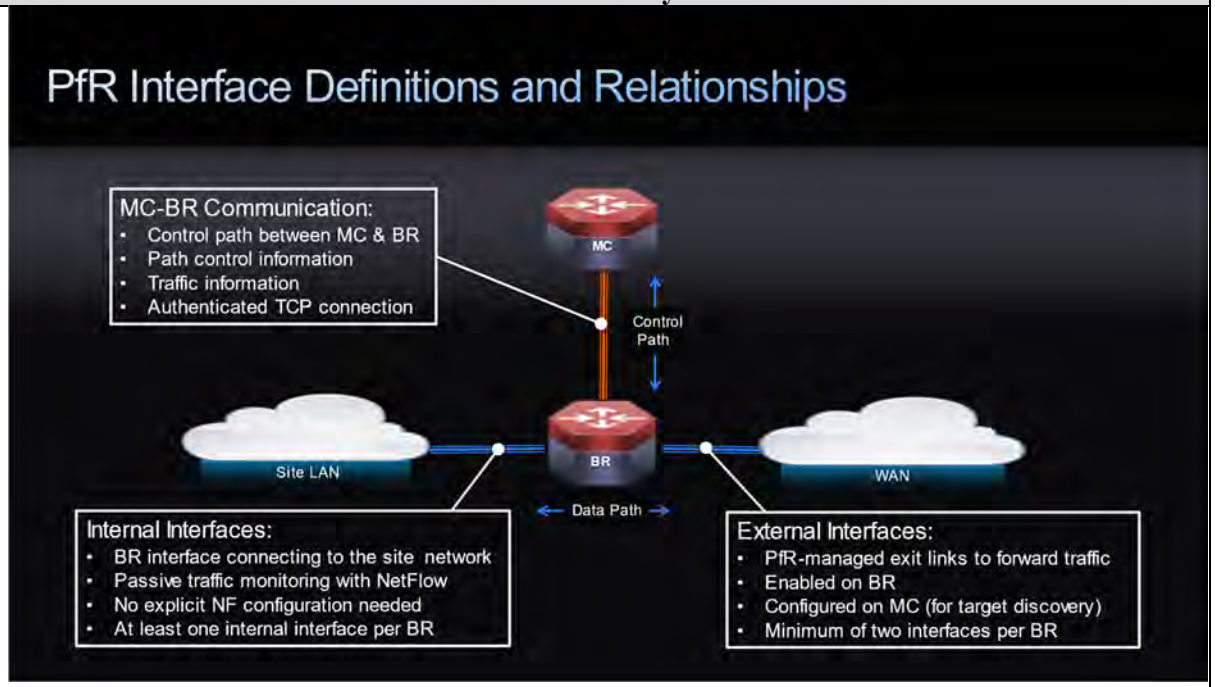
Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 992 of 1100

| No. | '111 Patent Claim 4 | Cisco IWAN System |
|---|---|---|
| | |  <br>Cisco IWAN - Uncompromised Experience |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 993 of 1100

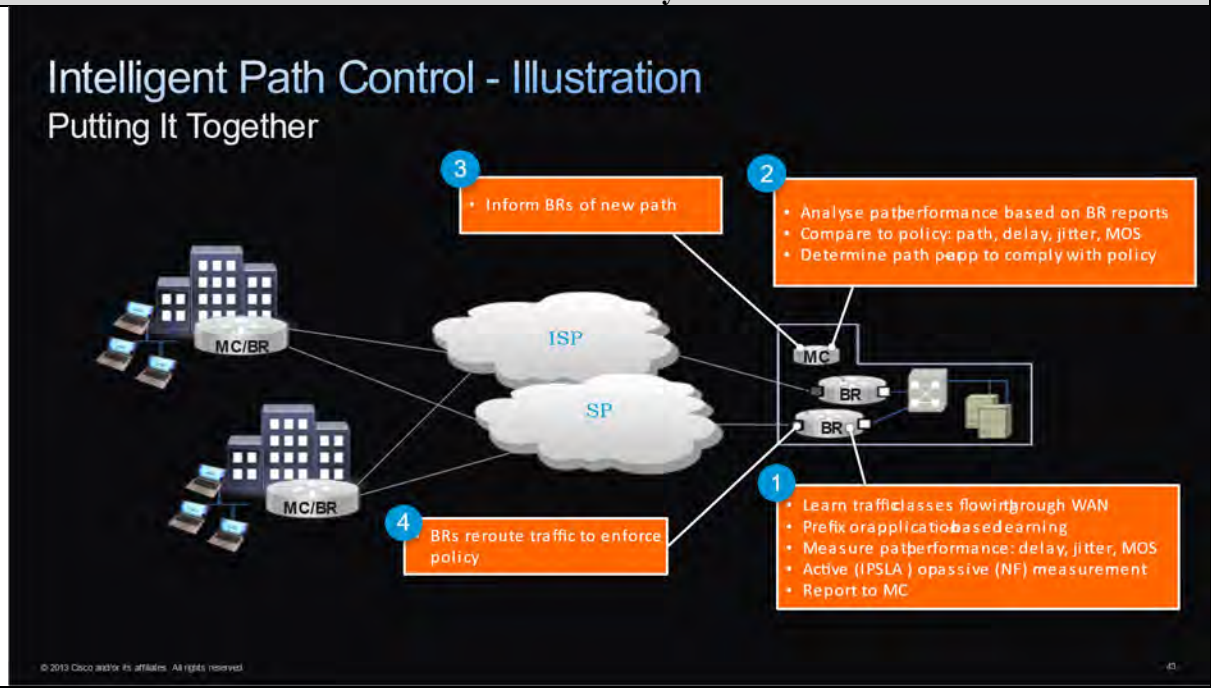
| No. | '111 Patent Claim 4 | Cisco IWAN System |
|---|---|---|
| | |  <br>Cisco IWAN - Uncompromised Experience |

| No. | '111 Patent Claim 4 | Cisco IWAN System |
|---|---|---|
| | |  |





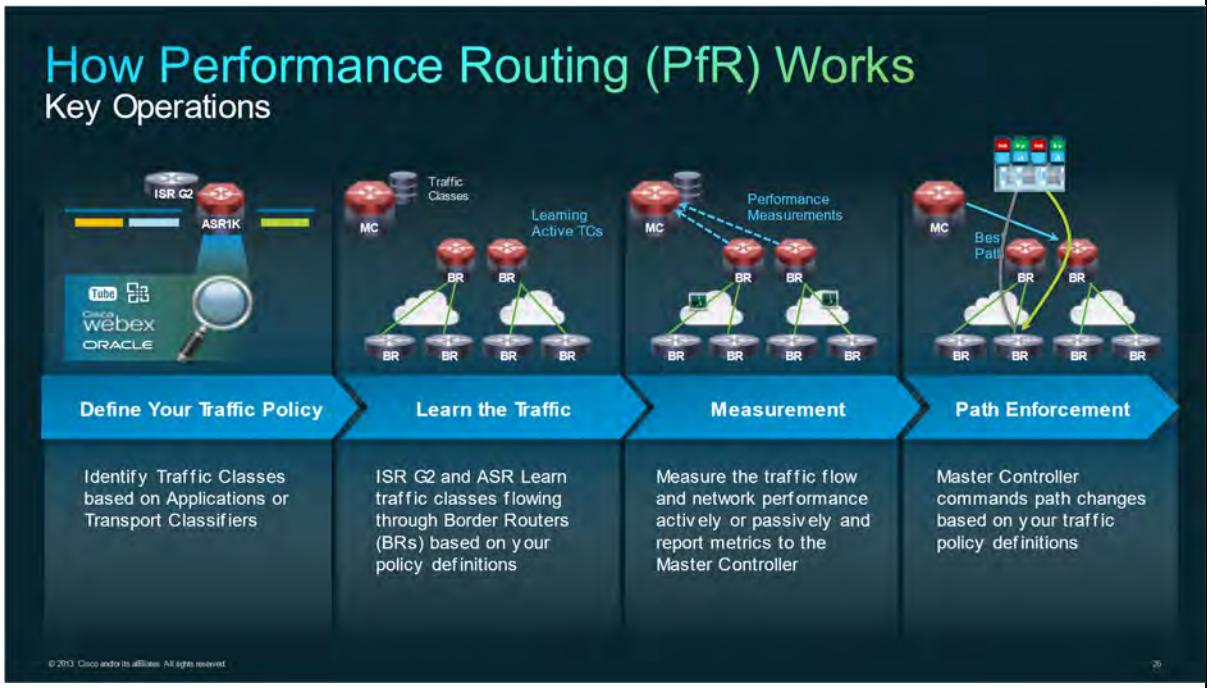
Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 995 of 1100

| No. | '111 Patent Claim 4 | Cisco IWAN System |
|---|---|---|
| | |  |

| No. | '111 Patent Claim 5 | Cisco IWAN System |
|---|---|---|
| 5 | The method according to claim 1, further comprising responsive to the packet satisfying the criterion and to the instruction, sending the packet or a portion thereof, by the network node, to the controller. | Cisco IWAN System discloses the method according to claim 1, further comprising responsive to the packet satisfying the criterion and to the instruction, sending the packet or a portion thereof, by the network node, to the controller.<br><br>For example, Cisco IWAN System discloses sending traffic flow metrics to the hub/master controller upon determining the traffic belongs to a particular traffic class. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Cisco IWAN System is found to not meet this limitation, responsive to the packet satisfying the criterion and to the instruction, sending the packet or a portion thereof, by the network node, to the controller would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>*See supra* at Claim 1. |

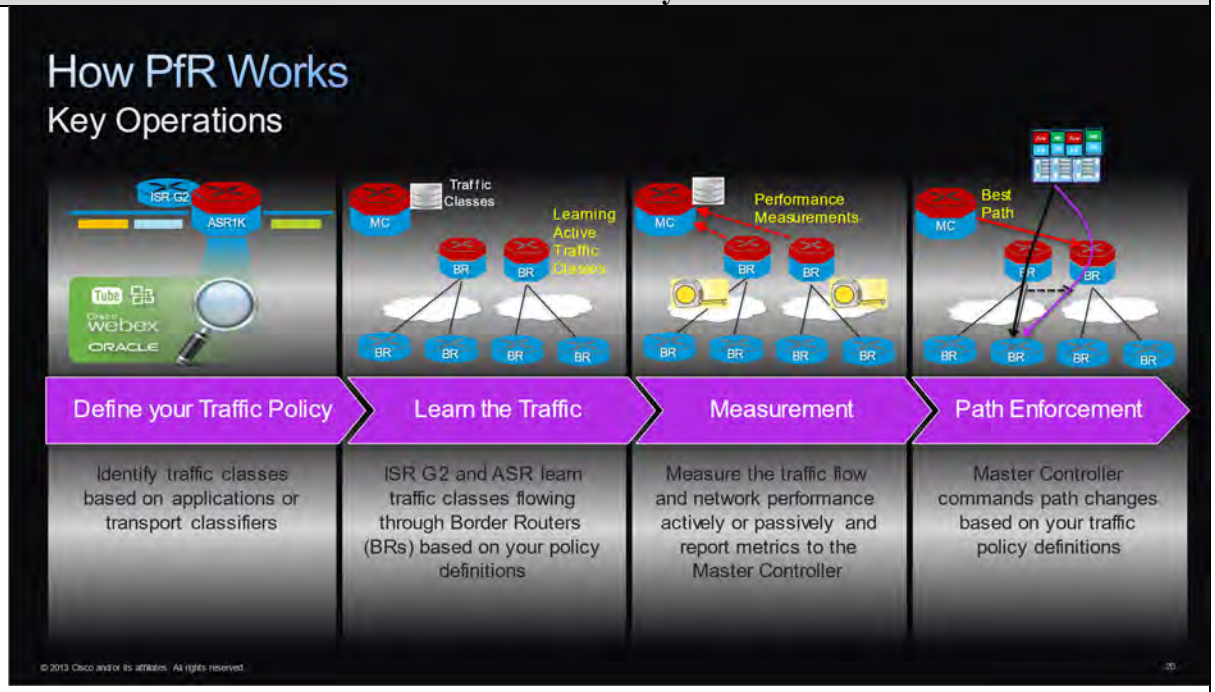


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 996 of 1100

| No. | '111 Patent Claim 5 | Cisco IWAN System |
|---|---|---|
| | | Cisco Next Generation<br>  <br><br>Cisco IWAN |

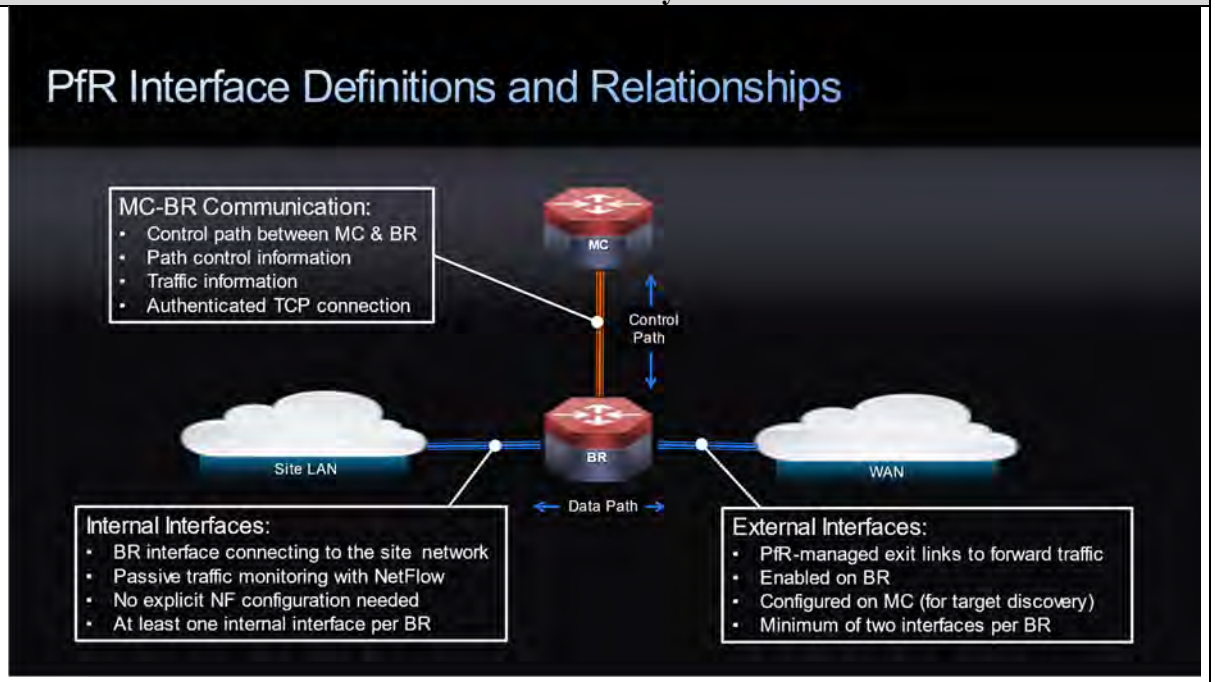


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 997 of 1100

| No. | '111 Patent Claim 5 | Cisco IWAN System |
|---|---|---|
| | |  <br>Cisco IWAN - Uncompromised Experience |



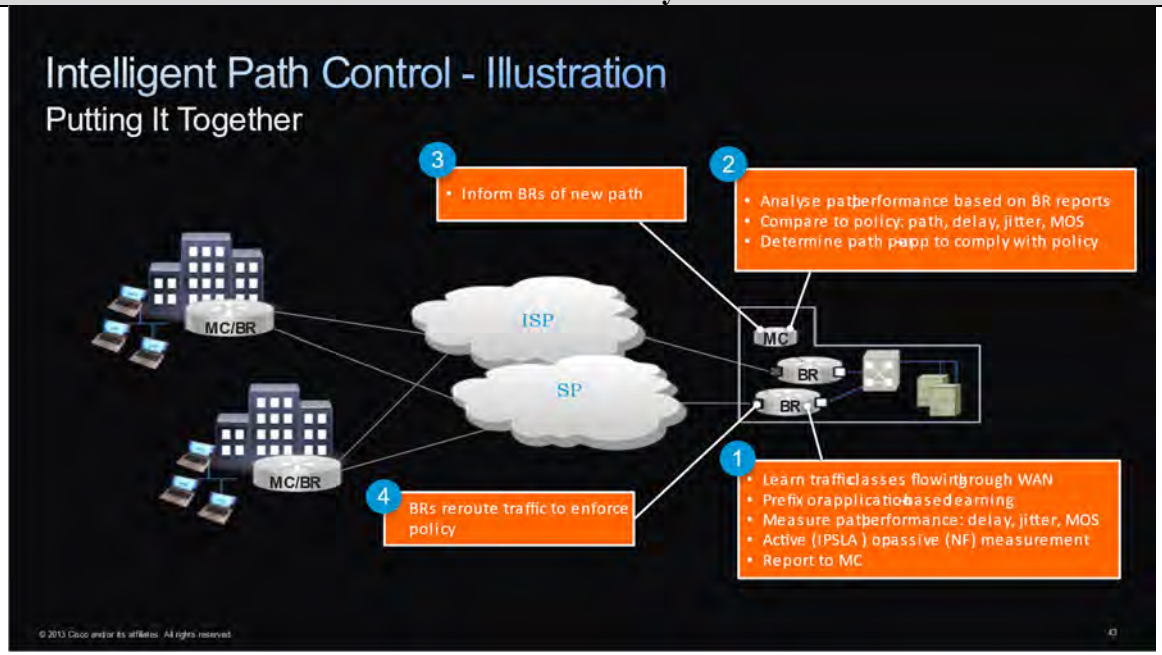
Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 998 of 1100

| No. | '111 Patent Claim 5 | Cisco IWAN System |
|---|---|---|
| | |  |





Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 999 of 1100

| No. | '111 Patent Claim 5 | Cisco IWAN System |
|---|---|---|
| | |  <br><br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Cisco IWAN System in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 5 of Exhibit E-4 renders the claim, including the present limitation, obvious. Below is an example.<br><br>For example, Copeland discloses sending packets and sampled packet headers to the intrusion detection engine on the monitoring appliance based on matching predetermined values associated with a concern index.<br><br>Copeland at [0067] ("The host servers 130 are directly or indirectly coupled to one or more network devices 135 such as routers or switches that support providing a sampled data stream such as that provided by sFlow. In a typical preferred configuration for the present invention, a monitoring appli-ance 150 operating a flow-based intrusion detection engine 155 is receiving sampled packet headers from one or more network devices 135. The |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1000 of 1100

| No. | '111 Patent Claim 5 | Cisco IWAN System |
|---|---|---|
| | | monitoring appliance 150 moni-tors the communications between the host server 130 and other hosts 120, 110 in the attempt to detect intrusion activity.")<br><br>Copeland [0079] ("Large packets tend to be fragmented by networks that cannot handle a large packet size. A 16-bit packet identification is used to reassemble fragmented packets. Three one-bit set of fragmentation flags control whether a packet is or may be fragmented. The 13-bit fragment offset is a sequence number for the 4-byte words in the packet when reassembled. In a series of fragments, the first offset will be zero.")<br><br>Copeland at [0097] ("The described TCP session 300 of FIG. 3 is a generic TCP session in which a network might engage. In accordance with the invention, flow data is collected about the session to help determine if the communication is abnormal. In the preferred embodiment, information such as the total number of packets sent, the total amount of data sent, the session start time and duration, and the TCP flags set in all of the packets, are collected, stored in the database 160, and analyzed to determine if the communication was suspicious. If a communication is deemed suspicious, i.e. it meets predetermined criteria, a predetermined concern index value associated with a determined category of suspicious activity is added to the cumulated CI value associated with the host that made the communication.")<br><br>Copeland at [0120] ("The sampled packet headers sent from the sFlow agent are captured and processed by the sample packet collector 505 in order to create a "Packet Data" data struc-ture that includes the sFlow agent source of the packets, the header of the sampled packets, and other information avail-able from the sFlow data stream that may be important. For<br>example, one data field that is optionally available pr vides the username of the user using the computer at the time of the communications. This information is extremely useful in some environments subject to regulatory requirements and monitoring of the communications on the network. In this case the username will be stored as "supplementary infor-mation" for auditing purposes in the flow data. Other infor-mation, including the sampling device and the physical port on which the communications was detected may also be retained for other uses such as mitigation, where a host may be removed from the network.") |


~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1001 of 1100

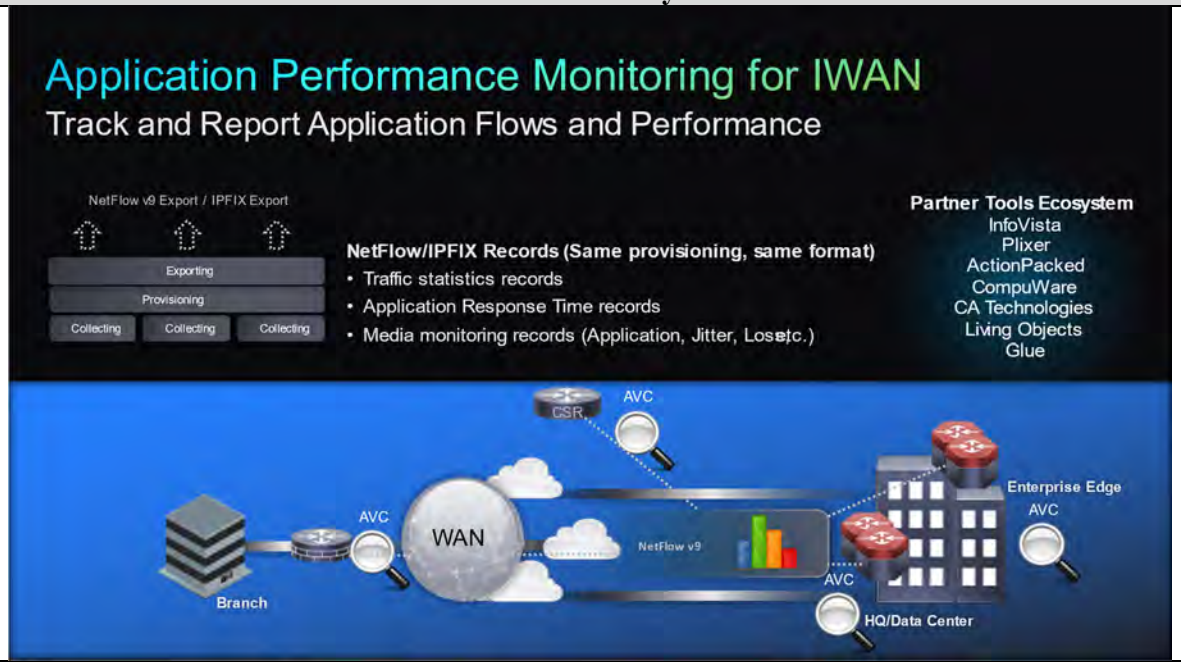
| No. | '111 Patent Claim 5 | Cisco IWAN System |
|---|---|---|
| | | Copeland at [0126]-[0129] ("If a particular packet 101 being processed by the packet classifier 510 matches a particular entry or record in the flow data structure 162, data from that particular packet 101 is used to update the statistics in the corresponding flow data structure record. A packet 101 is considered to match to a flow data structure record if both IP numbers match and the source of the sampled packet matches and:<br><br>(1) both port numbers match and no port is marked as the "server" port, or<br>(2) the port number previously marked as the "server" port matches, or<br>(3) one of the port numbers matches, but the other does not, and the neither port number has been marked as the server port (in this case the matching port number is marked as the "server" port).")<br><br>Copeland at [0144] ("Concern index (CI) values calculated from packet anomalies also add to a host's accumulated concern index value. Table II of FIG. 7 shows one scheme for assigning concern index values due to other events revealed by the flow analysis. For example, there are many combinations of TCP flag bits that are rarely or never seen in valid TCP connections. When the packet classifier thread 510 recog-nizes one of these combinations, it directly adds a predeter-mined value to the sending host's accumulated concern index value. When the packet classifier thread 510 searches along the flow linked-list (i.e. flow data 162) for a match to the current packet 101, it keeps count of the number of flows active with matching IP addresses but no matching port number. If this number exceeds a predetermined threshold value (e.g., 4) and is greater than the previous number noticed, CI is added for an amount corresponding to a "port scan." A bit in the host record is set to indicate that the host has received CI for "port scanning."")<br><br>Copeland at [0150] ("A preferred hardware configuration 800 of an embodiment that executes the functions of the above-described flow-based engine is described in reference to FIG. 8. FIG. 8 illustrates a typically hardware configuration 800 for a network intrusion detection system. A monitoring appliance 150 serves as a pass-by filter of network traffic. A network device 135, such as a router or switch supporting sFlow provides the location for connecting the monitoring appliance 150 to the network 899 for monitoring the network traffic.") |

| No. | '111 Patent Claim 5 | Cisco IWAN System |
|---|---|---|
| | | Copeland at [0159]-[0162] ("A packet 101 is considered to match to a flow data structure record if both IP numbers match and the source of the sampled data matches and:<br><br>(a). both port numbers match and no port is marked as the "server" port, or<br>(b). the port number previously marked as the "server" port matches, or<br>(c). one of the port numbers matches, but the other does not, and the neither port number has been marked as the server port (in this case the matching port number is marked as the "server" port).") |

| No. | '111 Patent Claim 6 | Cisco IWAN System |
|---|---|---|
| 6 | The method according to claim 5, further comprising storing the received packet or a portion thereof, by the controller, in a memory. | Cisco IWAN System discloses the method according to claim 5, further comprising storing the received packet or a portion thereof, by the controller, in a memory.<br><br>For example, Cisco IWAN System discloses storing traffic statistics and monitoring records of the traffic flow. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met.<br><br>*See supra* at Claim 5.<br><br>Cisco IWAN |



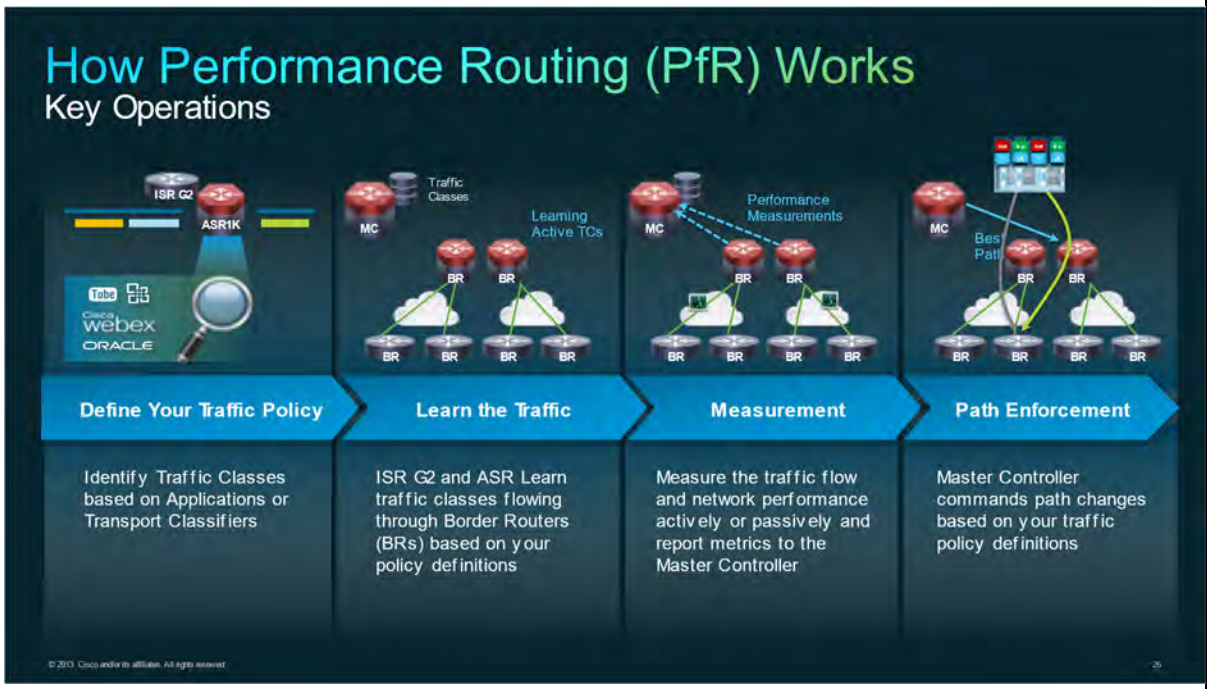
Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1003 of 1100

| No. | '111 Patent Claim 6 | Cisco IWAN System |
|---|---|---|
| | |  |

| No. | '111 Patent Claim 7 | Cisco IWAN System |
|---|---|---|
| 7 | The method according to claim 5, further comprising responsive to the packet satisfying the criterion and to the instruction, sending a portion of the packet, by the network node, to the controller. | Cisco IWAN System discloses the method according to claim 5, further comprising responsive to the packet satisfying the criterion and to the instruction, sending a portion of the packet, by the network node, to the controller.<br><br>For example, Cisco IWAN System discloses sending traffic flow metrics to the hub/master controller upon determining the traffic belongs to a particular traffic class. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Cisco IWAN System is found to not meet this limitation, responsive to the packet satisfying the criterion and to instruction, sending a portion of the packet, by the network node, to the controller would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>*See supra* at Claim 5. |

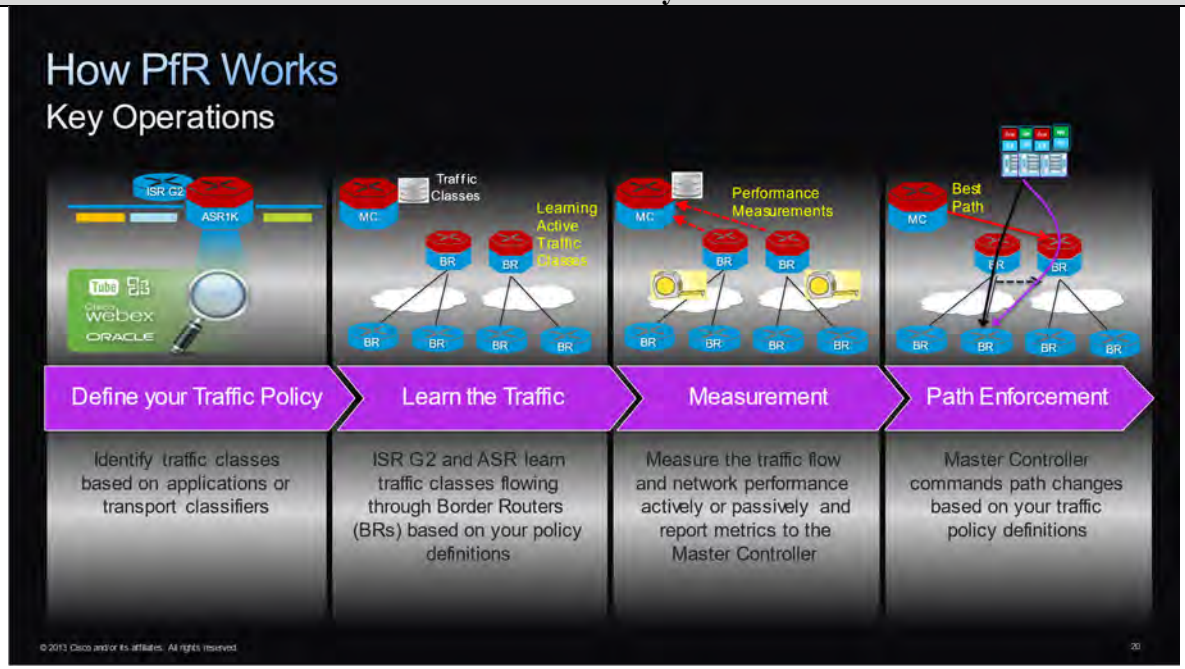


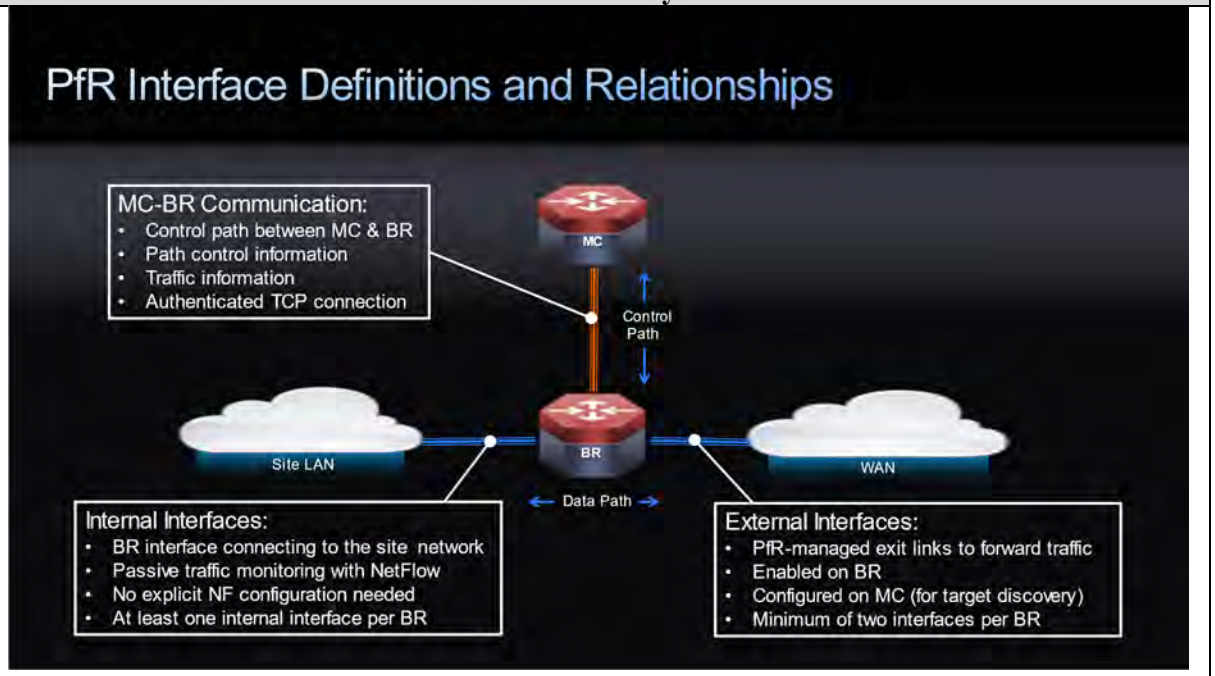
Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1004 of 1100

| No. | '111 Patent Claim 7 | Cisco IWAN System |
|---|---|---|
| | | Cisco Next Generation<br>  <br><br>Cisco IWAN |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1005 of 1100

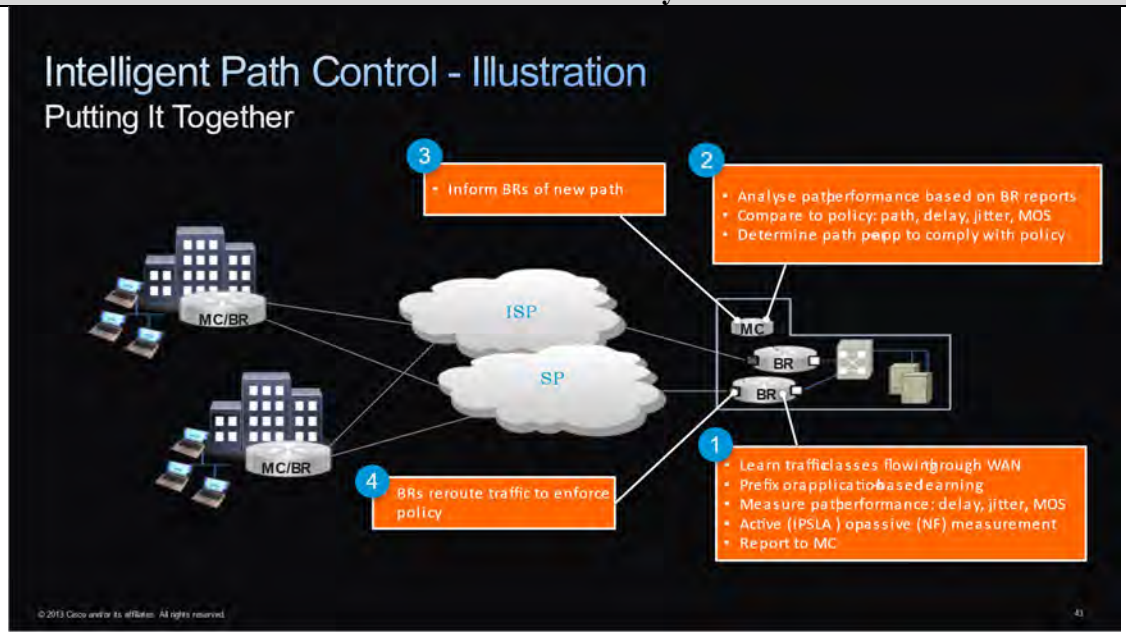
| No. | '111 Patent Claim 7 | Cisco IWAN System |
|---|---|---|
| | |  <br>Cisco IWAN - Uncompromised Experience |

| No. | '111 Patent Claim 7 | Cisco IWAN System |
|---|---|---|
| | |  |





Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1007 of 1100

| No. | '111 Patent Claim 7 | Cisco IWAN System |
|---|---|---|
| | |  <br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Cisco IWAN System in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 5 of Exhibit E-4 renders the claim, including the present limitation, obvious. Below is an example.<br><br>For example, Copeland discloses sending packets and sampled packet headers to the intrusion detection engine on the monitoring appliance based on matching predetermined values associated with a concern index.<br><br>Copeland at [0067] ("The host servers 130 are directly or indirectly coupled to one or more network devices 135 such as routers or switches that support providing a sampled data stream such as that provided by sFlow. In a typical preferred configuration for the present invention, a monitoring appli-ance 150 operating a flow-based intrusion detection engine 155 is receiving sampled packet headers from one or more network devices 135. The |

| No. | '111 Patent Claim 7 | Cisco IWAN System |
|---|---|---|
| | | monitoring appliance 150 moni-tors the communications between the host server 130 and other hosts 120, 110 in the attempt to detect intrusion activity.")<br><br>Copeland [0079] ("Large packets tend to be fragmented by networks that cannot handle a large packet size. A 16-bit packet identification is used to reassemble fragmented packets. Three one-bit set of fragmentation flags control whether a packet is or may be fragmented. The 13-bit fragment offset is a sequence number for the 4-byte words in the packet when reassembled. In a series of fragments, the first offset will be zero.")<br><br>Copeland at [0097] ("The described TCP session 300 of FIG. 3 is a generic TCP session in which a network might engage. In accordance with the invention, flow data is collected about the session to help determine if the communication is abnormal. In the preferred embodiment, information such as the total number of packets sent, the total amount of data sent, the session start time and duration, and the TCP flags set in all of the packets, are collected, stored in the database 160, and analyzed to determine if the communication was suspicious. If a communication is deemed suspicious, i.e. it meets predetermined criteria, a predetermined concern index value associated with a determined category of suspicious activity is added to the cumulated CI value associated with the host that made the communication.")<br><br>Copeland at [0120] ("The sampled packet headers sent from the sFlow agent are captured and processed by the sample packet collector 505 in order to create a "Packet Data" data struc-ture that includes the sFlow agent source of the packets, the header of the sampled packets, and other information avail-able from the sFlow data stream that may be important. For<br>example, one data field that is optionally available pr vides the username of the user using the computer at the time of the communications. This information is extremely useful in some environments subject to regulatory requirements and monitoring of the communications on the network. In this case the username will be stored as "supplementary infor-mation" for auditing purposes in the flow data. Other infor-mation, including the sampling device and the physical port on which the communications was detected may also be retained for other uses such as mitigation, where a host may be removed from the network.") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1009 of 1100

| No. | '111 Patent Claim 7 | Cisco IWAN System |
|---|---|---|
| | | Copeland at [0126]-[0129] ("If a particular packet 101 being processed by the packet classifier 510 matches a particular entry or record in the flow data structure 162, data from that particular packet 101 is used to update the statistics in the corresponding flow data structure record. A packet 101 is considered to match to a flow data structure record if both IP numbers match and the source of the sampled packet matches and:<br><br>(1) both port numbers match and no port is marked as the "server" port, or<br>(2) the port number previously marked as the "server" port matches, or<br>(3) one of the port numbers matches, but the other does not, and the neither port number has been marked as the server port (in this case the matching port number is marked as the "server" port).")<br><br>Copeland at [0144] ("Concern index (CI) values calculated from packet anomalies also add to a host's accumulated concern index value. Table II of FIG. 7 shows one scheme for assigning concern index values due to other events revealed by the flow analysis. For example, there are many combinations of TCP flag bits that are rarely or never seen in valid TCP connections. When the packet classifier thread 510 recog-nizes one of these combinations, it directly adds a predeter-mined value to the sending host's accumulated concern index value. When the packet classifier thread 510 searches along the flow linked-list (i.e. flow data 162) for a match to the current packet 101, it keeps count of the number of flows active with matching IP addresses but no matching port number. If this number exceeds a predetermined threshold value (e.g., 4) and is greater than the previous number noticed, CI is added for an amount corresponding to a "port scan." A bit in the host record is set to indicate that the host has received CI for "port scanning."")<br><br>Copeland at [0150] ("A preferred hardware configuration 800 of an embodiment that executes the functions of the above-described flow-based engine is described in reference to FIG. 8. FIG. 8 illustrates a typically hardware configuration 800 for a network intrusion detection system. A monitoring appliance 150 serves as a pass-by filter of network traffic. A network device 135, such as a router or switch supporting sFlow provides the location for connecting the monitoring appliance 150 to the network 899 for monitoring the network traffic.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1010 of 1100

| No. | '111 Patent Claim 7 | Cisco IWAN System |
|---|---|---|
| | | Copeland at [0159]-[0162] ("A packet 101 is considered to match to a flow data structure record if both IP numbers match and the source of the sampled data matches and:<br><br>(a). both port numbers match and no port is marked as the "server" port, or<br>(b). the port number previously marked as the "server" port matches, or<br>(c). one of the port numbers matches, but the other does not, and the neither port number has been marked as the server port (in this case the matching port number is marked as the "server" port).") |

| No. | '111 Patent Claim 8 | Cisco IWAN System |
|---|---|---|
| 8[a] | The method according to claim 7, wherein the portion of the packet consists of multiple consecutive bytes, and | Cisco IWAN System discloses the method according to claim 7, wherein the portion of the packet consists of multiple consecutive bytes.<br><br>On information and belief, Cisco IWAN System discloses fragmenting packets into smaller byte sizes. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Cisco IWAN System is found to not meet this limitation, wherein the portion of the packet consists of multiple consecutive bytes would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>*See supra* at Claim 7.<br><br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Cisco IWAN System in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 8(a) of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references.<br><br>For example, Kempf discloses consecutive bytes of a packet header field. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1011 of 1100

| No. | '111 Patent Claim 8 | Cisco IWAN System |
|---|---|---|
| | | Kempf at [0081] ("In one embodiment, OpenFlow is modified to pro-vide rules for GTP TEID Routing. FIG. 17 is a diagram of one embodiment of the OpenFlow flow table modification for GTP TEID routing. An OpenFlow switch that supports TEID routing matches on the 2 byte (16 bit) collection of header fields and the 4 byte (32 bit) GTP TEID, in addition to other OpenFlow header fields, in at least one flow table ( e.g., the first flow table). The GTP TEID flag can be wildcarded (i.e. matches are "don't care"). In one embodiment, the EPC pro-tocols do not assign any meaning to TEIDs other than as an endpoint identifier for tunnels, like ports in standard UDP/ TCP transport protocols. In other embodiments, the TEIDs can have a correlated meaning or semantics. The GTP header flags field can also be wildcarded, this can be partially matched by combining the following bitmasks: 0xFF00- Match the Message Type field; 0xe0-Match the Version field; 0xl0-Match the PT field; 0x04-Match the E field; 0x02- Match the S field; and 0x0l-Match the PN field.")<br><br>Kempf at [0083] ("If a packet either needs encapsulation or arrives encapsulated with nonzero header flags, header extensions, and/or the GTP-U packet is not a G-PDU packet (i.e. it is a GTP-U control packet), the processing must proceed via the gateway's slow path (software) control plane. GTP-C and GTP' packets directed to the gateway's IP address are a result of mis-configuration and are in error. They must be sent to the OpenFlow controller, since these packets are handled by the S-GW-C and P-GW-C control plane entities in the cloud computing system or to the billing entity handling GTP' and not the S-GW-D and P-GW-D data plane switches.")<br><br>Kempf at [0087] ("In one embodiment, slow path support for GTP is implemented with an OpenFlow gateway switch. An Open-Flow mobile gateway switch also contains support on the software control plane for slow path packet processing. This path is taken by G-PDU (message type 255) packets with nonzero header fields or extension headers, and user data plane packets requiring encapsulation with such fields or addition of extension headers, and by G TP-U control packets. For this purpose, the switch supports three local ports in the software control plane: LOCAL_GTP _CONTROL-the switch fast path forwards GTP encapsulated packets directed to the gateway IP address that contain GTP-U control mes-sages and the local switch software control plane initiates local control plane actions depending on the GTP-U control message; LOCAL_GTP _U_DECAP-the switch fast path forwards G-PDU packets to this port that have nonzero header fields or extension headers |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1012 of 1100

| No. | '111 Patent Claim 8 | Cisco IWAN System |
|---|---|---|
| | | (i.e. E!=0, S!=0, or PN!=0). These packets require specialized handling. The local switch software slow path processes the packets and performs the specialized handling; and LOCAL_GTP _U_ENCAP-the switch fast path forwards user data plane packets to this port that require encapsulation in a GTP tunnel with nonzero header fields or extension headers (i.e. E!=0, S!=0, or PN!=0). These packets require specialized handling. The local switch software slow path encapsulates the packets and performs the specialized handling. In addition to forwarding the packet, the switch fast path makes the OpenFlow metadata field avail-able to the slow path software.")<br><br>Kempf at [0091] ("When a packet header matches a rule associated with the virtual port, the GTP TEID is written into the lower 32 bits of the metadata and the packet is directed to the virtual port. The virtual port calculates the hash of the TEID and looks up the tunnel header information in the tunnel header table. If no such tunnel information is present, the packet is forwarded to the controller with an error indication. Other-wise, the virtual port constructs a GTP tunnel header and encapsulates the packet. Any DSCP bits or VLAN priority bits are additionally set in the IP or MAC tunnel headers, and any VLAN tags or MPLS labels are pushed onto the packet. The encapsulated packet is forwarded out the physical port to which the virtual port is bound.")<br><br>Kempf at [0092] ("In one embodiment, the system implements a GTP fast path decapsulation virtual port. When requested by the S-GW and P-GW control plane software running in the cloud computing system, the gateway switch installs rules and actions for routing GTP encapsulated packets out of GTP tunnels. The rules match the GTP header flags and the GTP TEID for the packet, in the modified OpenFlow flow table shown in FIG. 17 as follows: the IP destination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); the UDP destination port is the GTP-U destination port (2152); and the header fields and message type field is wildcarded with the flag 0XFFF0 and the upper two bytes of the field match the G-PDU message type (255) while the lower two bytes match 0x30, i.e. the packet is a GTP packet not a GTP' packet and the version number is 1.")<br><br>Kempf at [0098] ("The header flags and message type fields for the three rules are wildcarded with the following bitmasks and match as follows: bitmask 0xFFF4 and the upper two bytes match the G-PDU message type (255) while the lower two bytes are Ox34, |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1013 of 1100

| No. | '111 Patent Claim 8 | Cisco IWAN System |
|---|---|---|
| | | indicating that the version number is 1, the packet is a GTP packet, and there is an extension header present; bitmask 0xFFFF2 and the upper two bytes match the G-PDU message type (255) while the lower two bytes are 0x32, indicating that the version number is 1, the packet is a GTP packet, and there is a sequence number bitmask 0xFF0l and the upper two bytes match the G-PDU message type (255) while the lower two bytes are 0x31, indicating that the version number is 1, the packet is a GTP packet, and a N-PDU is present.")<br><br>Kempf at [0101] ("In one embodiment, the system implements han-dling of user data plane packets requiring GTP-U encapsula-tion with extension headers, sequence numbers, and N-PDU numbers. User data plane packets that require extension head-ers, sequence numbers, or N-PDU numbers during GTP encapsulation require special handling by the software slow path. For these packets, the OpenFlow controller programs a rule matching the 4 tuple: IP source address; IP destination address; UDP/TCP/SCTP source port; and UDP/TCP/SCTP destination port. The instructions for matching packets are:<br><br>Write-Metadata ( GTP-TEID, 0x FFFFFFFF)<br>Apply-Actions (Set-Output-Port LOCAL_GTP _U_ENCAP )")<br><br>For example, Copeland discloses fragmenting packets into smaller byte sizes, including headers and flags. Copeland further discloses sending sampled packet headers, consisting of fragmented packets of consecutive bytes to the monitoring device.<br><br>Copeland [0079] ("Large packets tend to be fragmented by networks that cannot handle a large packet size. A 16-bit packet identification is used to reassemble fragmented packets. Three one-bit set of fragmentation flags control whether a packet is or may be fragmented. The 13-bit fragment offset is a sequence number for the 4-byte words in the packet when reassembled. In a series of fragments, the first offset will be zero.") |
| 8[b] | wherein the instruction comprises identification of the consecutive bytes in the packet. | Cisco IWAN System discloses wherein the instruction comprises identification of the consecutive bytes in the packet.<br><br>On information and belief, Cisco IWAN System discloses wherein the instruction comprises identification of the consecutive bytes in the packet. Thus, at least under the apparent claim |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1014 of 1100

| No. | '111 Patent Claim 8 | Cisco IWAN System |
|---|---|---|
| | | scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Cisco IWAN System is found to not meet this limitation, coupling each of the one or more interface modules to a communication network using a second group of second physical links arranged in parallel would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Cisco IWAN System in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 8(b) of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references.<br><br>For example, Kempf discloses rules in which the flow table includes matching to the consecutive bytes of a packet header.<br><br>Kempf at [0081] ("In one embodiment, OpenFlow is modified to pro-vide rules for GTP TEID Routing. FIG. 17 is a diagram of one embodiment of the OpenFlow flow table modification for GTP TEID routing. An OpenFlow switch that supports TEID routing matches on the 2 byte (16 bit) collection of header fields and the 4 byte (32 bit) GTP TEID, in addition to other OpenFlow header fields, in at least one flow table ( e.g., the first flow table). The GTP TEID flag can be wildcarded (i.e. matches are "don't care"). In one embodiment, the EPC pro-tocols do not assign any meaning to TEIDs other than as an endpoint identifier for tunnels, like ports in standard UDP/ TCP transport protocols. In other embodiments, the TEIDs can have a correlated meaning or semantics. The GTP header flags field can also be wildcarded, this can be partially matched by combining the following bitmasks: 0xFF00- Match the Message Type field; 0xe0-Match the Version field; 0xl0-Match the PT field; 0x04-Match the E field; 0x02- Match the S field; and 0x0l-Match the PN field.")<br><br>Kempf at [0083] ("If a packet either needs encapsulation or arrives encapsulated with nonzero header flags, header extensions, and/or the GTP-U packet is not a G-PDU packet (i.e. it is a GTP-U control packet), the processing must proceed via the gateway's slow path (software) control plane. GTP-C and GTP' packets directed to the gateway's IP address are a result of mis-configuration and are in error. They must be sent to the OpenFlow controller, |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1015 of 1100

| No. | '111 Patent Claim 8 | Cisco IWAN System |
|---|---|---|
| | | since these packets are handled by the S-GW-C and P-GW-C control plane entities in the cloud computing system or to the billing entity handling GTP' and not the S-GW-D and P-GW-D data plane switches.")<br><br>Kempf at [0087] ("In one embodiment, slow path support for GTP is implemented with an OpenFlow gateway switch. An Open-Flow mobile gateway switch also contains support on the software control plane for slow path packet processing. This path is taken by G-PDU (message type 255) packets with nonzero header fields or extension headers, and user data plane packets requiring encapsulation with such fields or addition of extension headers, and by G TP-U control packets. For this purpose, the switch supports three local ports in the software control plane: LOCAL_GTP _CONTROL-the switch fast path forwards GTP encapsulated packets directed to the gateway IP address that contain GTP-U control mes-sages and the local switch software control plane initiates local control plane actions depending on the GTP-U control message; LOCAL_GTP _U_DECAP-the switch fast path forwards G-PDU packets to this port that have nonzero header fields or extension headers (i.e. E!=0, S!=0, or PN!=0). These packets require specialized handling. The local switch software slow path processes the packets and performs the specialized handling; and LOCAL_GTP _U_ENCAP-the switch fast path forwards user data plane packets to this port that require encapsulation in a GTP tunnel with nonzero header fields or extension headers (i.e. E!=0, S!=0, or PN!=0). These packets require specialized handling. The local switch software slow path encapsulates the packets and performs the specialized handling. In addition to forwarding the packet, the switch fast path makes the OpenFlow metadata field avail-able to the slow path software.")<br><br>Kempf at [0091] ("When a packet header matches a rule associated with the virtual port, the GTP TEID is written into the lower 32 bits of the metadata and the packet is directed to the virtual port. The virtual port calculates the hash of the TEID and looks up the tunnel header information in the tunnel header table. If no such tunnel information is present, the packet is forwarded to the controller with an error indication. Other-wise, the virtual port constructs a GTP tunnel header and encapsulates the packet. Any DSCP bits or VLAN priority bits are additionally set in the IP or MAC tunnel headers, and any VLAN tags or MPLS labels are pushed onto the packet. The encapsulated packet is forwarded out the physical port to which the virtual port is bound.") |



~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1016 of 1100

| No. | '111 Patent Claim 8 | Cisco IWAN System |
|---|---|---|
| | | Kempf at [0092] ("In one embodiment, the system implements a GTP fast path decapsulation virtual port. When requested by the S-GW and P-GW control plane software running in the cloud computing system, the gateway switch installs rules and actions for routing GTP encapsulated packets out of GTP tunnels. The rules match the GTP header flags and the GTP TEID for the packet, in the modified OpenFlow flow table shown in FIG. 17 as follows: the IP destination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); the UDP destination port is the GTP-U destination port (2152); and the header fields and message type field is wildcarded with the flag 0XFFF0 and the upper two bytes of the field match the G-PDU message type (255) while the lower two bytes match 0x30, i.e. the packet is a GTP packet not a GTP' packet and the version number is 1.")<br><br>Kempf at [0098] ("The header flags and message type fields for the three rules are wildcarded with the following bitmasks and match as follows: bitmask 0xFFF4 and the upper two bytes match the G-PDU message type (255) while the lower two bytes are Ox34, indicating that the version number is 1, the packet is a GTP packet, and there is an extension header present; bitmask 0xFFFF2 and the upper two bytes match the G-PDU message type (255) while the lower two bytes are 0x32, indicating that the version number is 1, the packet is a GTP packet, and there is a sequence number bitmask 0xFF0l and the upper two bytes match the G-PDU message type (255) while the lower two bytes are 0x31, indicating that the version number is 1, the packet is a GTP packet, and a N-PDU is present.")<br><br>Kempf at [0101] ("In one embodiment, the system implements han-dling of user data plane packets requiring GTP-U encapsula-tion with extension headers, sequence numbers, and N-PDU numbers. User data plane packets that require extension head-ers, sequence numbers, or N-PDU numbers during GTP encapsulation require special handling by the software slow path. For these packets, the OpenFlow controller programs a rule matching the 4 tuple: IP source address; IP destination address; UDP/TCP/SCTP source port; and UDP/TCP/SCTP destination port. The instructions for matching packets are:<br><br>Write-Metadata ( GTP-TEID, 0x FFFFFFFF)<br>Apply-Actions (Set-Output-Port LOCAL_GTP _U_ENCAP )") |



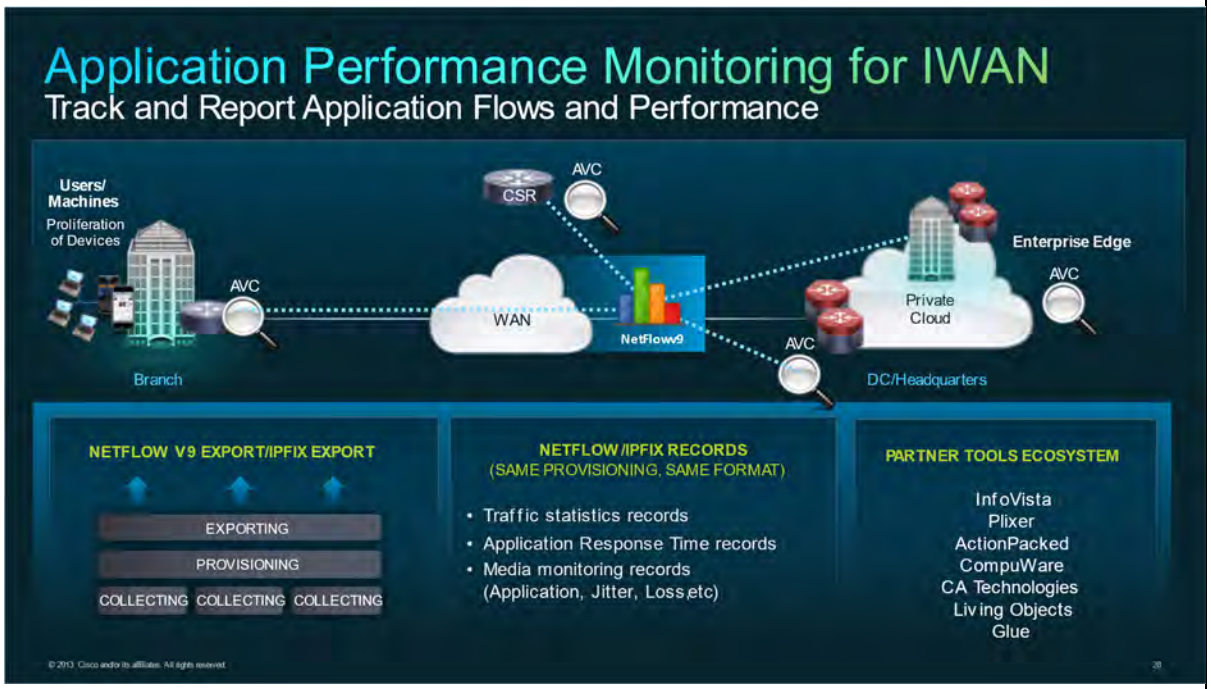
Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1017 of 1100

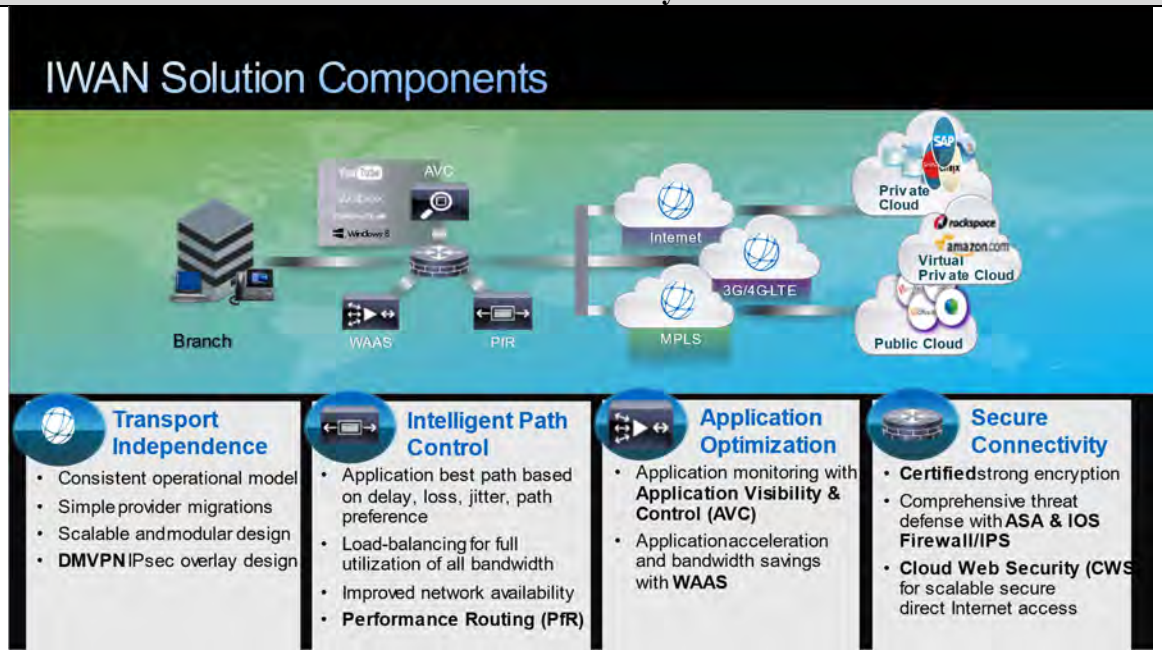
| No. | '111 Patent Claim 8 | Cisco IWAN System |
|---|---|---|
| | | For example, Copeland discloses identifying the sampled packet headers comprised of fragmented packets of smaller byte sizes.<br><br>Copeland [0079] ("Large packets tend to be fragmented by networks that cannot handle a large packet size. A 16-bit packet identification is used to reassemble fragmented packets. Three one-bit set of fragmentation flags control whether a packet is or may be fragmented. The 13-bit fragment offset is a sequence number for the 4-byte words in the packet when reassembled. In a series of fragments, the first offset will be zero.")<br><br>Copeland at [0080] ("After the fragmentation information, an 8-bit time to live field specifies the remaining life of a packet and is decremented each time the packet is relayed. If this field is 0, the packet is destroyed. Next is an 8-bit protocol field that specifies the transport protocol used in the data portion. The following 16-bit field is a header checksum on the header only. Finally, the last two fields illustrated contain the 32-bit source address and 32-bit destination address. IP packet data follows the address information.")<br><br>Copeland at [0081] ("In a TCP/IP datagram 210, the initial data of the IP datagram is the TCP header 230 information. The initial TCP header 230 information includes the 16-bit source and 16-bit destination port numbers. A 32-bit sequence number for the data in the packet follows the port numbers. Following the sequence number is a 32-bit acknowledgement number. If an ACK flag (discussed below) is set, this number is the next sequence number the sender of the packet expects to receive. Next is a 4-bit data offset, which is the number of 32-bit words in the TCP header. A 6-bit reserved field follows.") |

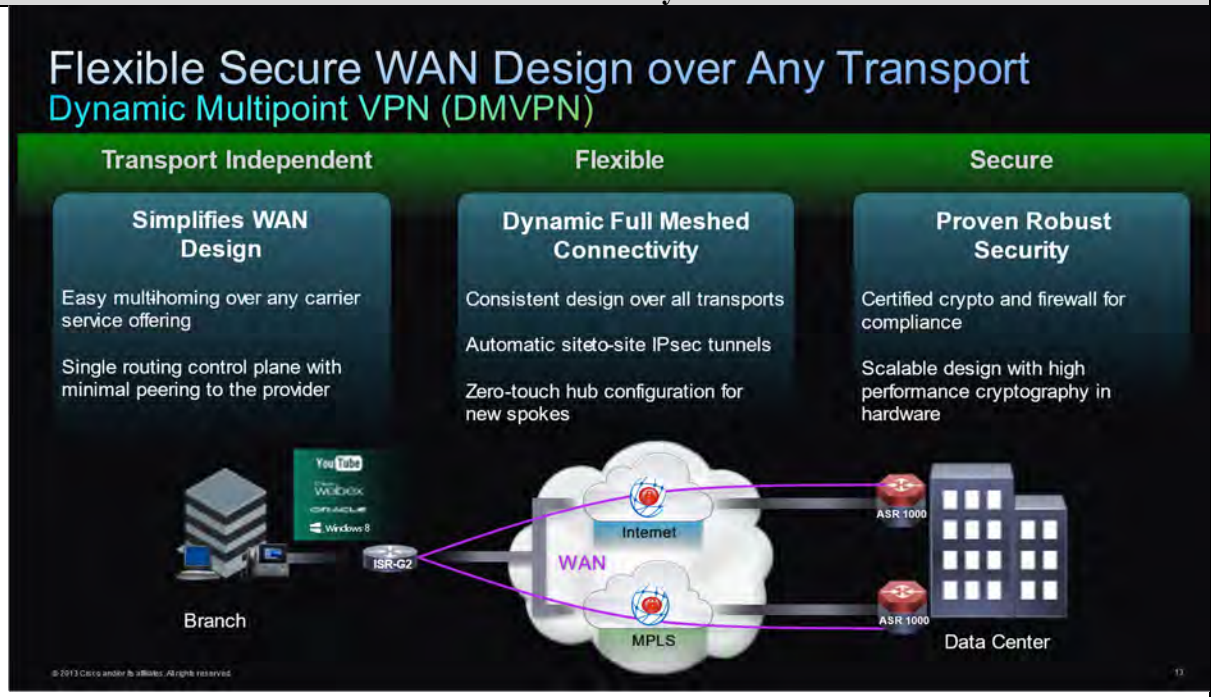
| No. | '111 Patent Claim 9 | Cisco IWAN System |
|---|---|---|
| 9 | The method according to claim 5, further comprising responsive to receiving the packet, analyzing the packet, by the controller. | Cisco IWAN System discloses the method according to claim 5, further comprising responsive to receiving the packet, analyzing the packet, by the controller.<br><br>For example, Cisco IWAN System discloses analyzing traffic flow metrics received by the hub/master controller to update and change traffic policy definitions. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Cisco IWAN System is found to not meet this limitation, |

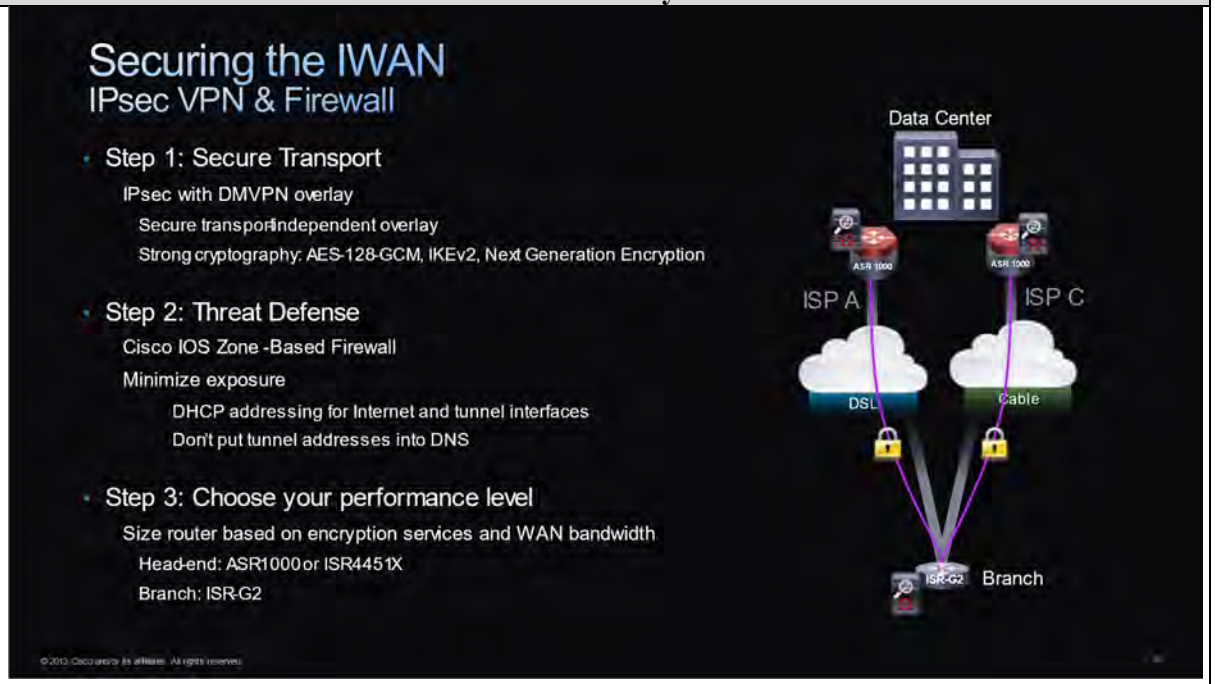


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1018 of 1100

| No. | '111 Patent Claim 9 | Cisco IWAN System |
|---|---|---|
| | | responsive to receiving the packet, analyzing the packet, by the controller would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>*See supra* at Claim 5.<br><br>Cisco Next Generation<br><br>  <br><br><br>Cisco IWAN |

| No. | '111 Patent Claim 9 | Cisco IWAN System |
|---|---|---|
| | |  <br>Cisco IWAN |

| No. | '111 Patent Claim 9 | Cisco IWAN System |
|---|---|---|
| | |  <br>Cisco IWAN |

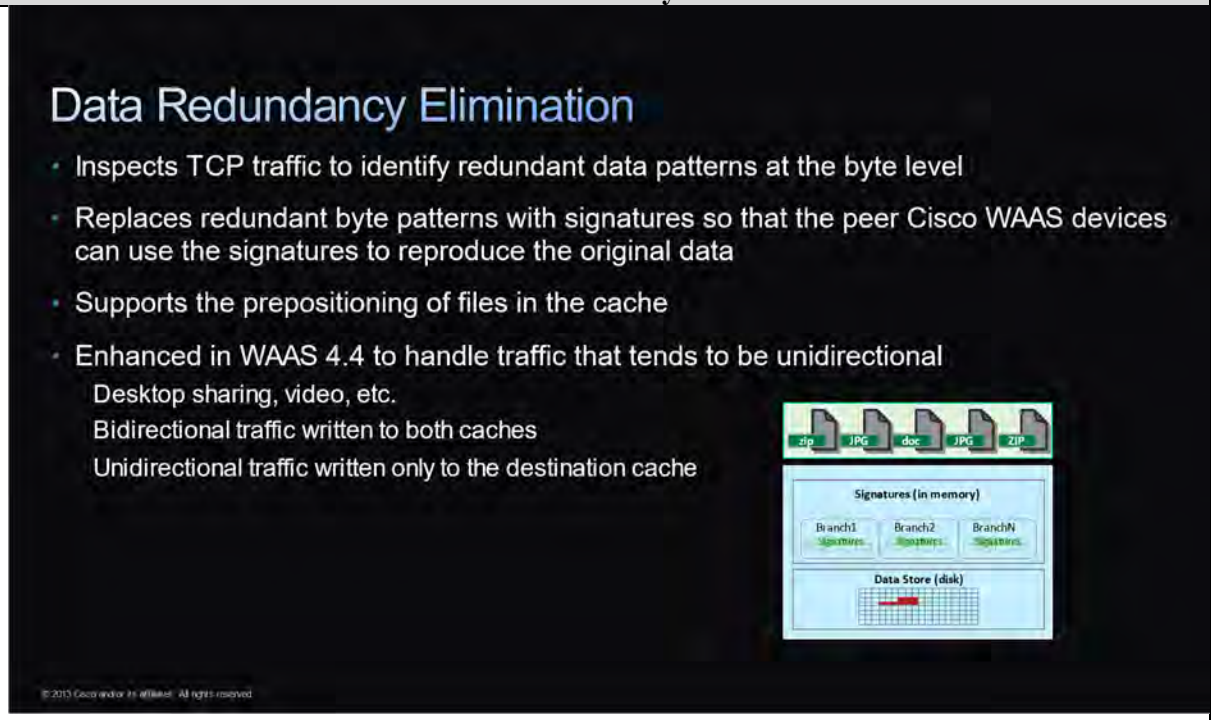
| No. | '111 Patent Claim 9 | Cisco IWAN System |
|---|---|---|
| | |  |

| No. | '111 Patent Claim 9 | Cisco IWAN System |
|---|---|---|
| | |  |

| No. | '111 Patent Claim 9 | Cisco IWAN System |
|---|---|---|
| | |  <br>Cisco IWAN - Uncompromised Experience |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1024 of 1100

| No. | '111 Patent Claim 9 | Cisco IWAN System |
|---|---|---|
| | |  <br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Cisco IWAN System in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 9 of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references.<br><br>For example, Swenson discloses the network controller comprising a flow analyzer for analyzing and inspecting the packet.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1025 of 1100

| No. | '111 Patent Claim 9 | Cisco IWAN System |
|---|---|---|
| | | inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0028] ("In contrast to conventional inline TCP throughput monitoring devices that monitor every single data packets transmitted and received, the network controller 140 is an "out-of-band" computer server that interfaces with the steer-ing device 130 to selectively inspect user flows of interest. The network controller 140 may further identify user flows (e.g., among the flows of interest) for optimization. In one embodiment, the network controller 140 may be imple-mented at the steering device 130 to monitor traffic. In other embodiments, the network controller 140 is coupled to and communicates with the steering device 130 for traffic moni-toring and optimization. When queried by the steering device 130, the network controller 140 determines if a given network flow should be ignored, monitored further or optimized. Opti-mization of a flow is often decided at the beginning of the flow because it is rarely possible to switch to optimized content mid-stream once non-optimized content delivery has begun. However, the network controller 140 may determine that existing flows associated with a particular subscriber or other entity should be optimized. In turn, new flows ( e.g., resulting from seek requests in media, new media requests, resume after pause, etc.) determined to be associated with the entity may be optimized. The network controller 140 uses the net-work state as well as historical traffic data in its decision for monitoring and optimization. Knowledge on the current net-work state, such as congestion, deems critical when it comes to data optimization.")<br><br>Swenson at [0029] ("As a flow is sent to the network controller 140 for inspection, historical network traffic data stored at the net-work controller 140 may be searched. The historical network traffic data includes information such as subscriber informa-tion, the cell towers to which the user devices attached, rout-ers through which the traffic is passing, geography regions, the backhaul segments, and time-of-day of the flows. For example, in a mobile network, the cell tower to which a user device is attached can be most useful, since it is the location where most congestion occurs due to limited bandwidth and high cost of the radio access network infrastructure. The network controller 140 looks into the historical traffic data for the average of the bandwidth per user at the particular cell tower. The |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1026 of 1100

| No. | '111 Patent Claim 9 | Cisco IWAN System |
|---|---|---|
| | | network controller 140 can then estimate the amount of bandwidth or degree of congestion for the new flow based on the historical record.")<br><br>Swenson at [0038] ("Turning back to FIG. 1, the network controller 140 allows network operators to apply fine granular optimization policies to ensure high quality of experience (QoE) based on cell tower congestion, device types, subscriber profiles and service plans with lower hardware and software costs. The architecture of the network controller 140 provides an excel-lent fit for the net neutrality guideline of "reasonable network management", and better compliance to the copyright law (DMCA) than solutions that rely on long-term caching. Hav-ing the ability of monitoring network traffic on a per sub-scriber, per flow, or per video file basis, the network controller 140 also selectively monitors and optimizes only a subset of traffic that benefits from optimization the most, thus achiev-ing both scalability and efficiency for optimization at a com-petitive price-point. The core element of the network control-ler 140 lies in its mechanisms for congestion detection and mitigation, which allows optimization resources to be utilized in the most efficient and surgical manner.")<br><br>Swenson at [0039] ("Referring now to FIG. 3, it illustrates one embodi-ment of an example architecture of the network controller 140 for providing selective real-time network monitoring and subscriber identification. The network controller 140 com-prises a flow analyzer 312, a policy engine 314, a steering device interface 316, a video optimizer redirector 318, a flow cache 322, and a subscriber log 324. In other embodiments, the network controller 140 may include additional, fewer, or different components for various applications. Conventional components such as network interfaces, security functions, failover servers, management and network operations con-soles, and the like are not shown so as to not obscure the details of the system architecture.")<br><br>Swenson at [0059] ("In one embodiment, as the steering device 130 monitors network responses, it is looking for flows that match one or more signatures for video and images. When a match-ing flow is detected, the steering device 130 forwards the HTTP request and a portion of the HTTP response to the network controller 140 over the ICAP client interface 404. After receiving the request and the portion of response at the ICAP server interface 406, the flow analyzer 312 of the net-work controller 140 performs a deep flow inspection to deter-mine if the flow is worth bandwidth monitoring and/or user detection. For example, |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1027 of 1100

| No. | '111 Patent Claim 9 | Cisco IWAN System |
|---|---|---|
| | | the flow inspection performed by the flow analyzer 312 may determine if the flow indeed contains large or medium object ( e.g., larger than 50 kB), and/or if the source IP address of the flow is from a user or a group of users that are required to be monitored by policies. The flow ana-lyzer 312 may also determine if the flow needs to be opti-mized based on historical flow statistical data.")<br><br>Swenson at [0060] ("If the flow is deemed of interest, the steering device 130 is notified to steer the flow through the network controller 140. This is known as the "continue" working mode for bandwidth monitoring. In the "continue" mode, the network controller 140 interfaces with the steering device 130 to func-tion, on-demand, as a traditional inline network element for flows deemed of interest. Thus, the network controller 140 ingests the network flow for inspection and subsequently forwards the network flow on the network response path. For example, for this particular flow, the origin server 160 responds to the user request by sending video or images over the network link 413 to the steering device 130, which for-wards the video or images to the network controller 140 over a network link 414. After the network controller 140 updates the flow statistics, the video or images are returned to the steering device 130 over a network link 415, which transmits the video or images to the user device 110 over the network link 416.")<br><br>Swenson at [0061] ("Once a flow is reported to the network controller 140, a flow cache entry is created for the flow in the flow cache 322. The flow cache entry keeps track of the flow and its associated bandwidth. For a flow that is marked in "continue" mode, each time the steering device 130 forwards a next portion of the flow payload to the network controller 140, the flow cache 3 22 updates the number of bytes for transmitted in the flow. By monitoring the number of bytes per flow over time, the flow analyzer 312 is capable of determining an estimate value of bandwidth associated with flow. Further-more, since the steering device 130 does not have infinite packet buffers, if congestion happens on the network link 416 from the steering device 130 to the user device 110, the TCP congestion control mechanism kicks in at the steering device 130, which may slows down and/or eventually stop receiving data over the network link 413 from origin server 160. During the congestion, the steering device 130 would not forward any data to the network controller 140, since the link 416 is congested and the network controller 140 would not be able to transmit data to the user device 110. Therefore, as an inline element, the network controller 140 can detect network con-gestions and estimate bandwidth associated with any flows of |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1028 of 1100

| No. | '111 Patent Claim 9 | Cisco IWAN System |
|---|---|---|
| | | interest selected by the network controller 140. However, in the "continue" mode, the network controller 140 does not modify and transform the HTTP messaged it receives over the ICAP interface. The network controller 140 simply updates the flow statistics and returns the video or images to the steering device 130 for transmission to the user device 110.")<br><br>Swenson at [0065] ("FIG. 5 is a block diagram illustrating an example event trace of "continue" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 512 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 512 to the intended origin server 160 and receives a response 514 back from the origin server 160. The steering device 130 then sends an ICAP request message 516 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 518. Upon receiving the instruc-tion, the steering device 130 re-writes the response 514 to an HTTP redirect response 520, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 520 directly to the user device 110. In case the flow dose not contain video or image objects, or the network controller 140 determines not to monitor the flow, the steering device 13 0 would forward the response to the user device 110.")<br><br>Swenson at [0069] ("FIG. 6 is a block diagram illustrating an example event trace of "counting" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 612 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 612 to the intended origin server 160 and receives a response 614 back from the origin server 160. The steering device 130 then sends an ICAP request message 616 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this |

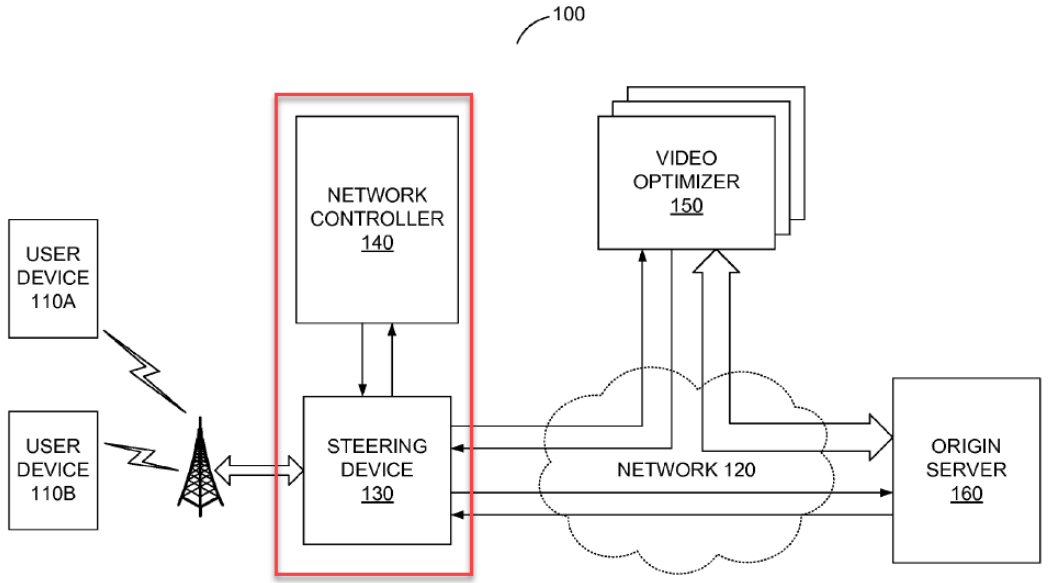


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1029 of 1100

| No. | '111 Patent Claim 9 | Cisco IWAN System |
|---|---|---|
| | | case, the network controller 140 responds with a redirect to optimize the video in ICAP response 618. Upon receiving the instruc-tion, the steering device 130 re-writes the response 614 to an HTTP redirect response 620, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 620 directly to the user device 110. In case the flow dose not contain video or image objects that need to be redirected, the steering device 130 would forward the response to the user device 110.")<br><br>Swenson at [0071] ("After receiving the request, the video optimizer 150 forwards the video HTTP GET requests 622 to the origin server 160 and in return, receives a video file 624 from the origin server 160. The video optimizer 150 transcodes the video file to a format usable by the client device 110 based on network bandwidth available to the user device 110. The optimized video 626 is then transmitted from the video opti-mizer 150 to the steering device 130. In one embodiment, the steering device 130 intercepts the optimized video 626. The steering device 130 will then send an ICAP request to the network controller 140 for inspection. The network controller 140 deems this flow to be monitored and sends ICAP response 630. The steering device 130 then allows the flow to go through to the user device 110. The steering device 130 next sends periodic ICAP "counting" updates 632 to the network controller 140 until the flow completes. As such, the client receives the optimized video 626 for substantially real-time playback on an application executing on the user device 110.")<br><br>Swenson at Figure 1 (annotation added) |

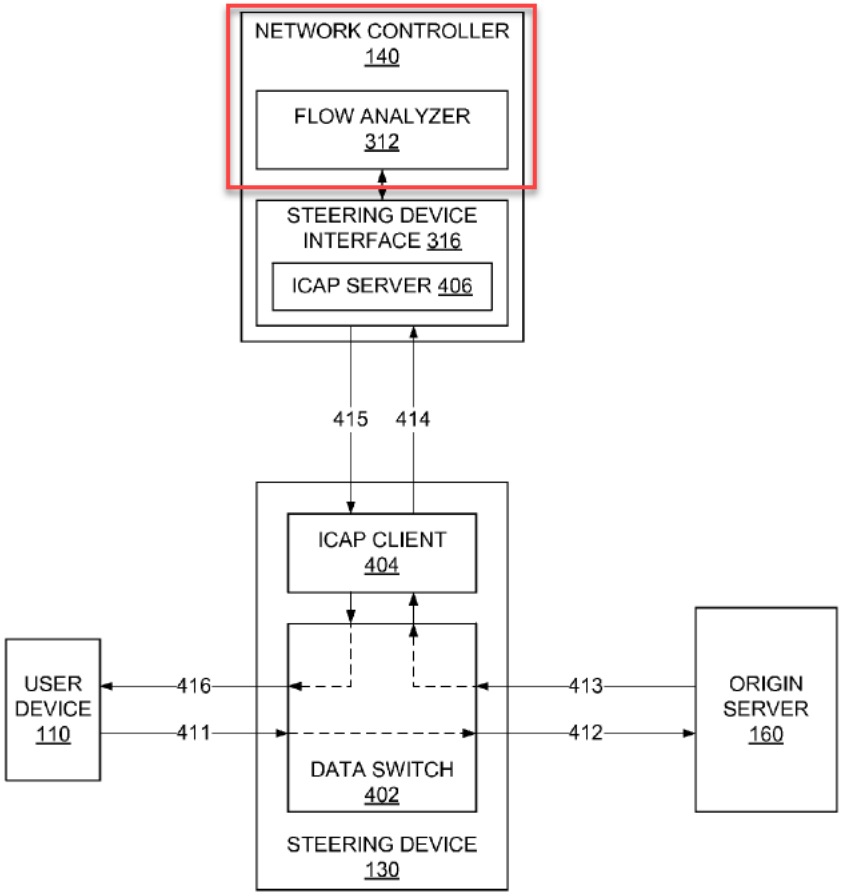


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1030 of 1100

| No. | '111 Patent Claim 9 | Cisco IWAN System |
|---|---|---|
| | |  <br>FIG. 1<br>Swenson at Figure 4A (annotation added) |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1031 of 1100

| No. | '111 Patent Claim 9 | Cisco IWAN System |
|---|---|---|
| | |  <br>FIG. 4A<br><br>For example, Copeland discloses analyzing packets received by the intrusion detection engine on the monitoring appliance. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1032 of 1100

| No. | '111 Patent Claim 9 | Cisco IWAN System |
|---|---|---|
| | | Copeland at [0021] ("The present invention provides an accurate and reliable method for detecting network attacks through the use of sampled packet headers that are provided by a source such as that as defined in sFlow and further based in large part on "flows" as opposed to signatures or anomalies. By utilizing the host and flow information structures that are inherent with flow-based analysis and applying rules of statistical significance and analysis, the intrusion detection system can operate with sampled data such as provided by sFlow in order to provide a hybrid solution that combines many of the benefits of a packet capture implementation with the distributed nature of an IDS that operates on Netflow, thus providing an enhanced wide-area IDS solu-tion.")<br><br>Copeland at [0023] ("According to one aspect of the invention, the detection system works by assigning sampled data packets to various client/server ( C/S) flows. Statistics are collected for each determined flow. Then, the flow statistics are analyzed to determine if the flow appears to be legitimate traffic or possible suspicious activity. A value, referred to as a "concern index," is assigned to each flow that appears suspicious. By assigning a value to each flow that appears suspicious and adding that value to an accumulated concern index associated with the responsible host, it is possible to identify hosts that are engaged in intruder activity without generation of significant unwarranted false alarms. When the concern index value of a host exceeds a preset alarm value, an alert is issued and appropriate action can be taken.")<br><br>Copeland at [0024] ("Generally speaking, the intrusion detection system analyzes network communication traffic for potential detrimental activity. The system collects flow data from sampled packet headers between two hosts or Internet Protocol (IP) addresses. Collecting flow data from packet headers asso-ciated with a single service where at least one port remains constant allows for more efficient analysis of the flow data. The collected flow data is analyzed to assign a concern index value to the flow based upon a probability that the flow was not normal for data communications. A host list is main-tained containing an accumulated concern index derived from the flows associated with the host. Once the accumu-lated concern index has exceeded an alarm threshold value, an alarm signal is generated.")<br><br>Copeland at [0027] ("According to one aspect of the invention, the detection system works by assigning sampled data packets to various client/server ( C/S) flows. Statistics are |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1033 of 1100

| No. | '111 Patent Claim 9 | Cisco IWAN System |
|---|---|---|
| | | collected for each determined flow. Then, the flow statistics are analyzed to determine if the flow appears to be legitimate traffic or possible suspicious activity. A value, referred to as a "concern index," is assigned to each flow that appears suspicious. By assigning a value to each flow that appears suspicious and adding that value to an accumulated concern index associated with the responsible host, it is possible to identify hosts that are engaged in intruder activity without generation of significant unwarranted false alarms. When the concern index value of a host exceeds a preset alarm value, an alert is issued and appropriate action can be taken.")<br><br>Copeland at [0028] ("Generally speaking, the intrusion detection system analyzes network communication traffic for potential detri-mental activity. The system collects flow data from sampled packet headers between two hosts or Internet Protocol (IP) addresses. Collecting flow data from packet headers asso-ciated with a single service where at least one port remains constant allows for more efficient analysis of the flow data. The collected flow data is analyzed to assign a concern index value to the flow based upon a probability that the flow was not normal for data communications. A host list is main-tained containing an accumulated concern index derived from the flows associated with the host. Once the accumu-lated concern index has exceeded an alarm threshold value, an alarm signal is generated.")<br><br>Copeland at [0063] ("Consequently, abnormal flows and/or events iden-tified by the intrusion detection engine 155 will raise the concern index (CI) for the associated host. The intrusion detection engine 155 analyzes the data flow between IP devices. However, different types of services have different flow characteristics associated with that service. Therefore, a C/S flow can be determined by the packets exchanged between the two hosts dealing with the same service.")<br><br>Copeland at [0065] ("The intrusion detection engine 155 analyzes the flow data 160 to determine if the flow appears to be legitimate traffic or possible suspicious activity. Flows with suspicious activity are assigned a predetermined concern index (CI) value based upon a heuristically predetermined assessment of the significance of the threat of the particular traffic or flow or suspicious activity. The flow concern index values have been derived heuristically from extensive net-work traffic analysis. Concern index values are associated with particular hosts and stored in the host data structure 166 (FIG. 1). Exemplary concern |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1034 of 1100

| No. | '111 Patent Claim 9 | Cisco IWAN System |
|---|---|---|
| | | index values for various exemplary flow-based events and other types of events are illustrated in connection with FIGS. 6 and 7.)<br><br>Copeland at [0069] ("It will now be appreciated that the disclosed meth-odology of intrusion detection is accomplished at least in part by analyzing communication flows to determine if such communications have the flow characteristics of probes or attacks. By analyzing communications for abnormal flow characteristics, attacks can be determined without the need for resource-intensive packet data analysis. A flow can be determined from the packets 101 that are transmitted between two hosts utilizing a single service. The addresses and port numbers of communications are easily discerned by analysis of the header information in a datagram.")<br><br>Copeland at [0087] ("As previously stated, the flow-based engine 155 does not analyze the data segments of packets for signature identification. Instead, the engine 155 associates all packets with a flow. It analyzes certain statistical data and assigns a concern index value to abnormal activity. The engine 155 builds a concern index for suspicious hosts by detecting suspicious activities on the network. An alarm is generated when those hosts build enough concern (in the form of a cumulated CI value) to cross the network administrator's predetermined threshold.")<br><br>Copeland at [0097] ("The described TCP session 300 of FIG. 3 is a generic TCP session in which a network might engage. In accordance with the invention, flow data is collected about the session to help determine if the communication is abnormal. In the preferred embodiment, information such as the total number of packets sent, the total amount of data sent, the session start time and duration, and the TCP flags set in all of the packets, are collected, stored in the database 160, and analyzed to determine if the communication was suspicious. If a communication is deemed suspicious, i.e. it meets predetermined criteria, a predetermined concern index value associated with a determined category of suspicious activity is added to the cumulated CI value associated with the host that made the communication.")<br><br>Copeland at [0111] ("As shown, the packets exchanged between two hosts associated with a single service can determine a flow. A port number designates a service application that is associated with the particular port. Communications utiliz-ing differing protocols or services |



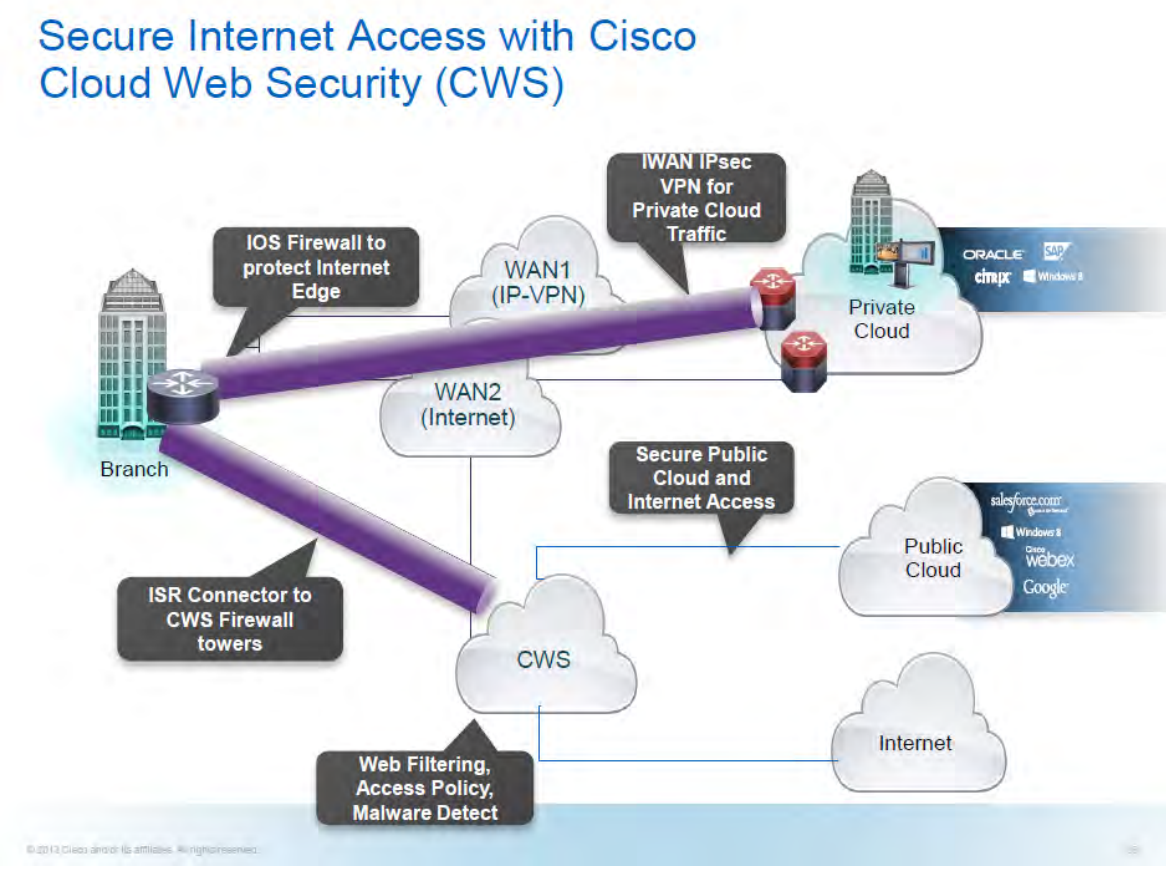
Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1035 of 1100

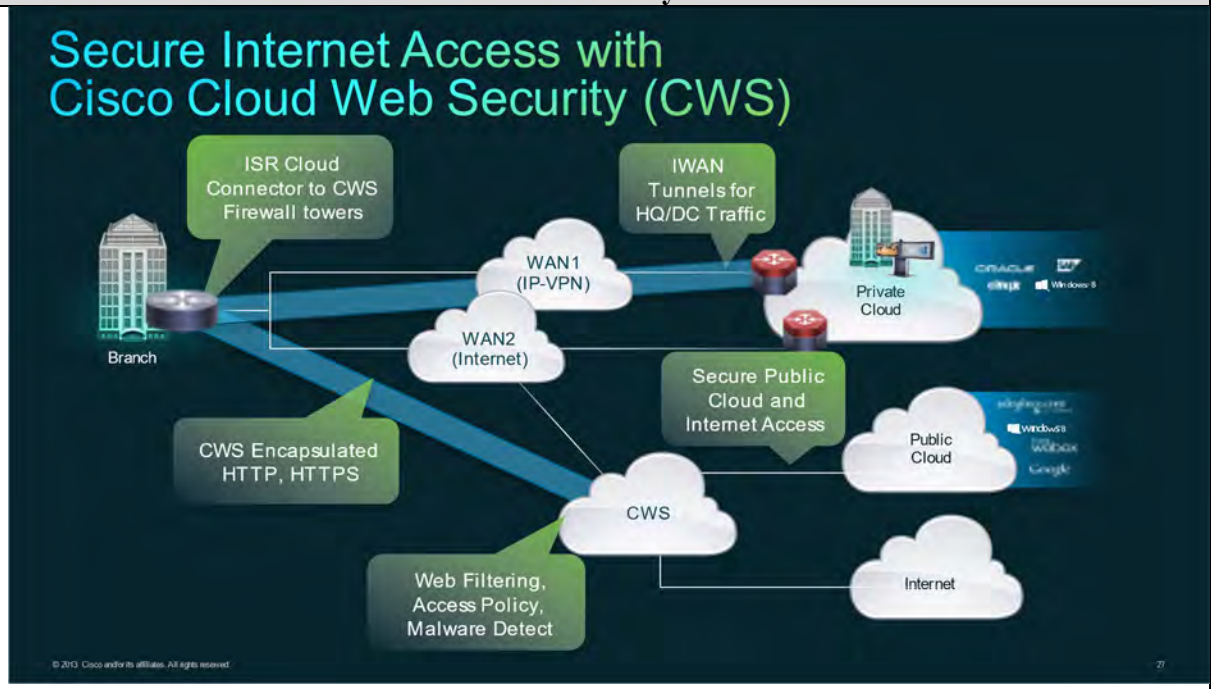
| No. | '111 Patent Claim 9 | Cisco IWAN System |
|---|---|---|
| | | provide differing flow characteristics. Consequently, the flow engine 155 analyzes each of the services separately.")<br><br>Copeland at [0150] ("A preferred hardware configuration 800 of an embodiment that executes the functions of the above-described flow-based engine is described in reference to FIG. 8. FIG. 8 illustrates a typically hardware configuration 800 for a network intrusion detection system. A monitoring appliance 150 serves as a pass-by filter of network traffic. A network device 135, such as a router or switch supporting sFlow provides the location for connecting the monitoring appliance 150 to the network 899 for monitoring the network traffic.") |

| No. | '111 Patent Claim 12 | Cisco IWAN System |
|---|---|---|
| 12 | The method according to claim 9, wherein the analyzing comprises applying security or data analytic application. | Cisco IWAN System discloses the method according to claim 9, wherein the analyzing comprises applying security or data analytic application.<br><br>For example, Cisco IWAN System discloses analyzing traffic flow metrics with secure transport and application performance monitoring. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met.<br><br>*See supra* at Claim 9.<br><br>IWAN Next Gen at 33 |



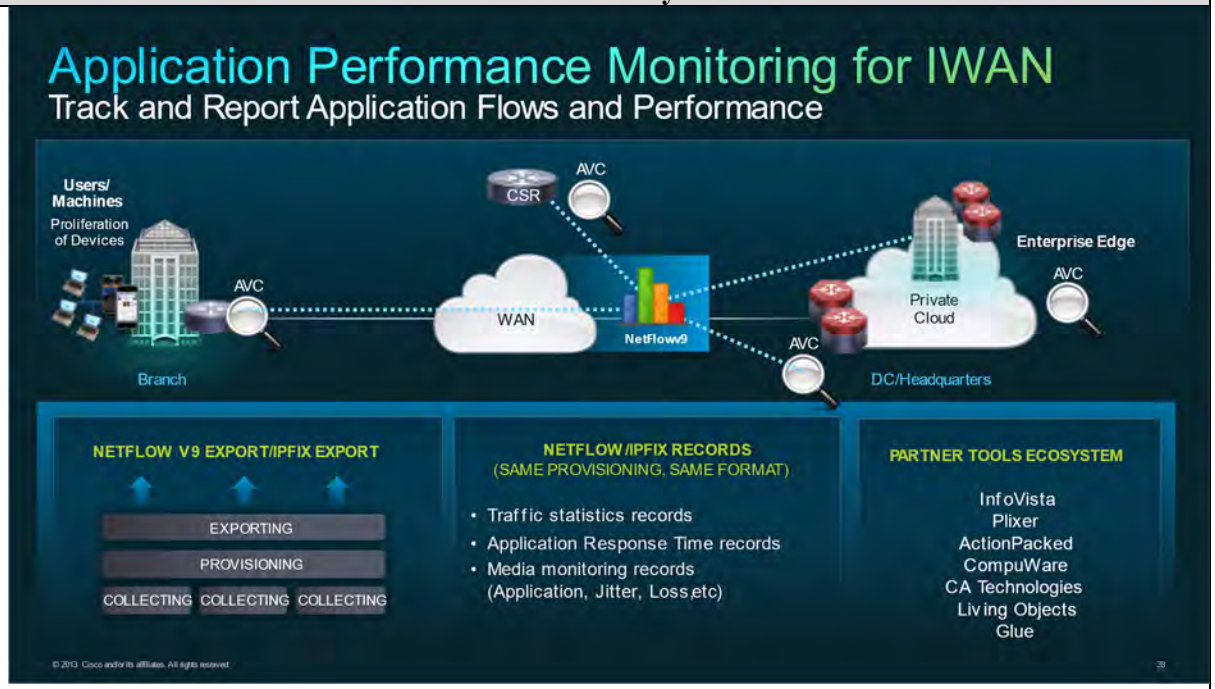
Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1036 of 1100

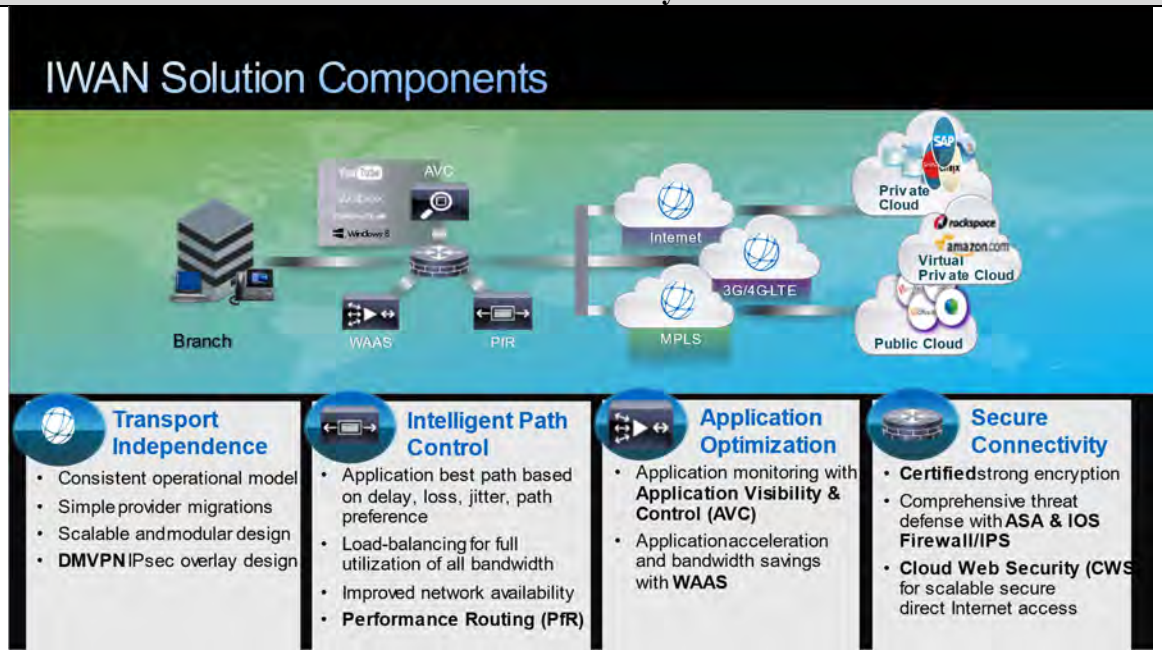
| No. | '111 Patent Claim 12 | Cisco IWAN System |
|---|---|---|
| | |  <br>Cisco Next Generation |

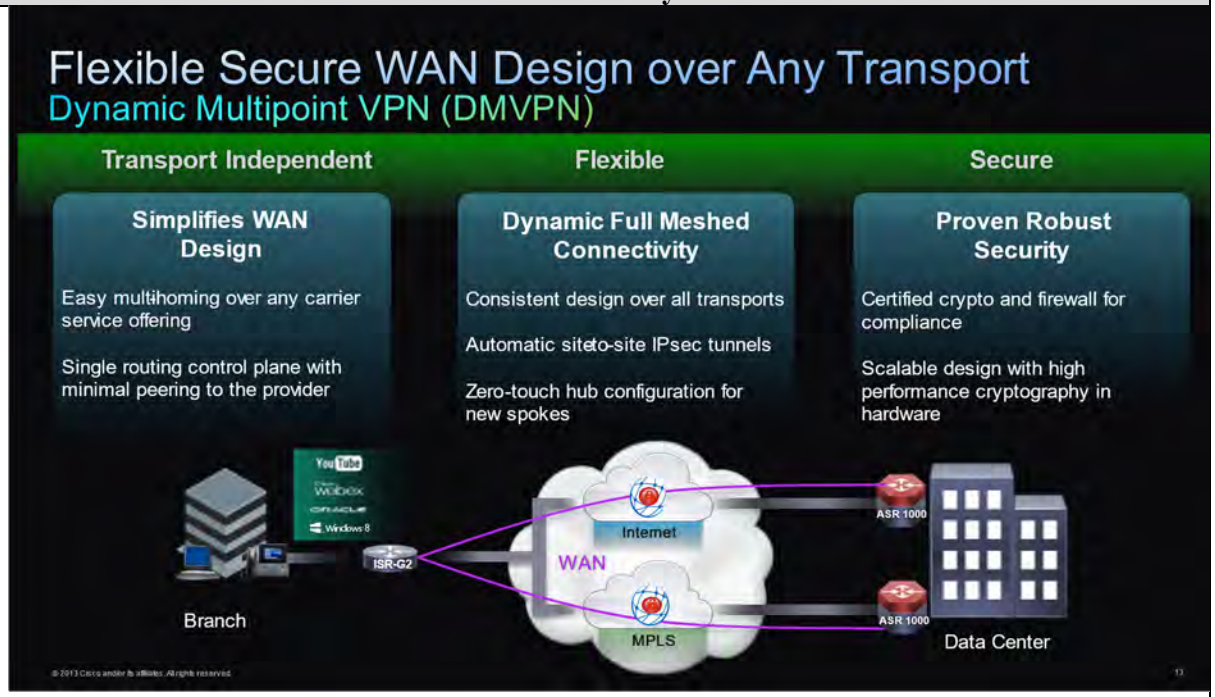
| No. | '111 Patent Claim 12 | Cisco IWAN System |
|---|---|---|
| | |  <br>Cisco Next Generation |

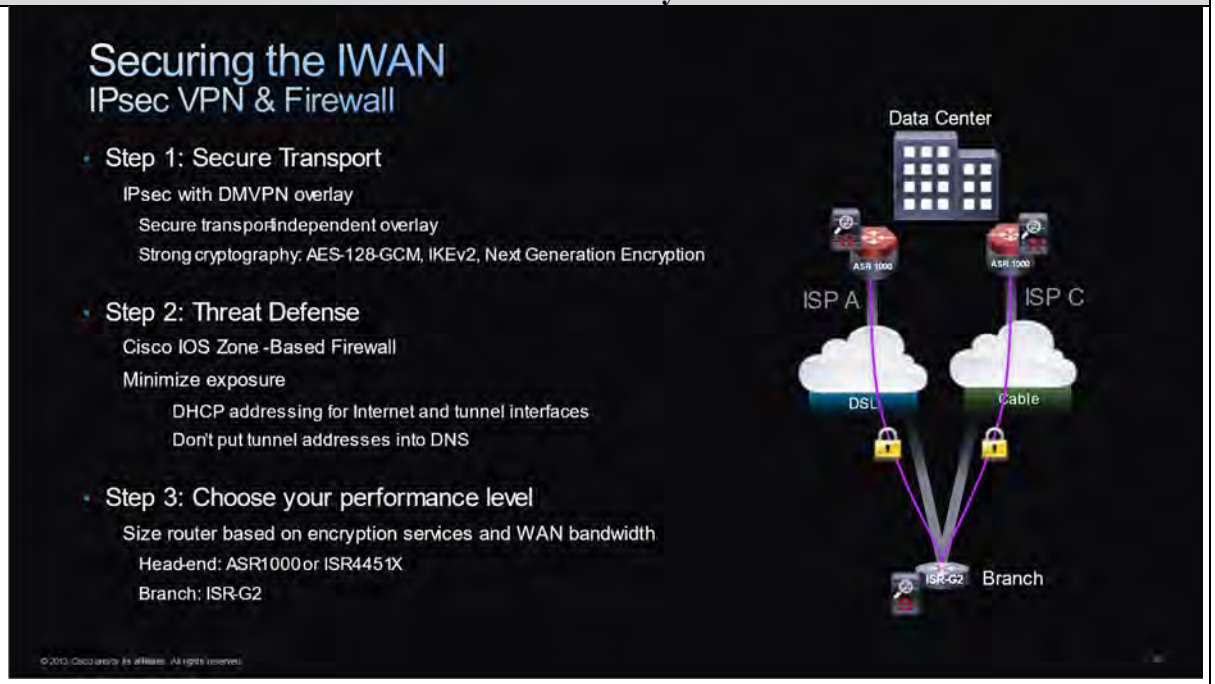


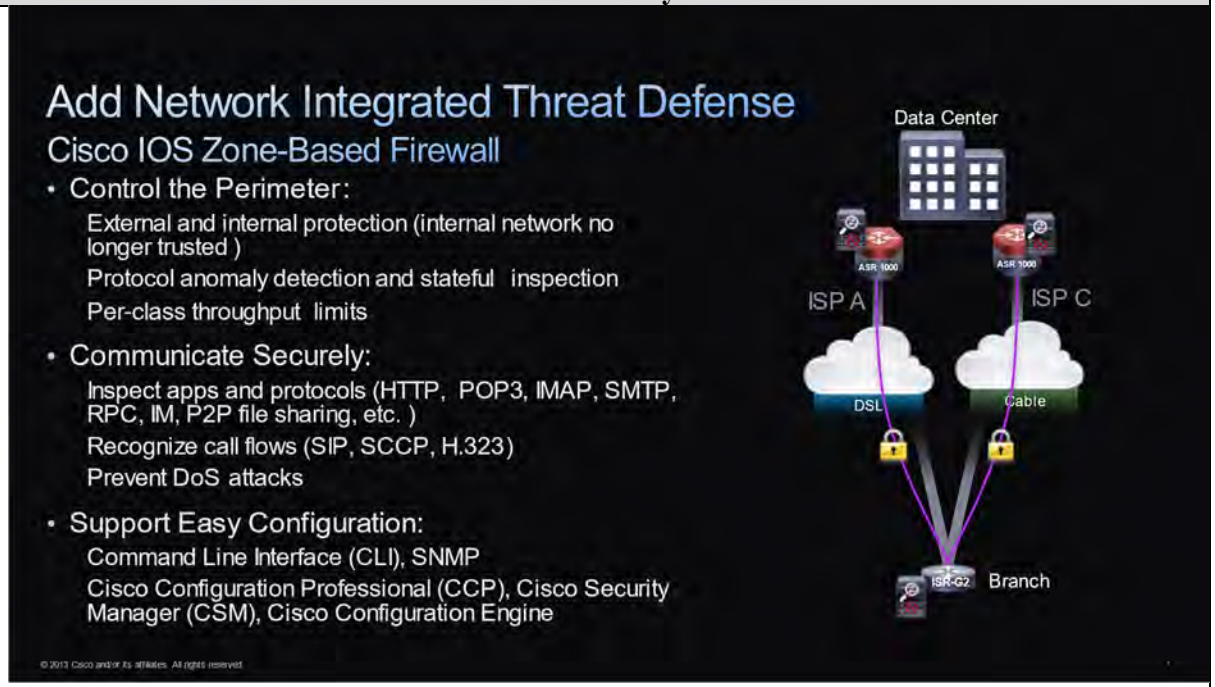
Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1038 of 1100

| No. | '111 Patent Claim 12 | Cisco IWAN System |
|---|---|---|
| | |  <br>Cisco IWAN |

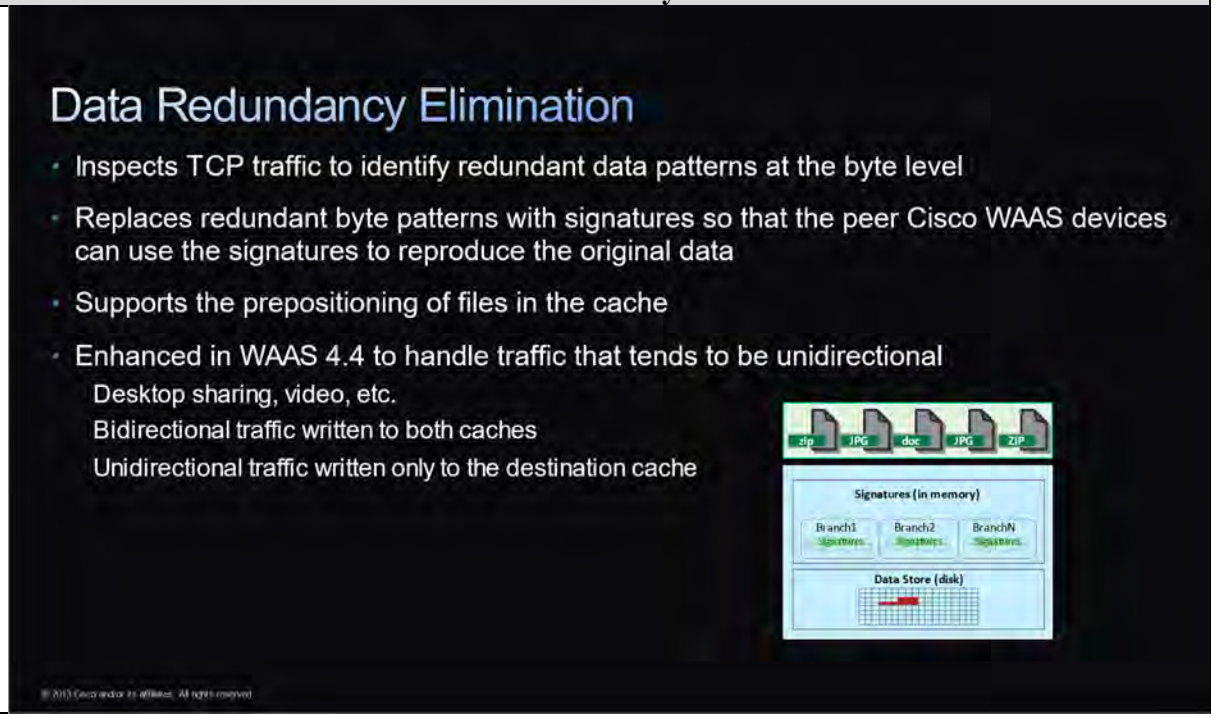
| No. | '111 Patent Claim 12 | Cisco IWAN System |
|---|---|---|
| | |  <br>Cisco IWAN |

| No. | '111 Patent Claim 12 | Cisco IWAN System |
|---|---|---|
| | |  <br>Cisco IWAN |

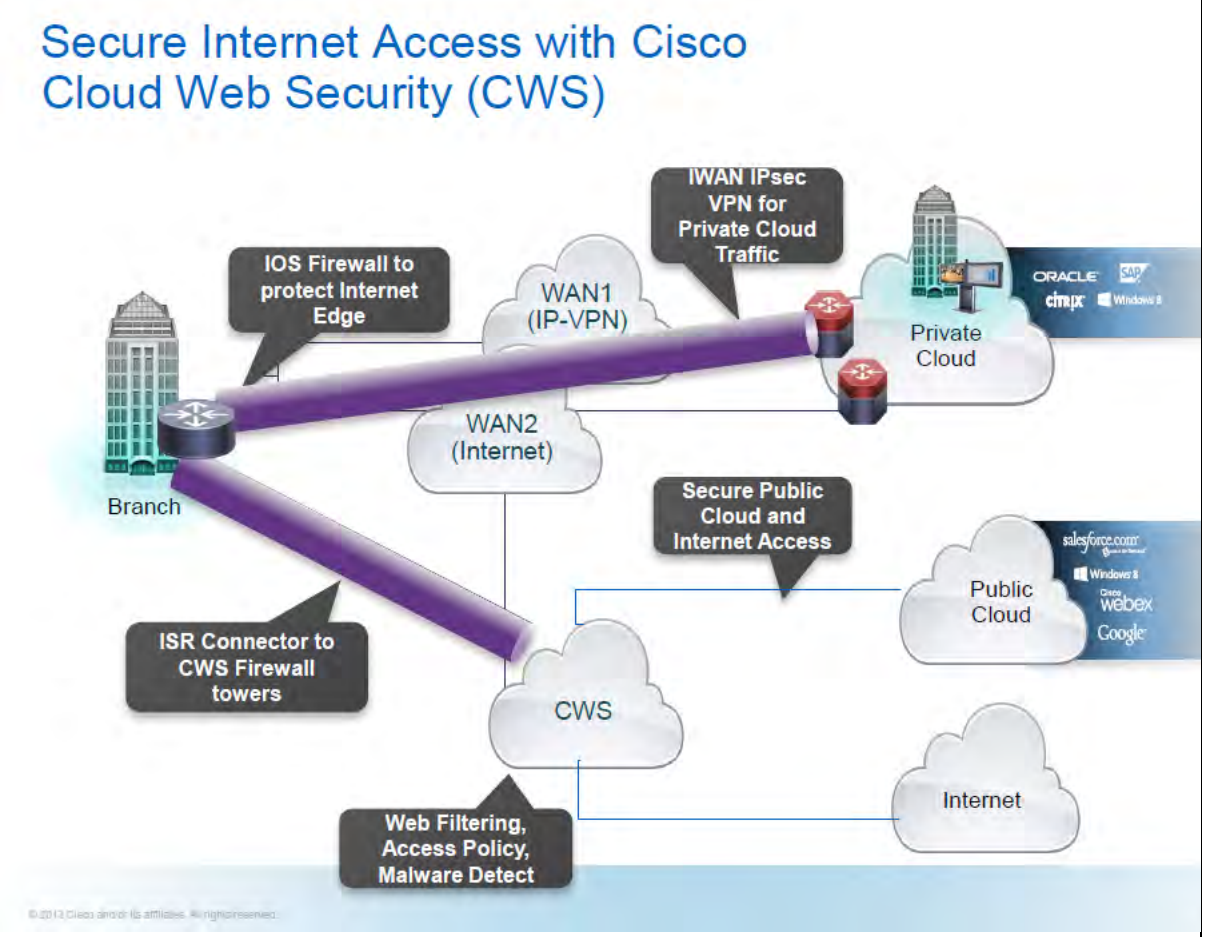
| No. | '111 Patent Claim 12 | Cisco IWAN System |
|---|---|---|
| | |  |

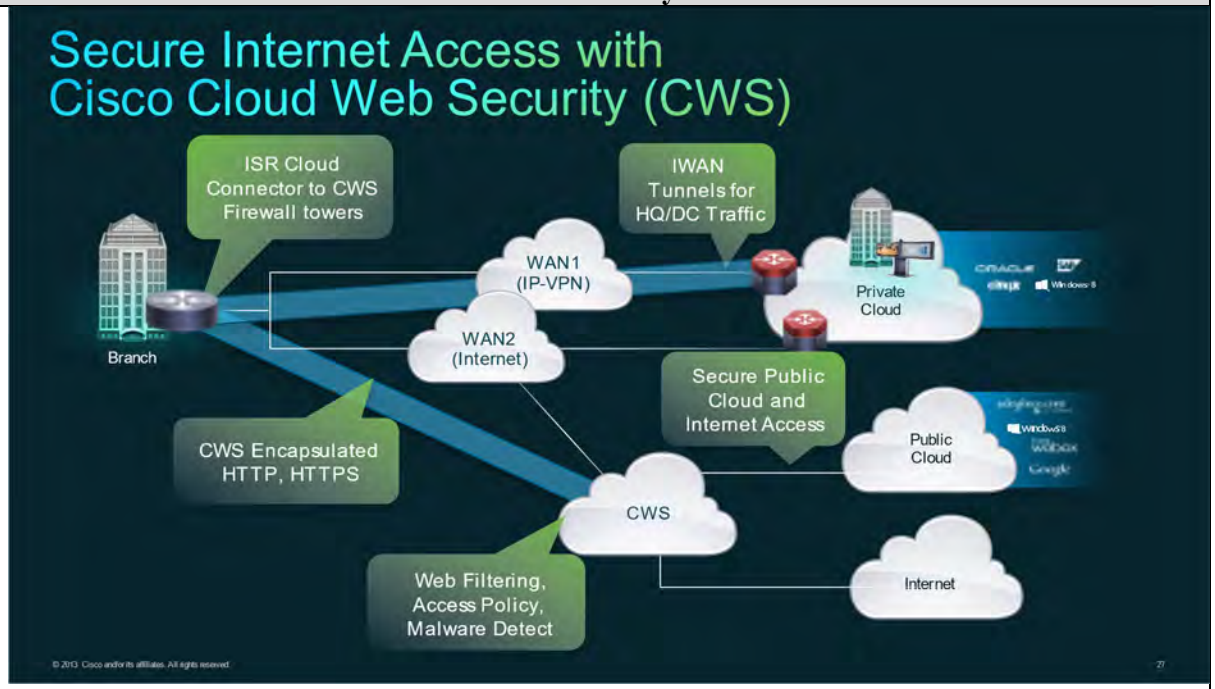
| No. | '111 Patent Claim 12 | Cisco IWAN System |
|---|---|---|
| | |  |

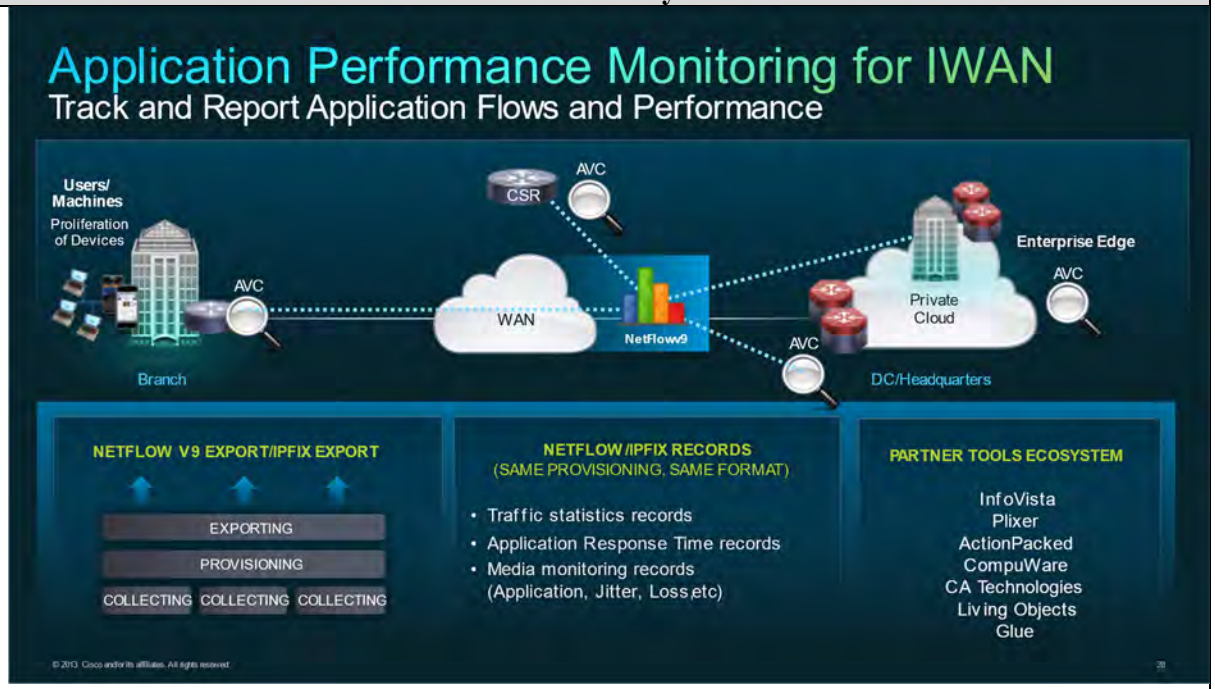
| No. | '111 Patent Claim 12 | Cisco IWAN System |
|---|---|---|
| | |  <br>Cisco IWAN - Uncompromised Experience |

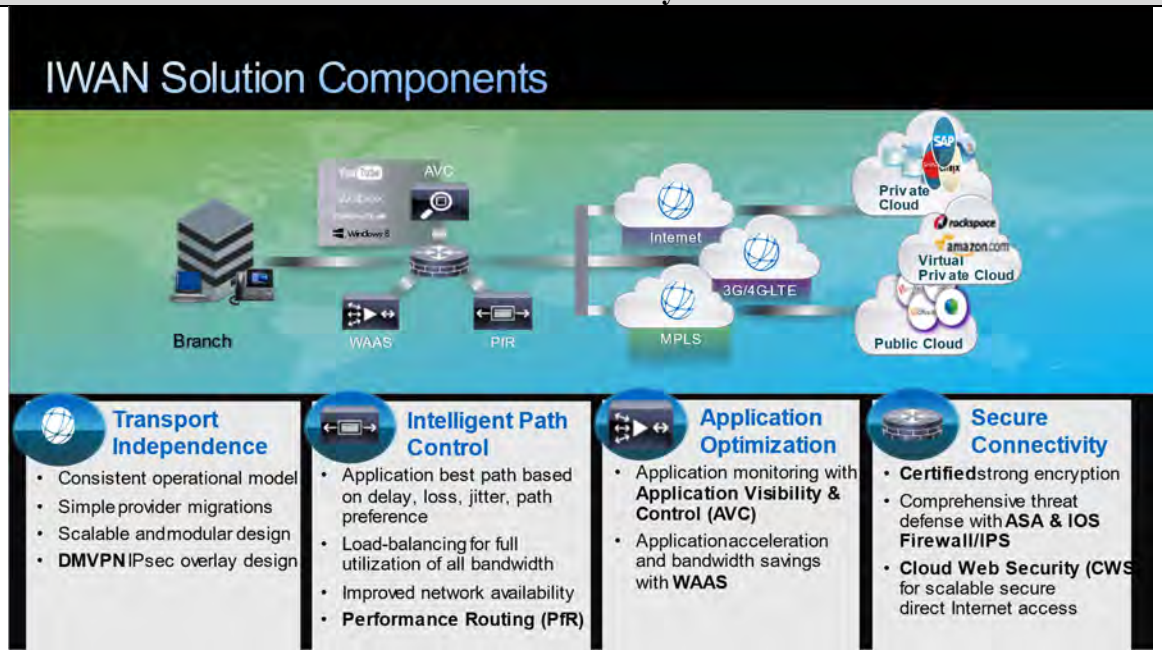
| No. | '111 Patent Claim 12 | Cisco IWAN System |
|---|---|---|
| | |  |

| No. | '111 Patent Claim 13 | Cisco IWAN System |
|---|---|---|
| 13 | The method according to claim 9, wherein the analyzing comprises applying security application that comprises firewall or intrusion detection functionality. | Cisco IWAN System discloses the method according to claim 9, wherein the analyzing comprises applying security application that comprises firewall or intrusion detection functionality.<br><br>For example, Cisco IWAN System discloses analyzing traffic flow metrics with zone based firewall and integrated threat defense functionality. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met.<br><br>*See supra* at Claim 9.<br><br>IWAN Next Gen at 33 |

| No. | '111 Patent Claim 13 | Cisco IWAN System |
|---|---|---|
| | |  <br>Cisco Next Generation |

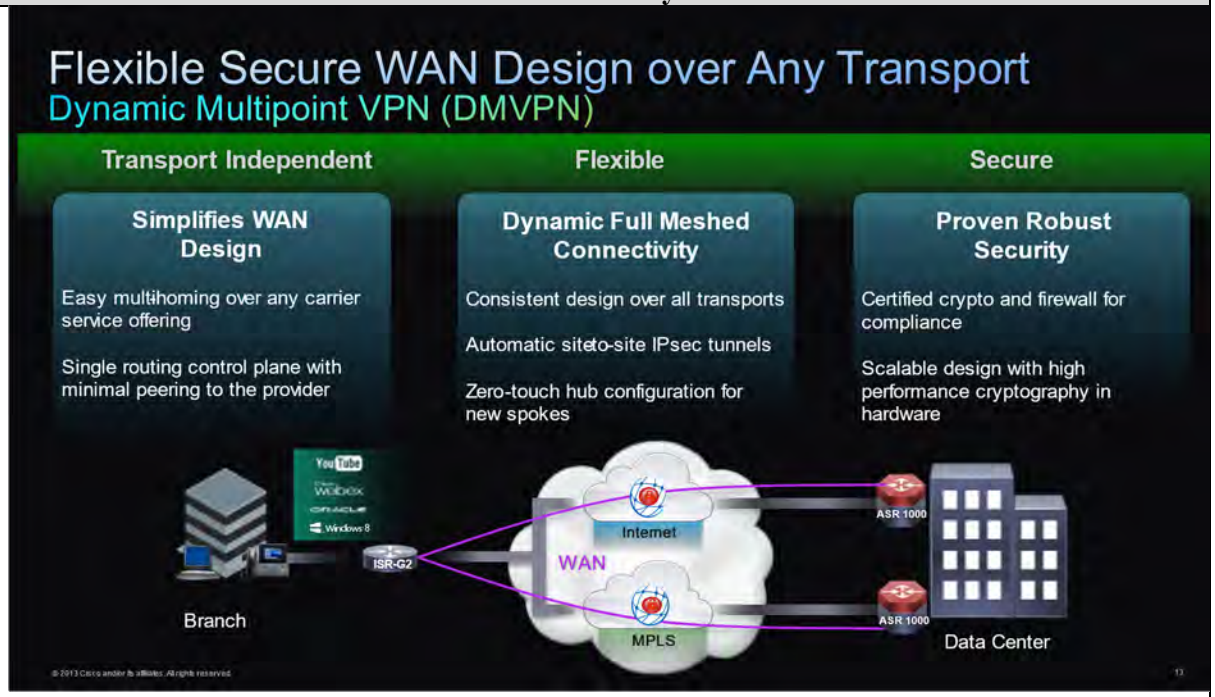
| No. | '111 Patent Claim 13 | Cisco IWAN System |
|---|---|---|
| | |  <br>Cisco Next Generation |

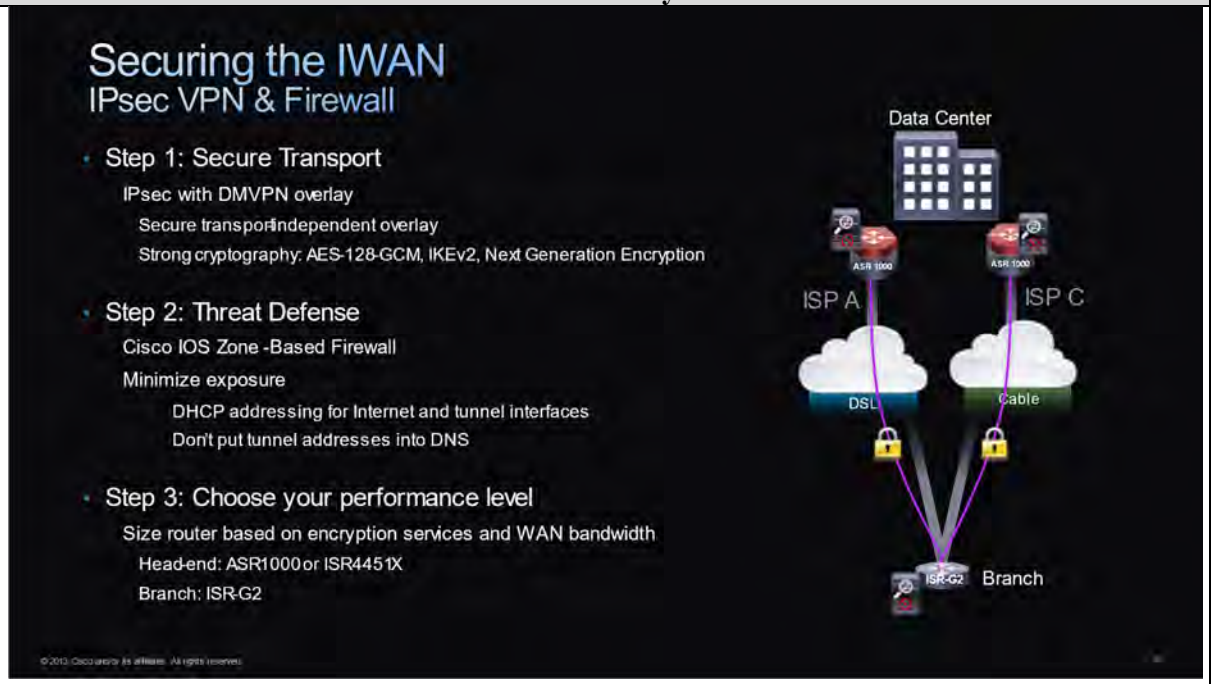
| No. | '111 Patent Claim 13 | Cisco IWAN System |
|---|---|---|
| | |  <br>Cisco IWAN |

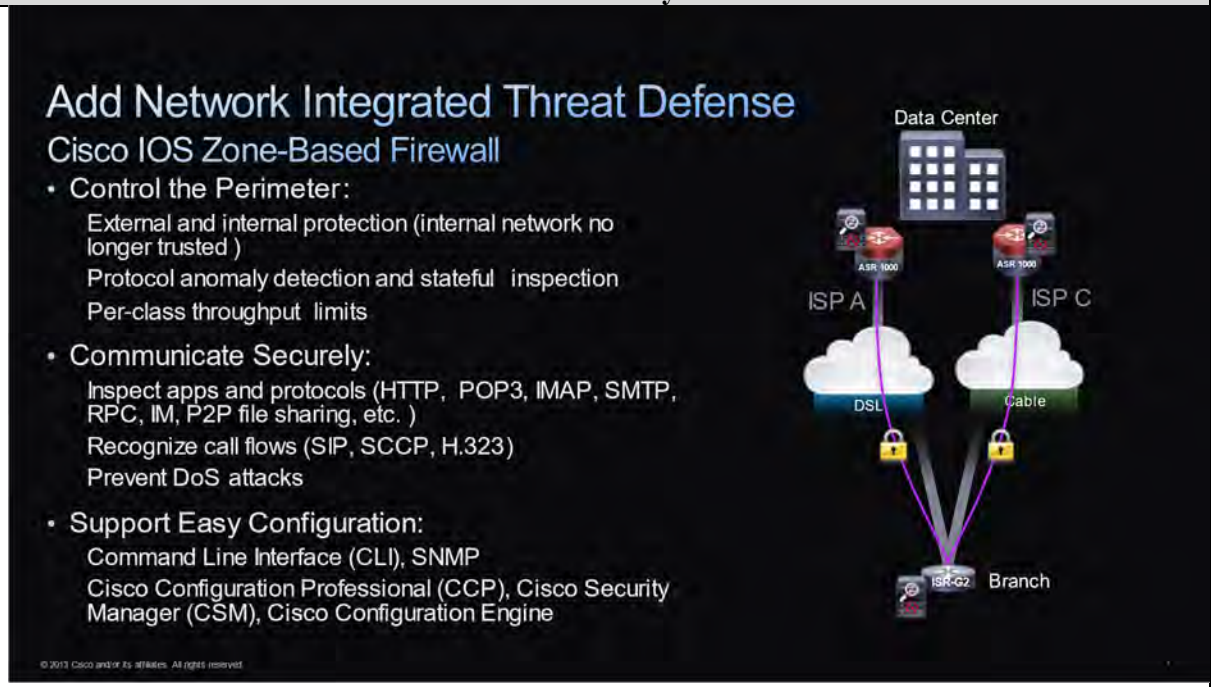
| No. | '111 Patent Claim 13 | Cisco IWAN System |
|---|---|---|
| | |  <br>Cisco IWAN |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1049 of 1100

| No. | '111 Patent Claim 13 | Cisco IWAN System |
|---|---|---|
| | |  <br>Cisco IWAN |

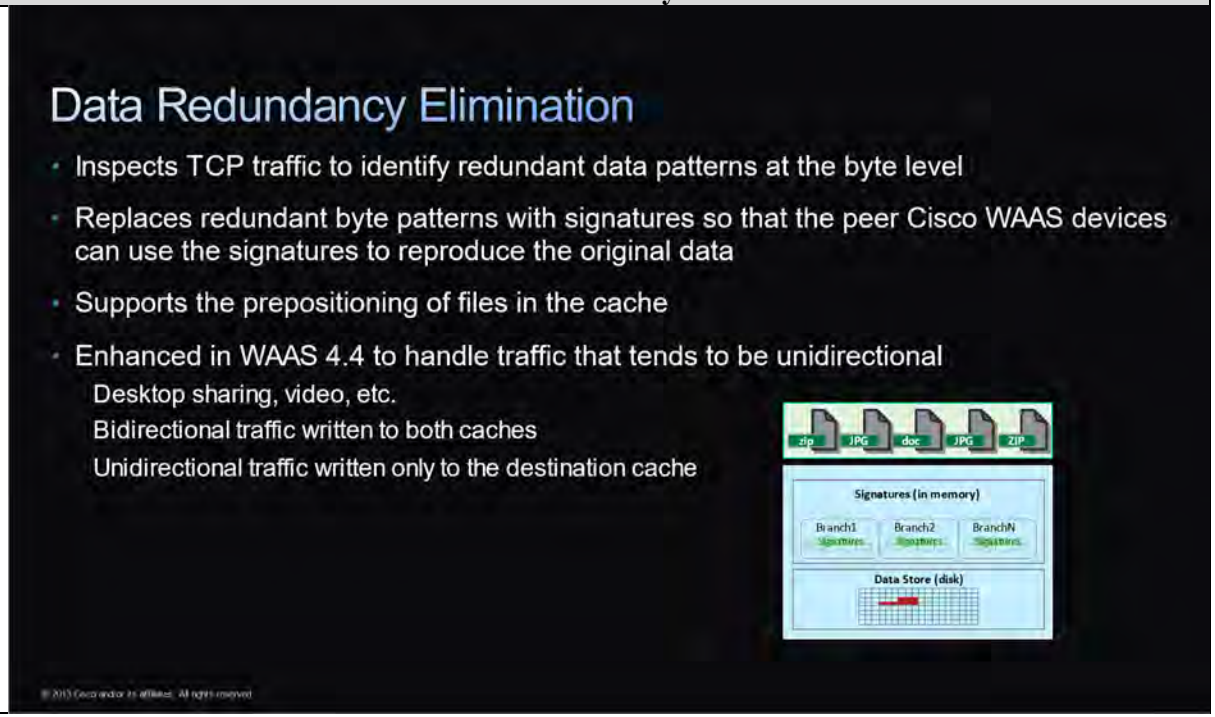
| No. | '111 Patent Claim 13 | Cisco IWAN System |
|---|---|---|
| | |  |

| No. | '111 Patent Claim 13 | Cisco IWAN System |
|---|---|---|
| | |  |

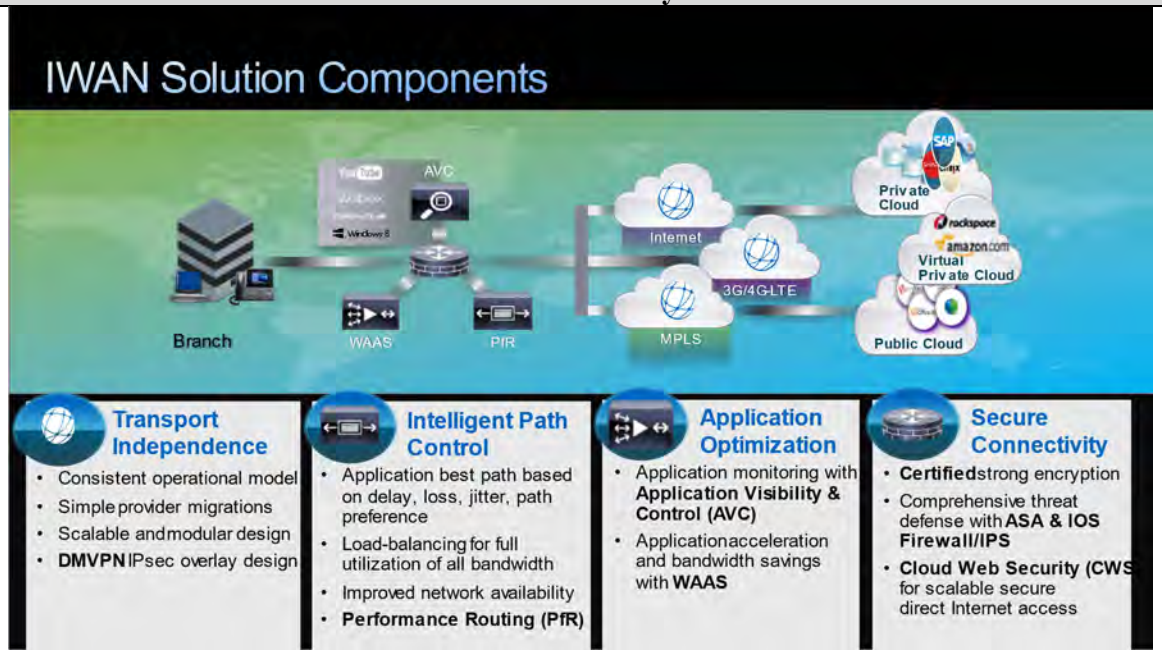
| No. | '111 Patent Claim 13 | Cisco IWAN System |
|---|---|---|
| | |  <br>Cisco IWAN - Uncompromised Experience |

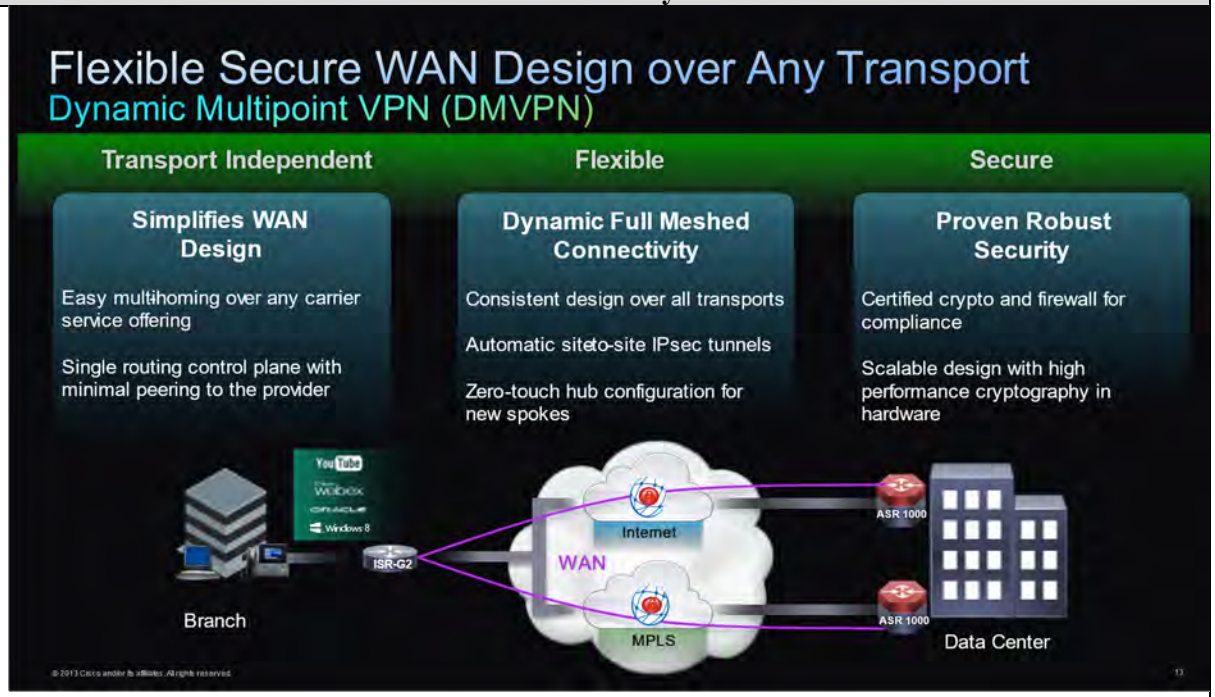


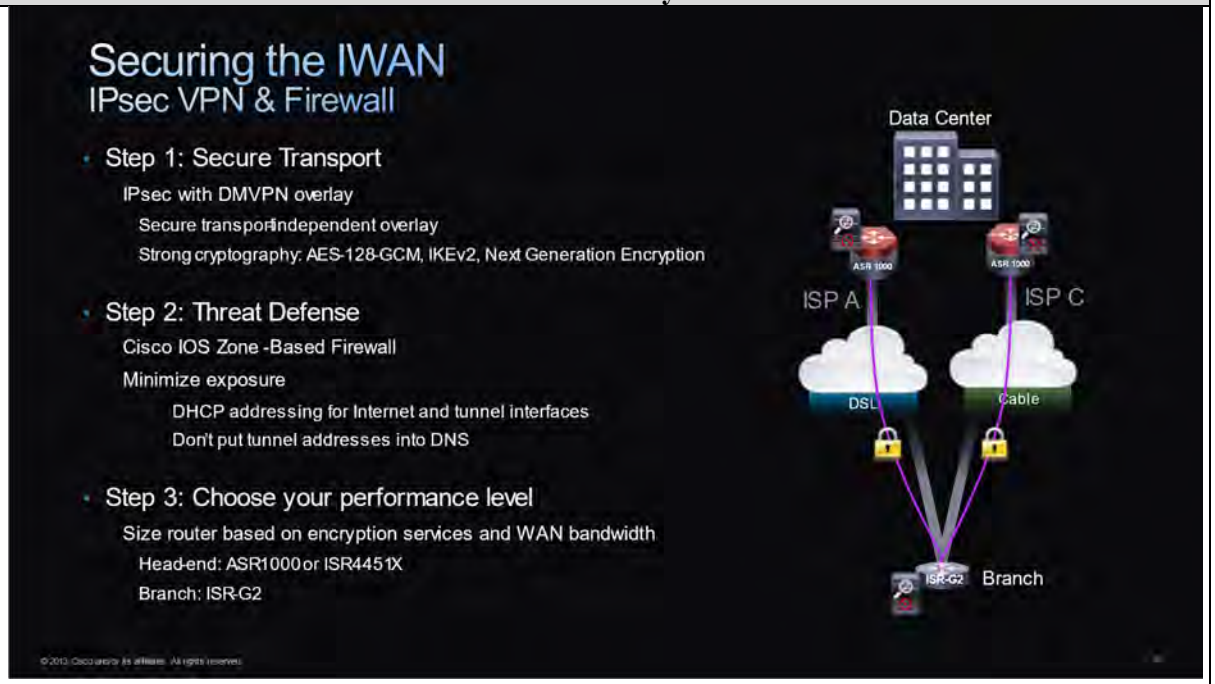
Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1053 of 1100

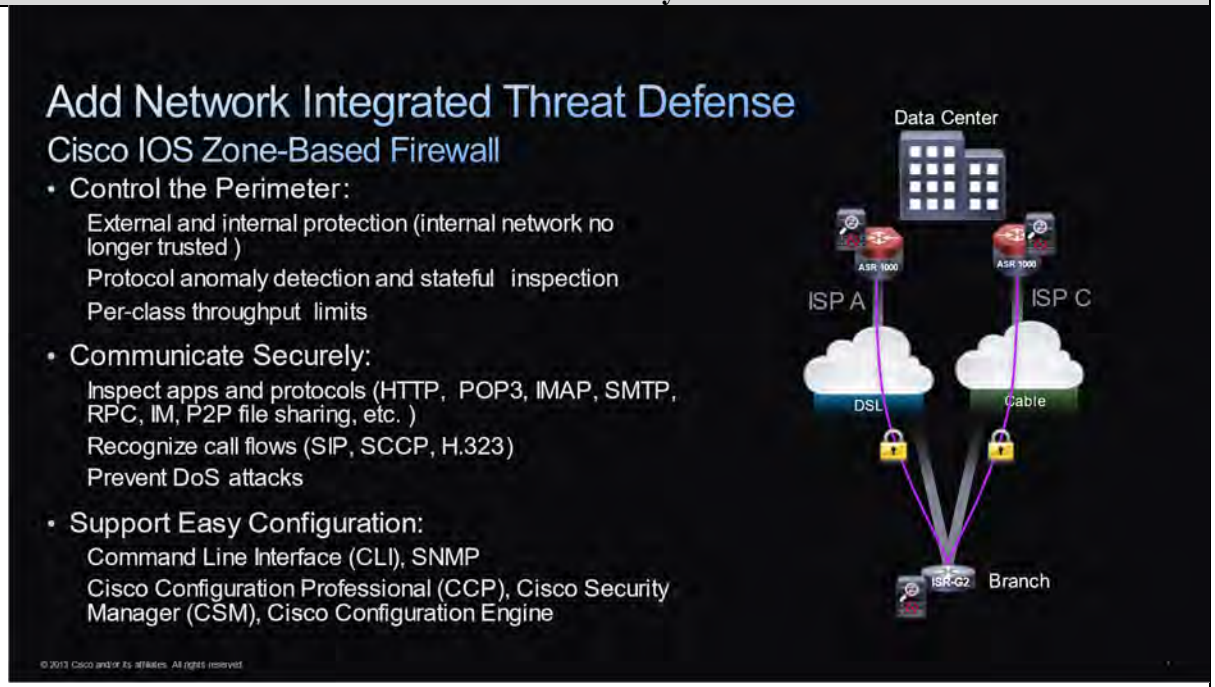
| No. | '111 Patent Claim 13 | Cisco IWAN System |
|---|---|---|
| | |  |

| No. | '111 Patent Claim 14 | Cisco IWAN System |
|---|---|---|
| 14 | The method according to claim 9, wherein the analyzing comprises performing Deep Packet Inspection (DPI) or using a DPI engine on the packet. | Cisco IWAN System discloses the method according to claim 9, wherein the analyzing comprises performing Deep Packet Inspection (DPI) or using a DPI engine on the packet.<br><br>For example, Cisco IWAN System discloses analyzing traffic flow metrics with inspection functionality.<br><br>*See supra* at Claim 9.<br><br>Cisco IWAN |

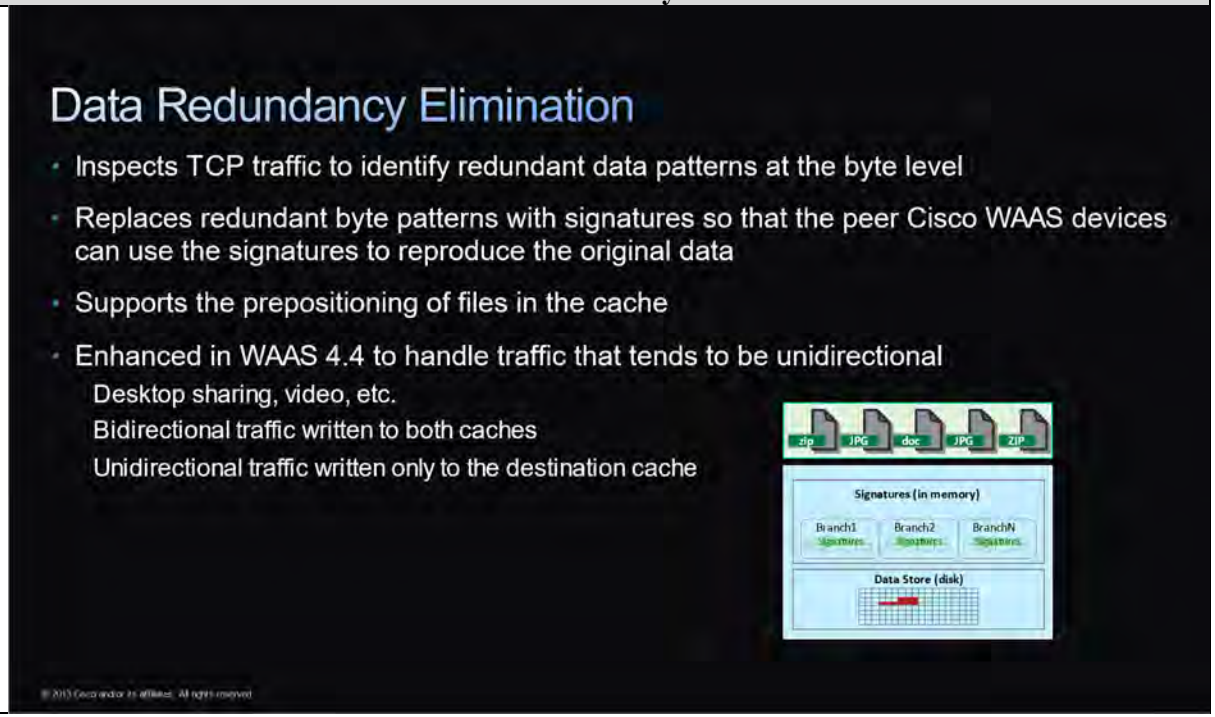
| No. | '111 Patent Claim 14 | Cisco IWAN System |
|---|---|---|
| | |  <br>Cisco IWAN |

| No. | '111 Patent Claim 14 | Cisco IWAN System |
|---|---|---|
| | |  <br>Cisco IWAN |

| No. | '111 Patent Claim 14 | Cisco IWAN System |
|---|---|---|
| | |  |

| No. | '111 Patent Claim 14 | Cisco IWAN System |
|---|---|---|
| | |  |

| No. | '111 Patent Claim 14 | Cisco IWAN System |
|---|---|---|
| | |  <br>Cisco IWAN - Uncompromised Experience |

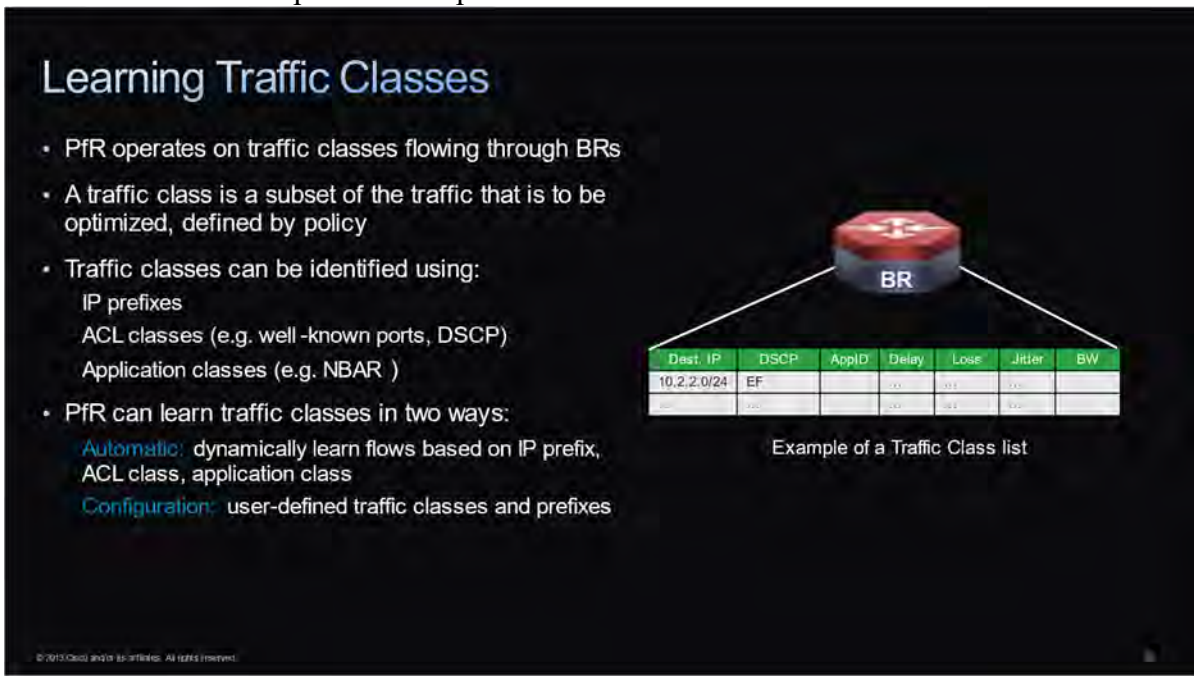
| No. | '111 Patent Claim 14 | Cisco IWAN System |
|---|---|---|
| | |  |

| No. | '111 Patent Claim 15 | Cisco IWAN System |
|---|---|---|
| 15[a] | The method according to claim 9, wherein the packet comprises distinct header and payload fields, and | Cisco IWAN System discloses the method according to claim 9, wherein the packet comprises distinct header and payload fields.<br><br>For example, Cisco IWAN System discloses data traffic that can be defined into subsets by prefixes. A person of ordinary skill in the art would understand that data traffic is made up of packets comprised of header and payload fields. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Cisco IWAN System is found to not meet this limitation, wherein the packet comprises distinct header and payload fields would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>*See supra* at Claim 9. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1060 of 1100

| No. | '111 Patent Claim 15 | Cisco IWAN System |
|---|---|---|
| | | Cisco IWAN - Uncompromised Experience |




Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Cisco IWAN System in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 15(a) of Exhibit E-4 renders the claim, including the present limitation, obvious. Below is an example.

For example, Swenson discloses packet flows with header and payload fields.

Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1061 of 1100

| No. | '111 Patent Claim 15 | Cisco IWAN System |
|---|---|---|
| | | interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0040] ("The flow analyzer 312 monitors large flows in the network, analyzes collected flow statistics to determine net-work throughput, and accordingly selects flows to be opti-mized. The flow analyzer 312 does not need to see all the flows in order to make an accurate estimate of network con-ditions. The flow analyzer 312 processes the traffic statistics stored in the flow cache 3 22 and user information stored in the subscriber log 324, for example, by associating network flows identified by source IP addresses to a mobile subscriber or user, which is identified by his or her current subscriber ID or device ID. The user flows are also mapped to a congestion level at the current sub-network (e.g., a cell with which the user devices are associated), so that an optimization decision can be made at the beginning of the data transmission.")<br><br>Swenson at [0049] ("The policy engine 314 defines policies for optimiz-ing large flows with media objects to mitigate network con-gestion. Detecting and acting on congestion in the network, the design focus of the network controller 140 is built on this very flexible policy engine. The policy engine 314 is capable of taking virtually any input, either deduced from HTTP headers and payload ( e.g., through RADIUS/Gx interface), or provided by the network infrastructure via API, and making decisions on how to apply optimization based on individual or a combination of these inputs. The optimization policies can be applied to large flows all the time or on a time-of-day basis, a per user basis, and/or depending on the network condition.")<br><br>Swenson at [0061] ("Once a flow is reported to the network controller 140, a flow cache entry is created for the flow in the flow cache 322. The flow cache entry keeps track of the flow and its associated bandwidth. For a flow that is marked in "continue" mode, each time the steering device 130 forwards a next portion of the flow payload to the network controller 140, the flow cache 3 22 updates the number of bytes for transmitted in the flow. By monitoring the number of bytes per flow over time, the flow analyzer 312 is capable of determining an estimate value of bandwidth associated with flow. Further-more, since the steering device 130 does not have infinite packet buffers, if congestion happens on the |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1062 of 1100

| No. | ’111 Patent Claim 15 | Cisco IWAN System |
|---|---|---|
| | | network link 416 from the steering device 130 to the user device 110, the TCP congestion control mechanism kicks in at the steering device 130, which may slows down and/or eventually stop receiving data over the network link 413 from origin server 160. During the congestion, the steering device 130 would not forward any data to the network controller 140, since the link 416 is congested and the network controller 140 would not be able to transmit data to the user device 110. Therefore, as an inline element, the network controller 140 can detect network con-gestions and estimate bandwidth associated with any flows of interest selected by the network controller 140. However, in the "continue" mode, the network controller 140 does not modify and transform the HTTP messaged it receives over the ICAP interface. The network controller 140 simply updates the flow statistics and returns the video or images to the steering device 130 for transmission to the user device 110.”)<br><br>Swenson at [0064] (Similar to the "continue" mode, after receiving the initial HTTP messages of a flow and determining to monitor the flow, the network controller 140 notify the steering device 130 to work in a "counting" mode for bandwidth monitoring. In contrast to the "continue" mode, when a matching flow is detected for "counting" mode, the steering device 130 for-wards the HTTP response directly to the user device 110. While at the same time, the steering device 130 send a cus-tomized ICAP message to the network controller 140 over the network link 425. In one embodiment, the customized ICAP message contains the HTTP request and response headers, as well as a count of payload size of the current flow. After updating the flow statistics, the network controller 140 may acknowledge the gateway over the network line 426. In the "counting" mode, the network controller 140 does not join the network response path as an inline network element, but simply listens to the counting of flow size. The benefit of the "counting" mode is to off-load the network controller 140 from ingesting and forwarding the network flow on the net- work response path, while still enabling the detection of con-gestions and estimation of bandwidth associated with the flows of interest.”)<br><br>Swenson at [0065] (“FIG. 5 is a block diagram illustrating an example event trace of "continue" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 512 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1063 of 1100

| No. | '111 Patent Claim 15 | Cisco IWAN System |
|---|---|---|
| | | one embodiment, the steering device 130 for-wards the HTTP get request 512 to the intended origin server 160 and receives a response 514 back from the origin server 160. The steering device 130 then sends an ICAP request message 516 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 518. Upon receiving the instruc-tion, the steering device 130 re-writes the response 514 to an HTTP redirect response 520, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 520 directly to the user device 110. In case the flow dose not contain video or image objects, or the network controller 140 determines not to monitor the flow, the steering device 13 0 would forward the response to the user device 110.")<br><br>Swenson at [0069] ("FIG. 6 is a block diagram illustrating an example event trace of "counting" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 612 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 612 to the intended origin server 160 and receives a response 614 back from the origin server 160. The steering device 130 then sends an ICAP request message 616 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 618. Upon receiving the instruc-tion, the steering device 130 re-writes the response 614 to an HTTP redirect response 620, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 620 directly to the user device 110. In case the flow dose not contain video or image objects that need to be redirected, the steering device 130 would forward the response to the user device 110.")<br><br>Swenson at [0073] ("FIG. 7 is a block diagram illustrating one embodi-ment of an example of internal components of the flow cache. The flow cache map 700 comprises a plurality of flow cache entries, such as flow cache entries 710 and 712 indexed by a hash. Not shown in |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1064 of 1100

| No. | '111 Patent Claim 15 | Cisco IWAN System |
|---|---|---|
| | | the example diagram is a possible linked list behind each flow cache entry which allows chaining of flow cache entries for a given hash index. The hash into the flow cache may be based on source IP address, MAC address, subscriber ID, or other identifier indicative of a given sub-scriber, group of subscribers or subscriber's device.")<br><br>Swenson at [0079] ("In the bandwidth calculation, flows are categorized into buckets based on the size of the objects being transferred. Small objects may not be factored into the bandwidth calcu-lation since they may come and go within a single interval. For example, flows with payload size less than 50 kB may be ignored because a transfer of 50 kB may never reach the full potential throughput of the link. While larger flows may reach the full throughput of the link for a long period of time intervals, they are grouped into 50-75 kB, 75-100 kB and 100 kB+ buckets because the characteristics of these flow sizes can be different, hence the bandwidth for each of the buckets is measured and calculated separately. In other embodiments, the flow size ranges (e.g., 50-75 kB, 75-100 kB and l00kB+) of the buckets may be altered depending on the network traffic and size of objects transmitted. Furthermore, the bucket sizes can also be adjusted based on network topology, such as buffer size, prior to transmission to the client. The calculated bandwidth per bucket is stored in a queue structure that allows for the computing and updating of minimum, maximum, and/or average measurements for each bucket. In one embodiment, the 100 kB+ bucket's current tail entry is checked against the average bandwidth for the 100 kB+ bucket. If the current entry is less than the average multiplied by the number of entries in the queue, the current entry is added to the bandwidth calculation for the current interval. This scheme can filter out large bursts of data from tempo-rarily idle flows. If the bandwidth exceeds the value, a number of bytes (e.g., 125 kB) will be subtracted from the current entry to account for TCP buffers in the network.")<br><br>Swenson at [0083] ("When a new flow is observed, flow cache entries are searched by matching source IP address 722 if the subscriber id or other identifiers of the flow are not available. In case of multiple users sharing an IP address, the flow analyzer 312 needs to find patterns or other identifiers in the flows to map them to particular subscribers. Flows without identified sub-scribers are added to the flow cache block under the default user flows 726, which is a default holding place for the new flows. The flow analyzer 312 later will scan through the default user flows that contain cookies or other identifiers that may be used to determine a real user or subscriber associated with the flow. If a flow contains |

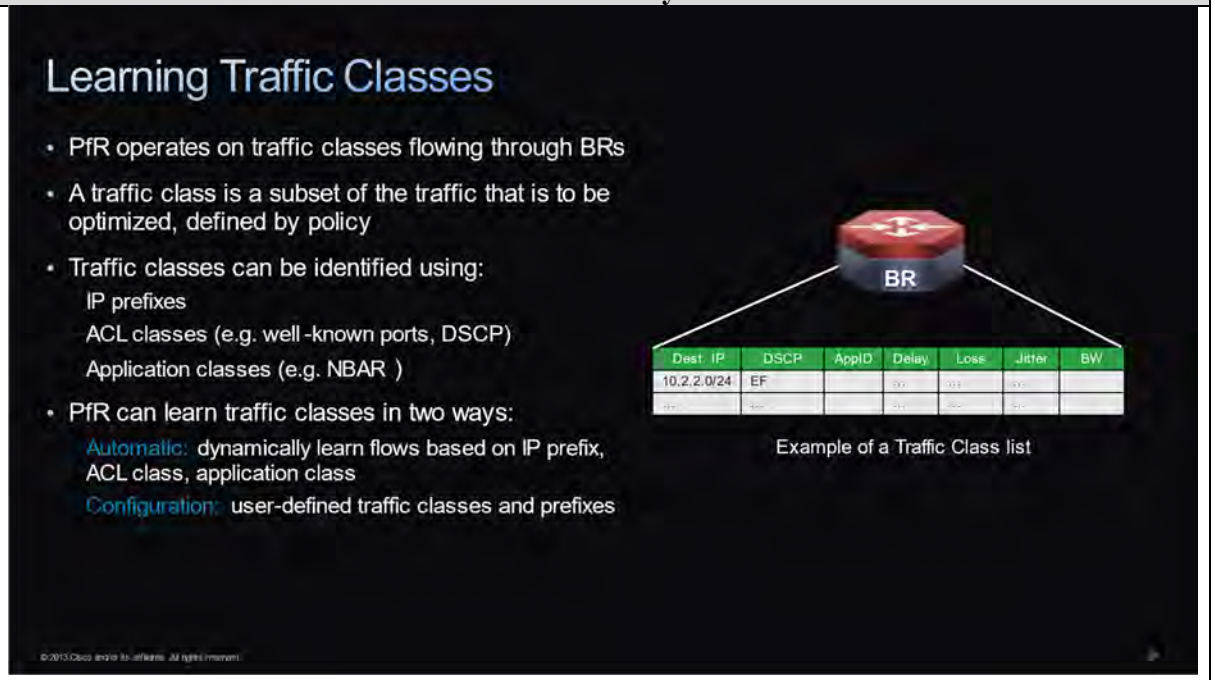


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1065 of 1100

| No. | '111 Patent Claim 15 | Cisco IWAN System |
|---|---|---|
| | | identifiers not associated with an existing real user, a new user or subscriber is created and the user flow block is moved to newly created (or mapped) user or subscriber.") |
| 15[b] | wherein the analyzing comprises checking part of, or whole of, the payload field. | Cisco IWAN System discloses wherein the analyzing comprises checking part of, or whole of, the payload field.<br><br>For example, Cisco IWAN System discloses determining traffic classes by identifying traffic characteristics. A person of ordinary skill in the art would understand that one characteristic of data traffic that can be used to identify traffic class is the payload field. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Cisco IWAN System is found to not meet this limitation, wherein the analyzing comprises checking part of, or whole of, the payload field would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>Cisco IWAN - Uncompromised Experience |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1066 of 1100

| No. | '111 Patent Claim 15 | Cisco IWAN System |
|---|---|---|
| | |  <br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Cisco IWAN System in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 15(b) of Exhibit E-4 renders the claim, including the present limitation, obvious. Below is an example.<br><br>For example, Swenson discloses inspecting the payload of a packet flow.<br><br>Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the user device traffic flows onto the network and vice versa. In one embodiment, the steering device 130 categorizes traffic routed through it to identify flows of inter-est for further inspection at the network controller 140. Alter-natively, the network controller 140 interfaces with the steer-ing device 130 to coordinate the monitoring and categorization of |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1067 of 1100

| No. | '111 Patent Claim 15 | Cisco IWAN System |
|---|---|---|
| | | network traffic, such as identifying large and small objects in HTTP traffic flows. In this case, the steering device 130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.")<br><br>Swenson at [0040] ("The flow analyzer 312 monitors large flows in the network, analyzes collected flow statistics to determine net-work throughput, and accordingly selects flows to be opti-mized. The flow analyzer 312 does not need to see all the flows in order to make an accurate estimate of network con-ditions. The flow analyzer 312 processes the traffic statistics stored in the flow cache 3 22 and user information stored in the subscriber log 324, for example, by associating network flows identified by source IP addresses to a mobile subscriber or user, which is identified by his or her current subscriber ID or device ID. The user flows are also mapped to a congestion level at the current sub-network (e.g., a cell with which the user devices are associated), so that an optimization decision can be made at the beginning of the data transmission.")<br><br>Swenson at [0049] ("The policy engine 314 defines policies for optimiz-ing large flows with media objects to mitigate network con-gestion. Detecting and acting on congestion in the network, the design focus of the network controller 140 is built on this very flexible policy engine. The policy engine 314 is capable of taking virtually any input, either deduced from HTTP headers and payload ( e.g., through RADIUS/Gx interface), or provided by the network infrastructure via API, and making decisions on how to apply optimization based on individual or a combination of these inputs. The optimization policies can be applied to large flows all the time or on a time-of-day basis, a per user basis, and/or depending on the network condition.")<br><br>Swenson at [0061] ("Once a flow is reported to the network controller 140, a flow cache entry is created for the flow in the flow cache 322. The flow cache entry keeps track of the flow and its associated bandwidth. For a flow that is marked in "continue" mode, each time the steering device 130 forwards a next portion of the flow payload to the network controller 140, the flow cache 3 22 updates the number of bytes for transmitted in the flow. By monitoring the number of bytes per flow over time, the flow analyzer 312 is capable of determining an estimate value of bandwidth associated with flow. Further-more, since the steering device 130 does not have infinite packet buffers, if congestion happens on the network link 416 from the steering device 130 to the user device 110, the TCP congestion |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1068 of 1100

| No. | '111 Patent Claim 15 | Cisco IWAN System |
|---|---|---|
| | | control mechanism kicks in at the steering device 130, which may slows down and/or eventually stop receiving data over the network link 413 from origin server 160. During the congestion, the steering device 130 would not forward any data to the network controller 140, since the link 416 is congested and the network controller 140 would not be able to transmit data to the user device 110. Therefore, as an inline element, the network controller 140 can detect network con-gestions and estimate bandwidth associated with any flows of interest selected by the network controller 140. However, in the "continue" mode, the network controller 140 does not modify and transform the HTTP messaged it receives over the ICAP interface. The network controller 140 simply updates the flow statistics and returns the video or images to the steering device 130 for transmission to the user device 110.")<br><br>Swenson at [0064] (Similar to the "continue" mode, after receiving the initial HTTP messages of a flow and determining to monitor the flow, the network controller 140 notify the steering device 130 to work in a "counting" mode for bandwidth monitoring. In contrast to the "continue" mode, when a matching flow is detected for "counting" mode, the steering device 130 for-wards the HTTP response directly to the user device 110. While at the same time, the steering device 130 send a cus-tomized ICAP message to the network controller 140 over the network link 425. In one embodiment, the customized ICAP message contains the HTTP request and response headers, as well as a count of payload size of the current flow. After updating the flow statistics, the network controller 140 may acknowledge the gateway over the network line 426. In the "counting" mode, the network controller 140 does not join the network response path as an inline network element, but simply listens to the counting of flow size. The benefit of the "counting" mode is to off-load the network controller 140 from ingesting and forwarding the network flow on the net- work response path, while still enabling the detection of con-gestions and estimation of bandwidth associated with the flows of interest.")<br><br>Swenson at [0065] ("FIG. 5 is a block diagram illustrating an example event trace of "continue" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 512 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 512 to the |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1069 of 1100

| No. | '111 Patent Claim 15 | Cisco IWAN System |
|---|---|---|
| | | intended origin server 160 and receives a response 514 back from the origin server 160. The steering device 130 then sends an ICAP request message 516 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 518. Upon receiving the instruc-tion, the steering device 130 re-writes the response 514 to an HTTP redirect response 520, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 520 directly to the user device 110. In case the flow dose not contain video or image objects, or the network controller 140 determines not to monitor the flow, the steering device 13 0 would forward the response to the user device 110.")<br><br>Swenson at [0069] ("FIG. 6 is a block diagram illustrating an example event trace of "counting" working mode between the user device 110, steering device 130, network controller 140, video optimizer 150, and origin server 160. The process starts when the user device 110 initiates an HTTP GET request 612 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 612 to the intended origin server 160 and receives a response 614 back from the origin server 160. The steering device 130 then sends an ICAP request message 616 comprising the HTTP GET request header and a portion of the response payload to the network controller 140, which inspects the message to determine whether to monitor the flow or optimize the video. In this case, the network controller 140 responds with a redirect to optimize the video in ICAP response 618. Upon receiving the instruc-tion, the steering device 130 re-writes the response 614 to an HTTP redirect response 620, causing the user device 110 to request the video file from the video optimizer 150. In another embodiment, the network controller 140 sends the HTTP redirect request 620 directly to the user device 110. In case the flow dose not contain video or image objects that need to be redirected, the steering device 130 would forward the response to the user device 110.")<br><br>Swenson at [0073] ("FIG. 7 is a block diagram illustrating one embodi-ment of an example of internal components of the flow cache. The flow cache map 700 comprises a plurality of flow cache entries, such as flow cache entries 710 and 712 indexed by a hash. Not shown in the example diagram is a possible linked list behind each flow cache entry which allows |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1070 of 1100

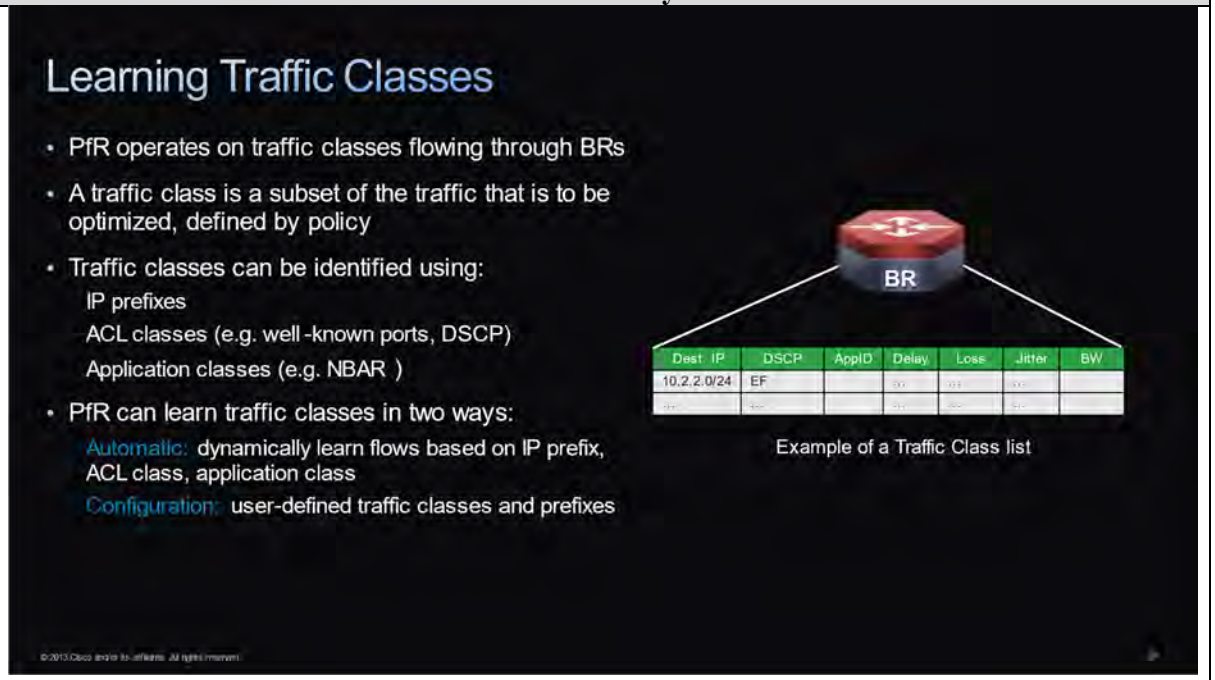
| No. | '111 Patent Claim 15 | Cisco IWAN System |
|---|---|---|
| | | chaining of flow cache entries for a given hash index. The hash into the flow cache may be based on source IP address, MAC address, subscriber ID, or other identifier indicative of a given sub-scriber, group of subscribers or subscriber's device.")<br><br>Swenson at [0079] ("In the bandwidth calculation, flows are categorized into buckets based on the size of the objects being transferred. Small objects may not be factored into the bandwidth calcu-lation since they may come and go within a single interval. For example, flows with payload size less than 50 kB may be ignored because a transfer of 50 kB may never reach the full potential throughput of the link. While larger flows may reach the full throughput of the link for a long period of time intervals, they are grouped into 50-75 kB, 75-100 kB and 100 kB+ buckets because the characteristics of these flow sizes can be different, hence the bandwidth for each of the buckets is measured and calculated separately. In other embodiments, the flow size ranges (e.g., 50-75 kB, 75-100 kB and l00kB+) of the buckets may be altered depending on the network traffic and size of objects transmitted. Furthermore, the bucket sizes can also be adjusted based on network topology, such as buffer size, prior to transmission to the client. The calculated bandwidth per bucket is stored in a queue structure that allows for the computing and updating of minimum, maximum, and/or average measurements for each bucket. In one embodiment, the 100 kB+ bucket's current tail entry is checked against the average bandwidth for the 100 kB+ bucket. If the current entry is less than the average multiplied by the number of entries in the queue, the current entry is added to the bandwidth calculation for the current interval. This scheme can filter out large bursts of data from tempo-rarily idle flows. If the bandwidth exceeds the value, a number of bytes (e.g., 125 kB) will be subtracted from the current entry to account for TCP buffers in the network.")<br><br>Swenson at [0083] ("When a new flow is observed, flow cache entries are searched by matching source IP address 722 if the subscriber id or other identifiers of the flow are not available. In case of multiple users sharing an IP address, the flow analyzer 312 needs to find patterns or other identifiers in the flows to map them to particular subscribers. Flows without identified sub-scribers are added to the flow cache block under the default user flows 726, which is a default holding place for the new flows. The flow analyzer 312 later will scan through the default user flows that contain cookies or other identifiers that may be used to determine a real user or subscriber associated with the flow. If a flow contains |

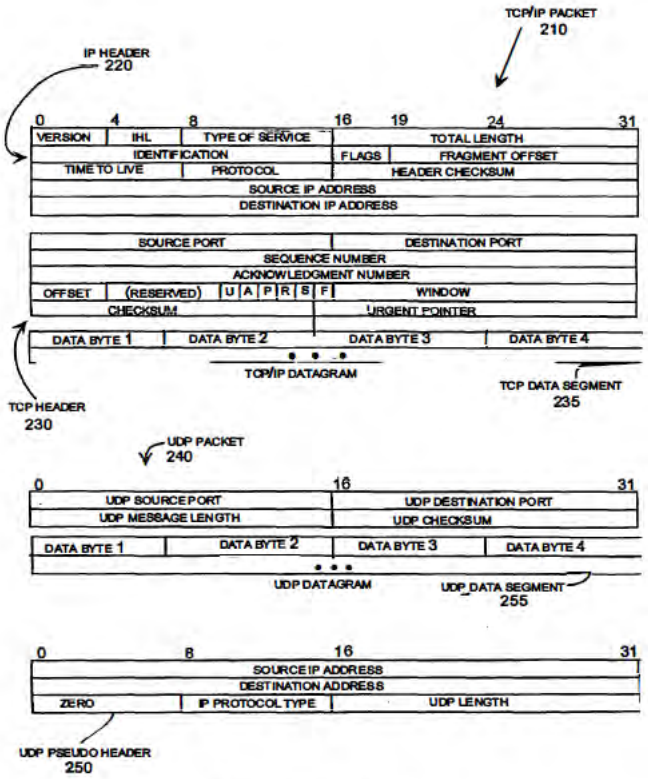
| No. | '111 Patent Claim 15 | Cisco IWAN System |
|---|---|---|
| | | identifiers not associated with an existing real user, a new user or subscriber is created and the user flow block is moved to newly created (or mapped) user or subscriber.") |

| No. | '111 Patent Claim 16 | Cisco IWAN System |
|---|---|---|
| 16[a] | The method according to claim 1, wherein the packet comprises distinct header and payload fields, | Cisco IWAN System discloses the method according to claim 1, wherein the packet comprises distinct header and payload fields.<br><br>*See supra* at Claim 1, 15[a]. |
| 16[b] | the header comprises one or more flag bits, and | Cisco IWAN System discloses the header comprises one or more flag bits.<br><br>On information and belief, the Cisco IWAN System discloses headers comprised of one or more flag bits. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Cisco IWAN System is found to not meet this limitation, the header comprises one or more flag bits would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>Cisco IWAN - Uncompromised Experience |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1072 of 1100

| No. | '111 Patent Claim 16 | Cisco IWAN System |
|---|---|---|
| | |  <br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Cisco IWAN System in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 16[b] of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references.<br><br>For example, Copeland discloses packet headers with flag bits.<br><br>Copeland at Figure 2 |

| No. | '111 Patent Claim 16 | Cisco IWAN System |
|---|---|---|
| | |  <br>FIG. 2<br><br>Copeland at [0076] ("FIG. 2 illustrates an exemplary TCP/IP packet or datagram 210 and an exemplary UDP datagram 240. In a typical TCP/IP packet like 210, each packet typically includes a header portion comprising an IP header 220 and a TCP header 230, followed by a data portion that contains the information to be communicated in the packet. The information in the IP header 220 contained in a TCP/IP packet 210, or any other IP packet, contains the IP addresses and assures that the packet is delivered to the right host. The transport layer protocol (TCP) header follows the Internet protocol header and specifies the port numbers for the associated service.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1074 of 1100

| No. | '111 Patent Claim 16 | Cisco IWAN System |
|---|---|---|
| | | Copeland at [0077] ("The header portion in the typical TCP/IP datagram 210 is 40 bytes including 20 bytes of IP header 220 information and 20 bytes of TCP header 230 information. The data portion or segment associated with the packet 210 follows the header information.")<br><br>Copeland at [0078] ("In regards to a typical IP packet 210, the first 4 bits of the IP header 220 identify the Internet protocol (IP) version. The following 4 bits identify the IP header length in 32 bit words. The next 8 bits differentiate the type of service by describing how the packet should be handled in transit. The following 16 bits convey the total packet length.")<br><br>Copeland at [0081] ("In a TCP/IP datagram 210, the initial data of the IP datagram is the TCP header 230 information. The initial TCP header 230 information includes the 16-bit source and 16-bit destination port numbers. A 32-bit sequence number for the data in the packet follows the port numbers. Following the sequence number is a 32-bit acknowledgement number. If an ACK flag (discussed below) is set, this number is the next sequence number the sender of the packet expects to receive. Next is a 4-bit data offset, which is the number of 32-bit words in the TCP header. A 6-bit reserved field follows.")<br><br>Copeland at [0082] ("Following the reserved field, the next 6 bits are a series of one-bit flags, shown in FIG. 2 as flags U, A, P, R, S, F. The first flag is the urgent flag (U). If the U flag is set, it indicates that the urgent pointer is valid and points to urgent data that should be acted upon as soon as possible. The next flag is the A ( or ACK or "acknowledgment") flag. The ACK flag indicates that an acknowledgment number is valid, and acknowledges that data has been received. The next flag, the push (P) flag, tells the receiving end to push all buffered data to the receiving application. The reset (R) flag is the following flag, which terminates both ends of the TCP connection. Next, the S (or SYN for "synchronize") flag is set in the initial packet of a TCP connection where both ends have to synchronize their TCP buffers. Following the SYN flag is the F (for FIN or "finish") flag. This flag signifies that the sending end of the communication and the host will not send any more data but still may acknowledge data that is received.") |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1075 of 1100

| No. | '111 Patent Claim 16 | Cisco IWAN System |
|---|---|---|
| | | Copeland at [0083] ("Following the TCP flag bits is a 16-bit receive window size field that specifies the amount of space avail-able in the receive buffer for the TCP connection. The checksum of the TCP header is a 16-bit field. Following the checksum is a 16 bit urgent pointer that points to the urgent data. The TCP/IP datagram data follows the TCP header.")<br><br>Copeland at [0116] ("These steps generally require manipulations of quantities such as IP addresses, packet length, header length, start times, end times, port numbers, and other packet related information. Usually, though not necessarily, these quanti-ties take the form of electrical, magnetic, or optical signals capable of being stored, transferred, combined, compared, or otherwise manipulated. It is conventional for those skilled in the art to refer to these signals as bits, bytes, words, values, elements, symbols, characters, terms, numbers, points, records, objects, images, files or the like. It should be kept in mind, however, that these and similar terms should be associated with appropriate quantities for computer opera-tions and that these terms are merely conventional labels applied to quantities that exist within and during operation of the computer.")<br><br>As another example, Kempf discloses packet headers with flag bits.<br><br>Kempf at [0081] ("In one embodiment, OpenFlow is modified to pro-vide rules for GTP TEID Routing. FIG. 17 is a diagram of one embodiment of the OpenFlow flow table modification for GTP TEID routing. An OpenFlow switch that supports TEID routing matches on the 2 byte (16 bit) collection of header fields and the 4 byte (32 bit) GTP TEID, in addition to other OpenFlow header fields, in at least one flow table ( e.g., the first flow table). The GTP TEID flag can be wildcarded (i.e. matches are "don't care"). In one embodiment, the EPC pro-tocols do not assign any meaning to TEIDs other than as an endpoint identifier for tunnels, like ports in standard UDP/ TCP transport protocols. In other embodiments, the TEIDs can have a correlated meaning or semantics. The GTP header flags field can also be wildcarded, this can be partially matched by combining the following bitmasks: 0xFF00- Match the Message Type field; 0xe0-Match the Version field; 0xl0-Match the PT field; 0x04-Match the E field; 0x02- Match the S field; and 0x0l-Match the PN field.")<br><br>Kempf at [0082] ("In one embodiment, OpenFlow can be modified to support virtual ports for fast path GTP TEID encapsulation and decapsulation. An OpenFlow mobile gateway |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1076 of 1100

| No. | '111 Patent Claim 16 | Cisco IWAN System |
|---|---|---|
| | | can be used to support GTP encapsulation and decapsulation with virtual ports. The GTP encapsulation and decapsulation virtual ports can be used for fast encapsulation and decapsulation of user data packets within GTP-U tunnels, and can be designed simply enough that they can be implemented in hardware or firmware. For this reason, GTP virtual ports may have the following restrictions on traffic they will handle: Protocol Type (PT) field= 1, where GTP encapsulation ports only sup-port GTP, not GTP' (PT field=0); Extension Header flag (E)=0, where no extension headers are supported, Sequence Number flag (S)=0, where no sequence numbers are sup-ported; N-PDU flag (PN)=0; and Message type=255, where Only G-PDU messages, i.e. tunneled user data, is supported in the fast path.")<br><br>Kempf at [0083] ("If a packet either needs encapsulation or arrives encapsulated with nonzero header flags, header extensions, and/or the GTP-U packet is not a G-PDU packet (i.e. it is a GTP-U control packet), the processing must proceed via the gateway's slow path (software) control plane. GTP-C and GTP' packets directed to the gateway's IP address are a result of mis-configuration and are in error. They must be sent to the OpenFlow controller, since these packets are handled by the S-GW-C and P-GW-C control plane entities in the cloud computing system or to the billing entity handling GTP' and not the S-GW-D and P-GW-D data plane switches.")<br><br>Kempf at [0088] ("To support slow path encapsulation, the software control plane on the switch maintains a hash table with keys calculated from the GTP-U TEID. The TEID hash keys are calculated using a suitable hash algorithm with low collision frequency, for example SHA-1. The flow table entries contain a record of how the packet header, including the GTP encap-sulation header, should be configured. This includes: the same header fields as for the hardware or firmware encapsu-lation table in FIG.18; values for the GTP header flags (PT, E, S, and PN); the sequence number and/or the N-PDU number if any; if the E flag is 1, then the flow table contains a list of the extension headers, including their types, which the slow path should insert into the GTP header.")<br><br>Kempf at [0092] ("In one embodiment, the system implements a GTP fast path decapsulation virtual port. When requested by the S-GW and P-GW control plane software running in the cloud computing system, the gateway switch installs rules and actions for routing GTP encapsulated packets out of GTP tunnels. The rules match the GTP header |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1077 of 1100

| No. | '111 Patent Claim 16 | Cisco IWAN System |
|---|---|---|
| | | flags and the GTP TEID for the packet, in the modified OpenFlow flow table shown in FIG. 17 as follows: the IP destination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); the UDP destination port is the GTP-U destination port (2152); and the header fields and message type field is wildcarded with the flag 0XFFF0 and the upper two bytes of the field match the G-PDU message type (255) while the lower two bytes match 0x30, i.e. the packet is a GTP packet not a GTP' packet and the version number is 1.")<br><br>Kempf at [0094] ("In one embodiment, the system implements han-dling of GTP-U control packets. The OpenFlow controller programs the gateway switch flow tables with 5 rules for each gateway switch IP address used for GTP traffic. These rules contain specified values for the following fields: the IP des-tination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); the UDP destination port is the GTP-U destination port (2152); the GTP header flags and message type field is wildcarded with 0xFFF0; the value of the header flags field is 0x30, i.e. the version number is 1 and the PT field is 1; and the value of the message type field is one of 1 (Echo Request), 2 (Echo Response), 26 (Error Indication), 31 (Support for Extension Headers Notification), or 254 (End Marker).")<br><br>Kempf at [0098] ("The header flags and message type fields for the three rules are wildcarded with the following bitmasks and match as follows: bitmask 0xFFF4 and the upper two bytes match the G-PDU message type (255) while the lower two bytes are Ox34, indicating that the version number is 1, the packet is a GTP packet, and there is an extension header present; bitmask 0xFFFF2 and the upper two bytes match the G-PDU message type (255) while the lower two bytes are 0x32, indicating that the version number is 1, the packet is a GTP packet, and there is a sequence number present; and bitmask 0xFF0l and the upper two bytes match the G-PDU message type (255) while the lower two bytes are 0x31, indicating that the version number is 1, the packet is a GTP packet, and a N-PDU is present.")<br><br>Kempf at [0114] ("The gtp_type_n_flags field contains the GTP mes-sage type in the upper 8 bits and the GTP header flags in the lower 8 bits. The gtp_teid field contains the GTP TEID. The gtp_ wildcard field indicates whether the GTP type and flags and TEID should be matched. If the lower four bits are 1, the type and flags field should be ignored, while if |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1078 of 1100

| No. | '111 Patent Claim 16 | Cisco IWAN System |
|---|---|---|
| | | the upper four bits are 1, the TEID should be ignored. If the lower bits are 0, the type and fields flag should be matched subject to the flags in the gtp_flag_mask field, while if the upper bits are 0 the TEID should be matched. The mask is combined with the message type and header field of the packet using logical AND; the result becomes the value of the match. Only those parts of the field in which the mask has a 1 value are matched.")<br><br>Kempf at [0117] ("The gtp_type_n_flags field contains the GTP mes-sage type in the upper 8 bits and the GTP header flags in the lower 8 bits. The gtp_<br>teid field contains the GRP TEID. When the value of the oxm_type ( oxm_class+oxm_field is GTP _ MATCH and the HM bit is zero, the flaw's GTP header must match these values exactly. If the HM flag is one, the value contains an ersmt_gtp_match field and an ermst_gtp_mask field, as specified by the OpenFlow 1.2 specification. We define ermst_gtp_mask field for selecting flows based on the settings of flag bits:<br><br>`struct ermst_gtp_mask {`<br>`  uint32_t gtp_wildcard;`<br>`  uint16_t gtp_flag_mask;`<br>`};`<br><br>Kempf at [0118] ("The gtp_ wildcard field indicates whether the TEID should be matched. If the value is 0xFFFFFFFF, the TEID should be matched and not the flags, if the value is 0x00000000, the flags should be matched and not the TEID. If the gtp_ wildcard indicates the flags should be matched, the gtp_flag_mask is combined with the message type and header field of the packet using logical AND, the result becomes the value of the match. Only those parts of the field in which the mask has a 1 value are matched.") |
| 16[c] | wherein the packet-applicable criterion is that one or more of the flag bits is set. | Cisco IWAN System discloses wherein the packet-applicable criterion is that one or more of the flag bits is set.<br><br>On information and belief, the Cisco IWAN System discloses headers comprised of one or more flag bits. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Cisco IWAN System is found to not meet this limitation, wherein the packet applicable criterion is that one or |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1079 of 1100

| No. | '111 Patent Claim 16 | Cisco IWAN System |
|---|---|---|
| | | more of the flag bits is set would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Cisco IWAN System in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 16[c] of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references.<br><br>For example, Copeland discloses packet specific characteristics including flag bits that are set.<br><br>Copeland at [0081] ("In a TCP/IP datagram 210, the initial data of the IP datagram is the TCP header 230 information. The initial TCP header 230 information includes the 16-bit source and 16-bit destination port numbers. A 32-bit sequence number for the data in the packet follows the port numbers. Following the sequence number is a 32-bit acknowledgement number. If an ACK flag (discussed below) is set, this number is the next sequence number the sender of the packet expects to receive. Next is a 4-bit data offset, which is the number of 32-bit words in the TCP header. A 6-bit reserved field follows.")<br><br>Copeland at [0082] ("Following the reserved field, the next 6 bits are a series of one-bit flags, shown in FIG. 2 as flags U, A, P, R, S, F. The first flag is the urgent flag (U). If the U flag is set, it indicates that the urgent pointer is valid and points to urgent data that should be acted upon as soon as possible. The next flag is the A ( or ACK or "acknowledgment") flag. The ACK flag indicates that an acknowledgment number is valid, and acknowledges that data has been received. The next flag, the push (P) flag, tells the receiving end to push all buffered data to the receiving application. The reset (R) flag is the following flag, which terminates both ends of the TCP connection. Next, the S (or SYN for "synchronize") flag is set in the initial packet of a TCP connection where both ends have to synchronize their TCP buffers. Following the SYN flag is the F (for FIN or "finish") flag. This flag signifies that the sending end of the communication and the host will not send any more data but still may acknowledge data that is received.") |

| No. | '111 Patent Claim 16 | Cisco IWAN System |
|---|---|---|
| | | Copeland at [0083] ("Following the TCP flag bits is a 16-bit receive window size field that specifies the amount of space avail-able in the receive buffer for the TCP connection. The checksum of the TCP header is a 16-bit field. Following the checksum is a 16 bit urgent pointer that points to the urgent data. The TCP/IP datagram data follows the TCP header.")<br><br>Copeland at [0089] ("FIG. 3 illustrates an exemplary TCP/IP session 300. As discussed in reference to FIG. 2, the SYN flag is set whenever one host initiates a session with another host. In the initial packet, Hostl sends a message with only the SYN flag set. The SYN flag is designed to establish a TCP connection and allow both ends to synchronize their TCP buffers. Hostl provides the sequence of the first data packet it will send.")<br><br>Copeland at [0125] ("For purposes of the description, which follows, the IP address with the lower value, when considered as a 32-bit unsigned integer, is designated ip[0] and the corresponding port number is designated pt[0]. The higher IP address is designated ip[l] and the corresponding TCP or UDP port number is designated pt[l]. At some point, either pt[0] or pt[l] may be designated the "server" port by setting an appropriate bit in a bit map that is part of the flow record (record "state", bit 1 or 2 is set).")<br><br>Copeland at [0145] ("A list IP of addresses contacted or probed by each host can be maintained. When this list indicates that more than a threshold number of other hosts (e.g., 8) have been contacted in the same subnet, CI is added to the to the host and a bit in the host record is set to indicate that the host has received CI for "address scanning." Note that the number of hosts to designate a scan is not required to be a fixed value, but could be adjusted based on the sample rate or other means to enhance the accuracy making the number of hosts scanned "statistically significant". These and other values of concern index are shown for non-flow based events in FIG. 7.")<br><br>As another example, Kempf flow table matches in which the flag bits is set,<br><br>Kempf at [0081] ("In one embodiment, OpenFlow is modified to pro-vide rules for GTP TEID Routing. FIG. 17 is a diagram of one embodiment of the OpenFlow flow table modification for GTP TEID routing. An OpenFlow switch that supports TEID routing matches on the 2 byte (16 bit) collection of header fields and the 4 byte (32 bit) GTP TEID, in addition to other OpenFlow header fields, in at least one flow table ( e.g., the first flow |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1081 of 1100

| No. | '111 Patent Claim 16 | Cisco IWAN System |
|---|---|---|
| | | table). The GTP TEID flag can be wildcarded (i.e. matches are "don't care"). In one embodiment, the EPC pro-tocols do not assign any meaning to TEIDs other than as an endpoint identifier for tunnels, like ports in standard UDP/ TCP transport protocols. In other embodiments, the TEIDs can have a correlated meaning or semantics. The GTP header flags field can also be wildcarded, this can be partially matched by combining the following bitmasks: 0xFF00- Match the Message Type field; 0xe0-Match the Version field; 0xl0-Match the PT field; 0x04-Match the E field; 0x02- Match the S field; and 0x0l-Match the PN field.")<br><br>Kempf at [0082] ("In one embodiment, OpenFlow can be modified to support virtual ports for fast path GTP TEID encapsulation and decapsulation. An OpenFlow mobile gateway can be used to support GTP encapsulation and decapsulation with virtual ports. The GTP encapsulation and decapsulation virtual ports can be used for fast encapsulation and decapsulation of user data packets within GTP-U tunnels, and can be designed simply enough that they can be implemented in hardware or firmware. For this reason, GTP virtual ports may have the following restrictions on traffic they will handle: Protocol Type (PT) field= 1, where GTP encapsulation ports only sup-port GTP, not GTP' (PT field=0); Extension Header flag (E)=0, where no extension headers are supported, Sequence Number flag (S)=0, where no sequence numbers are sup-ported; N-PDU flag (PN)=0; and Message type=255, where Only G-PDU messages, i.e. tunneled user data, is supported in the fast path.")<br><br>Kempf at [0083] ("If a packet either needs encapsulation or arrives encapsulated with nonzero header flags, header extensions, and/or the GTP-U packet is not a G-PDU packet (i.e. it is a GTP-U control packet), the processing must proceed via the gateway's slow path (software) control plane. GTP-C and GTP' packets directed to the gateway's IP address are a result of mis-configuration and are in error. They must be sent to the OpenFlow controller, since these packets are handled by the S-GW-C and P-GW-C control plane entities in the cloud computing system or to the billing entity handling GTP' and not the S-GW-D and P-GW-D data plane switches.")<br><br>Kempf at [0088] ("To support slow path encapsulation, the software control plane on the switch maintains a hash table with keys calculated from the GTP-U TEID. The TEID hash keys are calculated using a suitable hash algorithm with low collision frequency, for |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1082 of 1100

| No. | '111 Patent Claim 16 | Cisco IWAN System |
|---|---|---|
| | | example SHA-1. The flow table entries contain a record of how the packet header, including the GTP encap-sulation header, should be configured. This includes: the same header fields as for the hardware or firmware encapsu-lation table in FIG.18; values for the GTP header flags (PT, E, S, and PN); the sequence number and/or the N-PDU number if any; if the E flag is 1, then the flow table contains a list of the extension headers, including their types, which the slow path should insert into the GTP header.")<br><br>Kempf at [0092] ("In one embodiment, the system implements a GTP fast path decapsulation virtual port. When requested by the S-GW and P-GW control plane software running in the cloud computing system, the gateway switch installs rules and actions for routing GTP encapsulated packets out of GTP tunnels. The rules match the GTP header flags and the GTP TEID for the packet, in the modified OpenFlow flow table shown in FIG. 17 as follows: the IP destination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); the UDP destination port is the GTP-U destination port (2152); and the header fields and message type field is wildcarded with the flag 0XFFF0 and the upper two bytes of the field match the G-PDU message type (255) while the lower two bytes match 0x30, i.e. the packet is a GTP packet not a GTP' packet and the version number is 1.")<br><br>Kempf at [0094] ("In one embodiment, the system implements han-dling of GTP-U control packets. The OpenFlow controller programs the gateway switch flow tables with 5 rules for each gateway switch IP address used for GTP traffic. These rules contain specified values for the following fields: the IP des-tination address is an IP address on which the gateway is expecting GTP traffic; the IP protocol type is UDP (17); the UDP destination port is the GTP-U destination port (2152); the GTP header flags and message type field is wildcarded with 0xFFF0; the value of the header flags field is 0x30, i.e. the version number is 1 and the PT field is 1; and the value of the message type field is one of 1 (Echo Request), 2 (Echo Response), 26 (Error Indication), 31 (Support for Extension Headers Notification), or 254 (End Marker).")<br><br>Kempf at [0098] ("The header flags and message type fields for the three rules are wildcarded with the following bitmasks and match as follows: bitmask 0xFFF4 and the upper two bytes match the G-PDU message type (255) while the lower two bytes are Ox34, indicating that the version number is 1, the packet is a GTP packet, and there is an extension |

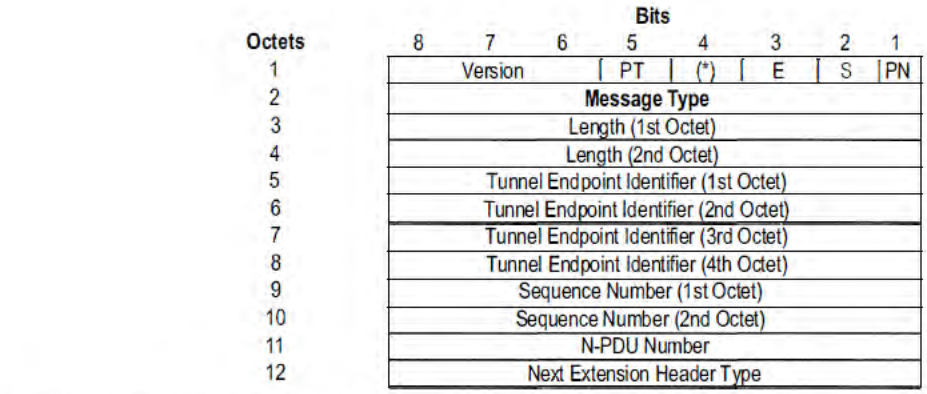


Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1083 of 1100

| No. | '111 Patent Claim 16 | Cisco IWAN System |
|---|---|---|
| | | header present; bitmask 0xFFFF2 and the upper two bytes match the G-PDU message type (255) while the lower two bytes are 0x32, indicating that the version number is 1, the packet is a GTP packet, and there is a sequence number present; and bitmask 0xFF0l and the upper two bytes match the G-PDU message type (255) while the lower two bytes are 0x31, indicating that the version number is 1, the packet is a GTP packet, and a N-PDU is present.")<br><br>Kempf at [0114] ("The gtp_type_n_flags field contains the GTP mes-sage type in the upper 8 bits and the GTP header flags in the lower 8 bits. The gtp_teid field contains the GTP TEID. The gtp_ wildcard field indicates whether the GTP type and flags and TEID should be matched. If the lower four bits are 1, the type and flags field should be ignored, while if the upper four bits are 1, the TEID should be ignored. If the lower bits are 0, the type and fields flag should be matched subject to the flags in the gtp_flag_mask field, while if the upper bits are 0 the TEID should be matched. The mask is combined with the message type and header field of the packet using logical AND; the result becomes the value of the match. Only those parts of the field in which the mask has a 1 value are matched.")<br><br>Kempf at [0117] ("The gtp_type_n_flags field contains the GTP mes-sage type in the upper 8 bits and the GTP header flags in the lower 8 bits. The gtp_teid field contains the GRP TEID. When the value of the oxm_type ( oxm_class+oxm_field is GTP _ MATCH and the HM bit is zero, the flaw's GTP header must match these values exactly. If the HM flag is one, the value contains an ersmt_gtp_match field and an ermst_gtp_mask field, as specified by the OpenF!ow 1.2 specification. We define ermst_gtp_mask field for selecting flows based on the settings of flag bits:<br><br>`struct ermst_gtp_mask {`<br>`  uint32_t gtp_wildcard;`<br>`  uint16_t gtp_flag_mask;`<br>`};`<br><br>Kempf at [0118] ("The gtp_ wildcard field indicates whether the TEID should be matched. If the value is 0xFFFFFFFF, the TEID should be matched and not the flags, if the value is 0x00000000, the flags should be matched and not the TEID. If the gtp_ wildcard indicates |



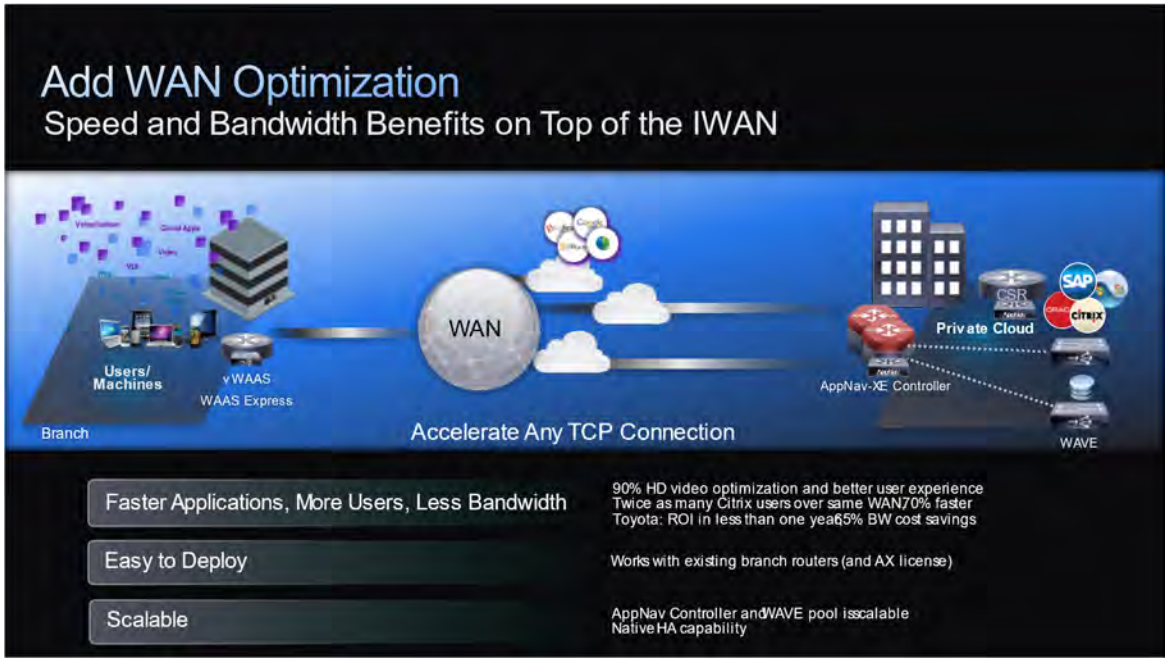
Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1084 of 1100

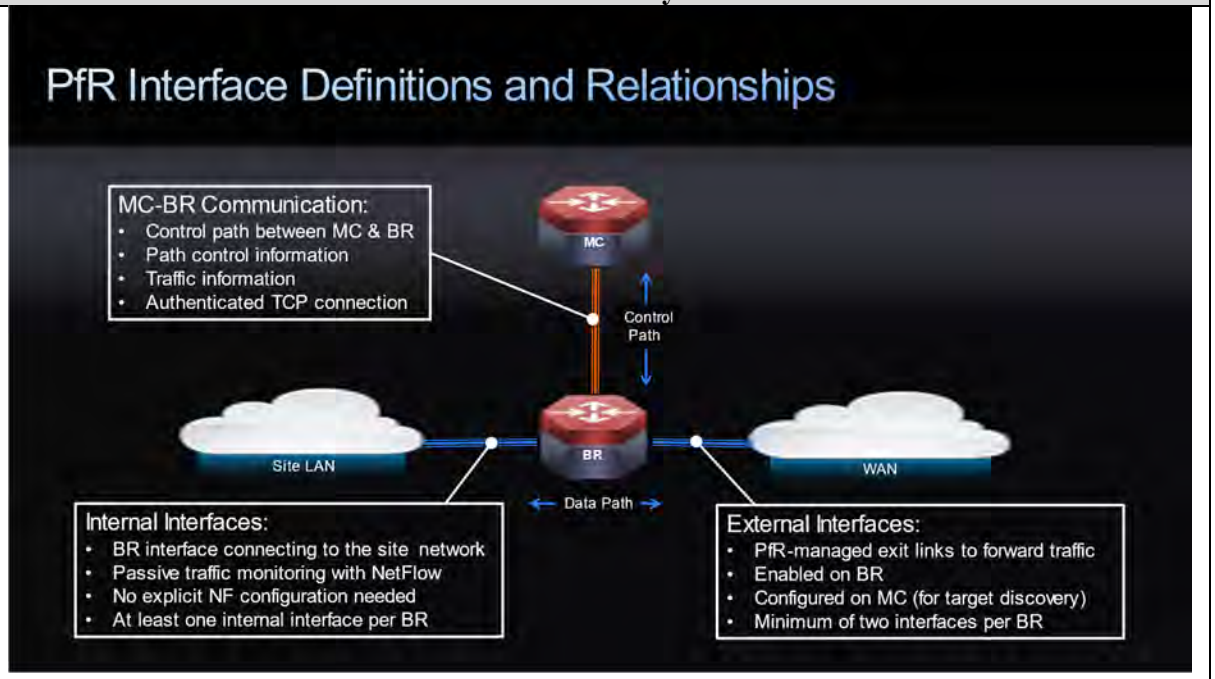
| No. | '111 Patent Claim 16 | Cisco IWAN System |
|---|---|---|
| | | the flags should be matched, the gtp_flag_mask is combined with the message type and header field of the packet using logical AND, the result becomes the value of the match. Only those parts of the field in which the mask has a 1 value are matched.")<br><br>Kempf at Figure 10<br><br>  <br>Bits<br>Octets \| 8 \| 7 \| 6 \| 5 \| 4 \| 3 \| 2 \| 1<br>1 \| Version \| PT \| (*) \| E \| S \| PN<br>2 \| Message Type<br>3 \| Length (1st Octet)<br>4 \| Length (2nd Octet)<br>5 \| Tunnel Endpoint Identifier (1st Octet)<br>6 \| Tunnel Endpoint Identifier (2nd Octet)<br>7 \| Tunnel Endpoint Identifier (3rd Octet)<br>8 \| Tunnel Endpoint Identifier (4th Octet)<br>9 \| Sequence Number (1st Octet)<br>10 \| Sequence Number (2nd Octet)<br>11 \| N-PDU Number<br>12 \| Next Extension Header Type<br>NOTE 0: (*) This bit is a spare bit. It shall be sent as '0'. The receiver shl not evaluate this bit,<br>NOTE 1: 1) This field shall only be evaluated when indicated by the S flag set to 1.<br>NOTE 2: 2) This field shall only be evaluated when indicated by the PN flag set to 1.<br>NOTE 3: 3) This field shall only be evaluated when indicated by the E flag set to 1.<br>NOTE 4: 4) This field shall be present if and only if any one or more of the S. PN and E flags are set.<br><br>FIG. 10 |

| No. | '111 Patent Claim 17 | Cisco IWAN System |
|---|---|---|
| 17[a] | The method according to claim 16, wherein the packet is an Transmission Control | Cisco IWAN System discloses the method according to claim 16, wherein the packet is an Transmission Control Protocol (TCP) packet.<br><br>For example, Cisco IWAN System discloses traffic flows that are part of a TCP connection. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1085 of 1100

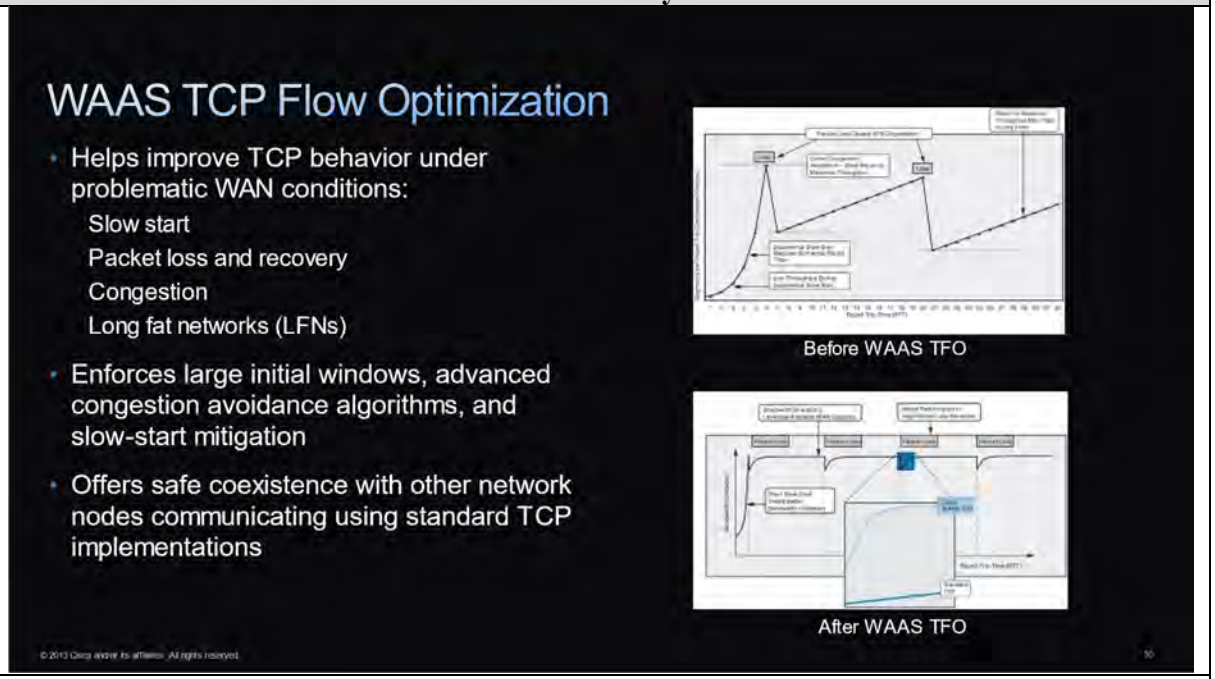
| No. | '111 Patent Claim 17 | Cisco IWAN System |
|---|---|---|
| | Protocol (TCP) packet, and | *See supra* at Claim 16.<br><br>Cisco IWAN<br>  <br>Cisco IWAN - Uncompromised Experience |

| No. | '111 Patent Claim 17 | Cisco IWAN System |
|---|---|---|
| | |  |





Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1087 of 1100

| No. | '111 Patent Claim 17 | Cisco IWAN System |
|---|---|---|
| | |  |
| 17[b] | wherein the one or more flag bits comprises comprise a SYN flag bit, an ACK flag bit, a FIN flag bit, a RST flag bit, or any combination thereof. | On information and belief, the Cisco IWAN System discloses wherein the one or more flag bits comprises comprise a SYN flag bit, an ACK flag bit, a FIN flag bit, a RST flag bit, or any combination thereof. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Cisco IWAN System is found to not meet this limitation, wherein the one or more flag bits comprises comprise a SYN flag bit, an ACK flag bit, a FIN flag bit, a RST flag bit, or any combination thereof would have been obvious to a person having ordinary skill in the art, as explained below.<br><br>Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Cisco IWAN System in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one or more together) of the references identified in element 17[b] of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references. |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1088 of 1100

| No. | '111 Patent Claim 17 | Cisco IWAN System |
|---|---|---|
| | | For example, Copeland discloses TCP packets with flag bits including SYN, ACK, FIN, and R flag bits.<br><br>Copeland at [0081] ("In a TCP/IP datagram 210, the initial data of the IP datagram is the TCP header 230 information. The initial TCP header 230 information includes the 16-bit source and 16-bit destination port numbers. A 32-bit sequence number for the data in the packet follows the port numbers. Following the sequence number is a 32-bit acknowledgement number. If an ACK flag (discussed below) is set, this number is the next sequence number the sender of the packet expects to receive. Next is a 4-bit data offset, which is the number of 32-bit words in the TCP header. A 6-bit reserved field follows.")<br><br>Copeland at [0082] ("Following the reserved field, the next 6 bits are a series of one-bit flags, shown in FIG. 2 as flags U, A, P, R, S, F. The first flag is the urgent flag (U). If the U flag is set, it indicates that the urgent pointer is valid and points to urgent data that should be acted upon as soon as possible. The next flag is the A ( or ACK or "acknowledgment") flag. The ACK flag indicates that an acknowledgment number is valid, and acknowledges that data has been received. The next flag, the push (P) flag, tells the receiving end to push all buffered data to the receiving application. The reset (R) flag is the following flag, which terminates both ends of the TCP connection. Next, the S (or SYN for "synchronize") flag is set in the initial packet of a TCP connection where both ends have to synchronize their TCP buffers. Following the SYN flag is the F (for FIN or "finish") flag. This flag signifies that the sending end of the communication and the host will not send any more data but still may acknowledge data that is received.")<br><br>Copeland at [0089] ("FIG. 3 illustrates an exemplary TCP/IP session 300. As discussed in reference to FIG. 2, the SYN flag is set whenever one host initiates a session with another host. In the initial packet, Hostl sends a message with only the SYN flag set. The SYN flag is designed to establish a TCP connection and allow both ends to synchronize their TCP buffers. Hostl provides the sequence of the first data packet it will send.")<br><br>Copeland at [0090] ("Host2 responds with a SYN-ACK packet. In this message, both the SYN flag and the ACK flag are set. Host2 provides the initial sequence number for its data to Hostl. Host2 also sends to Hostl the acknowledgment number that is the next sequence number Host2 expects to receive from host 1. In the SYN-ACK packet sent by Host2, the |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1089 of 1100

| No. | '111 Patent Claim 17 | Cisco IWAN System |
|---|---|---|
| | | acknowl-edgment number is the initial sequence number of Hostl plus 1, which should be the next sequence number received.")<br><br>Copeland at [0091] ("Hostl responds to the SYN-ACK with a packet with just the ACK flag set. Hostl acknowledges that the next packet of information received from Host2 will be Host2's initial sequence number plus 1. The three-way handshake is complete and data is transferred.")<br><br>Copeland at [0092] ("Host2 responds to ACK packet with its own ACK packet. Host2 acknowledges the data it has received from Hostl by sending an acknowledgment number one greater than its last received data sequence number. Both hosts send packets with the ACK flag set until the session is to end although the P and U flags may also be set, if warranted.")<br><br>Copeland at [0093] ("As illustrated, when Hostl terminates its end of the session, it sends a packet with the FIN and ACK flags set. The FIN flag informs Host2 that Hostl will send no more data. The ACK flag acknowledges the last data received by Hostl by informing Host2 of the next sequence number it expects to receive.")<br><br>Copeland at [0094] ("Host2 acknowledges the FIN packet by sending its own ACK packet. The ACK packet has the acknowledge-ment number one greater than the sequence number of Hostl's FIN-ACK packet. ACK packets are still delivered between the two hosts, except that HOSTl's packets have no data appended to the TCP/IP end of the headers.")<br><br>Copeland at [0095] ("When Host 2 is ready to terminate the session, it sends its own packet with the FIN and ACK flags set. Hostl responds that it has received the final packet with an ACK packet providing to Host2 an acknowledgment number one greater than the sequence number provided in the FIN-ACK packet of Host2.")<br><br>As another example, Uchida discloses the TCP (Transmission Control Protocol) FIN flag, RST flag, and SYN flag.<br><br>Uchida at [0040] ("A flow end can be detected by various methods as below. For example, in one method, a protocol end message is checked. For example, in the TCP (Transmission |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1090 of 1100

| No. | '111 Patent Claim 17 | Cisco IWAN System |
|---|---|---|
| | | Control Protocol), a FIN flag is checked. In this way, the end of communication, that is, the end of a flow using communica-tion, can be detected. In practice, after a FIN flag, communi-cation with an ACK packet is generated in a reverse-direction flow (a flow in which the source and the destination are reversed). Thus, by detecting the ACK flag in the reverse-direction flow after the FIN packet, a flow end can be deter-mined. Further, since the TCP is used in bidirectional com-munication, the forward- and reverse-direction flows can be used as a pair to determine a flow end. Namely, if the end of a flow is detected, a process rule corresponding to the reverse-direction flow of the flow can also be determined to be unnec-essary. Alternatively, a communication end can also be deter-mined when a predetermined time elapses after reception of a SYN packet and a timeout is determined. Still alternatively, a communication end can be determined by reception of a RST packet. These methods will be described in more detail later as specific examples.")<br><br>Uchida at [0050] ("The flow end check unit can use at least one of a TCP (Transmission Control Protocol) FIN flag, RST flag, and SYN flag extracted by the end determination information extraction unit to determine a flow end.")<br><br>Uchida at [0055] ("In the process rule update method, a flow end can be determined by at least one of a TCP (Transmission Control Protocol) FIN flag, RST flag, and SYN flag.")<br><br>Uchida at [0102] ("Next, specific examples 1 to 3 will be described. In the examples 1 to 3, a flow end is determined by combining features of the above individual exemplary embodiments and using TCP (Transmission Control Protocol) flags.")<br><br>Uchida at [0103] ("FIG. 6 is a state transition diagram of TCP connec-tion. "CLOSED" at the top of FIG. 6 represents the end of TCP communication, and portions connected thereto repre-sent states prior to the end of TCP communication. Approxi-mately 2MSL (MSL: Maximum Segment Lifetime) is the maximum amount of time required to reach the above "CLOSED," that is, if the packet forwarding apparatus stands by for approximately 2MSL after both FINs flow, the above "CLOSED" is reached. Thus, after a FIN is confirmed in either direction, if this 2MSL elapses, basically, a communi-cation end can be determined. Even if the state does not change smoothly because of packet loss or the like (for example, even if an ACK packet does not arrive after "CLOS-ING"), a retransmitted packet is forwarded immediately after this 2MSL. Thus, the end of TCP communication can be |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1091 of 1100

| No. | '111 Patent Claim 17 | Cisco IWAN System |
|---|---|---|
| | | determined if a new FIN packet is not received within the time corresponding to the 2MSL and a margin (2MSL+a) at long-est.")<br><br>Uchida at [0104] ("Hereinafter, the description will be made, assuming that a packet forwarding apparatus Cl according to the present invention relays TCP communication between a com-puter (client) Dl 0 and a server D20 that use network configu-rations illustrated in FIG. 7. In the example of FIG. 7, the computer Dl0 belongs to a network represented by 192.168. 0./24 and is set by 192.168.0.10. The server D20 belongs to a network represented by 192.168.1./24 and is set by 192.168. 1.10. As in the case of the OpenFlow controller described in Non-Patent Documents 1 and 2, a control apparatus ( control-ler) Dl is connected to the packet forwarding apparatus Cl via a dedicated channel and manages connection between the two networks. In the following description, the control appa-ratus (controller) Dl controls the packet forwarding appara-tus Cl so that connection from other networks appears as communication from network number 1 (192.168.1.1) of the respective networks (see process rule actions in FIG. 19). In addition, in the present specific example, since FIN packets are monitored, the end determination information extraction unit Cl 7 monitors a protocol stack, including: fields in which the TCP is determined; and the FIN flag in the TCP header.")<br><br>Uchida at [0105] ("FIG. 8 is a flow chart of a flow end determination process using FIN flags. In FIG. 8, steps relating to a timeout determination are added to steps Slll to S116 in the flow chart in FIG. 3. Thus, the flow chart in FIG. 8 includes more detailed steps than the flow chart of FIG. 3. Hereinafter, operations will be described with reference to FIGS. 3, 6, and 8 and FIGS. 9 to 13. In practice, prior to TCP/IP communi-cation, ARP (Address Resolution Protocol) communication is executed, and a process rule may be set in that stage. However, for ease of description, description of the ARP communication will be omitted. The following description will be made based on communication at the TCP/IP level.")<br><br>Uchida at [0106] ("First, the computer Dl0 starts communication with the server D20. For an initial establishment of communica-tion, a packet (SYN) is inputted to the packet forwarding apparatus Cl (start of ACTIVE OPEN through SYN forward-ing in FIG. 6). The packet reception unit Cl0 receives and stores this first packet in the packet storage unit Cll (steps SlOl to S102 in FIG. 3).") |


~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1092 of 1100

| No. | '111 Patent Claim 17 | Cisco IWAN System |
|---|---|---|
| | | Uchida at [0107] ("The packet reception unit C10 notifies the packet process information extraction unit C12 and the end determination information extraction unit C17 of reception of the packet. The packet process information extraction unit C12 refers to the packet storage unit C11 and extracts information such as IP source and destination information that is necessary to search for a process rule (step S103 in FIG. 3). Hereinafter, a process corresponding to steps S103 to S110 in FIG. 3 will be executed.")<br><br>Uchida at [0115] ("Upon receiving a notification that the packet has been received by the packet reception unit Cl 0, the end deter-mination information extraction unit Cl 7 refers to the packet storage unit Cll, monitors a TCP FIN flag, and finds a FIN flag (step S201 in FIG. 8).")<br><br>Uchida at [0116] ("Since a FIN flag is set, the end determination infor-mation extraction unit Cl 7 determines that the packet includes information necessary for determining a flow end. Thus, the end determination information extraction unit Cl 7 extracts information for identifying a process rule to be deleted (the ingress port is 1; the source address is 192.168. 0.10; the destination is 192.168.1.10; and the protocol is TCP (the type is Ox0006)) and stands by until forwarding of the packet. Upon receiving a notification that the packet has been transmitted by the packet forwarding unit C16, the end deter-mination information extraction unit Cl 7 further extracts information for identifying a process rule to be deleted from the packet storage unit Cll. Since the IP address is replaced, the extracted information for identifying a process rule to be deleted represents that the source address is 192.168.1.1; the destination is 192.168.1.1 0; and the protocol is TCP (the type is 0x0006). The information is used for marking of the reverse flow. The end determination information extraction unit Cl 7 notifies the flow end check unit C18 of the notification that the FIN packet has been received and these items of information (step S202 in FIG. 8).")<br><br>Uchida at [0117] ("Upon receiving the above information from the end determination information extraction unit Cl 7, the flow end check unit C18 checks whether or not a FIN flag is set in a predetermined packet header position (step S203). These steps correspond to steps Slll to S114 in FIG. 3.")<br><br>Uchida at [0121] ("Next, after an ACK reply in response to the FIN packet from the computer DlO is forwarded from the server D20 in the same way as the above normal |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1093 of 1100

| No. | '111 Patent Claim 17 | Cisco IWAN System |
|---|---|---|
| | | packet (start of PASSIVE CLOSE in FIG. 6), the server D20 transmits a FIN packet to the computer DlO. When this FIN packet is inputted to the packet forwarding apparatus Cl, the flow end determi-nation process from steps Slll to S116 is started, as in the case of the above start of ACTIVE CLOSE.")<br><br>Uchida at [0122] ("Upon receiving a notification that the packet has been received from the packet reception unit Cl0, the end determination information extraction unit Cl 7 refers to the packet storage unit Cll, monitors a TCP FIN flag, and finds a FIN packet (step S201 in FIG. 8).")<br><br>Uchida at [0123] ("Since a FIN flag is set, the end determination infor-mation extraction unit Cl 7 determines that the packet includes information necessary for determining a flow end. Thus, the end determination information extraction unit Cl 7 extracts information for identifying a process rule to be deleted (the ingress port is 2; the source address is 192.168. 1.10; the destination is 192.168.1.1; and the protocol is TCP (the type is Ox.0006)) and stands by until the packet is trans-mitted. Upon receiving a notification that the packet has been transmitted from the packet forwarding unit C16, the end determination information extraction unit Cl 7 further extracts information for identifying a modified process rule from the packet storage unit Cll. Since the IP address is replaced, the extracted information for identifying a modified process rule represents that the source address is 192.168.1. 10; the destination is 192.168.0.10; and the protocol is TCP (the type is 0x0006). The information is used for marking of the reverse flow. The end determination information extrac-tion unit Cl 7 notifies the flow end check unit C18 of the notification that the FIN packet has been received and these items of information (step S202 in FIG. 8).")<br><br>Uchida at [0124] ("Upon receiving the above information from the end determination information extraction unit Cl 7, the flow end check unit C18 checks whether or not a FIN flag is set in a predetermined packet header position (step S203 in FIG. 8). These steps correspond to steps Slll to S114 in FIG. 3.")<br><br>Uchida at [0125] ("At this point, since a FIN packet has been transmit-ted, the flow end check unit C18 uses the information for identifying a process rule to be deleted as a key, extracts the process rule (process rule corresponding to ingress port 2 in FIG. 11) from the process rule storage unit C13, and marks a FIN packet reception flag (steps S204 to S205 in |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1094 of 1100

| No. | '111 Patent Claim 17 | Cisco IWAN System |
|---|---|---|
| | | FIG. 8). This process corresponds to the internal state update process in step S115 in FIG. 3.")<br><br>Uchida at [0134] ("Referring back to the state transition diagram of TCP connection in FIG. 6, there are two cases where "CLOSED" at the top of FIG. 6 is reached without a state transition involving FIN flags. One case arises when the ses-sion is closed from SYN_SENT, which is reached when a SYN packet in which a SYN flag is marked is transmitted. The other case arises when a timeout is generated. In such case, while the packet forwarding apparatus cannot monitor the closed session, the packet forwarding apparatus can con-firm a timeout in the following way. In the present specific example, a flow end is determined by this timeout.")<br><br>Uchida at [0135] ("n the present specific example, if a SYN/ ACK packet does not flow in a direction opposite to the SYN packet flow direction within a predetermined time (from "SYN_ RCVD" to "SYN_SENT" in FIG. 6), a timeout is determined.")<br><br>Uchida at [0136] ("FIG. 14 is a flow chart illustrating a flow end deter-mination process using a SYN flag. Since the basic operations are the same as those of the above specific example 1, the following description will be made with a focus on the dif-ference.")<br><br>Uchida at [0137] ("In FIG. 14, upon receiving a notification that the packet has been received by the packet reception unit ClO, the end determination information extraction unit Cl 7 refers to the packet storage, unit Cll, monitors a TCP SYN flag, and finds a SYN packet (step S301 in FIG. 14).")<br><br>Uchida at [0138] ("Since a SYN flag is set, the end determination infor-mation extraction unit Cl 7 determines that the packet includes information necessary for determining a flow end. Thus, the end determination information extraction unit Cl 7 extracts information for identifying a process rule to be deleted (the ingress port is 2; the source address is 192.168. 1.10; the destination is 192.168.1.1; and the protocol is TCP (the type is Ox.0006)) and stands by until the packet is trans-mitted. Upon receiving a notification that the packet has been transmitted by the packet forwarding unit C16, the end deter-mination information extraction unit Cl 7 further extracts information for identifying a modified process rule from the packet storage unit Cll. Since the IP address is replaced, the extracted information for |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1095 of 1100

| No. | '111 Patent Claim 17 | Cisco IWAN System |
|---|---|---|
| | | identifying a process rule repre-sents that the source address is 192.168.1.10; the destination is 192.168.0.10; and the protocol is TCP (the type is 0x0006). The information is used for marking of the reverse flow. The end determination information extraction unit Cl 7 notifies the flow end check unit C18 of the notification that the SYN packet has been received and these items of information (step S302 in FIG. 14).")<br><br>Uchida at [0139] ("Upon receiving the above information from the end determination information extraction unit Cl 7, the flow end check unit C18 checks whether a SYN flag is set in a prede-termined packet header position and an ACK flag is not marked (step S303 in FIG. 14). These steps correspond to steps Slll to S114 in FIG. 3.")<br><br>Uchida at [0148] (" Next, a third specific example in which a flow end determination is executed by using a TCP RST (reset) flag will be described.")<br><br>Uchida at [0149] ("Referring back to the state transition diagram of TCP connection in FIG. 6, there is a transition from "SYN_ RCVD," which is a communication establishment standby state, to "LISTEN," which is a communication standby state. A TCP RST (reset) flag signifies release of connection and retry of communication. Namely, since a RST packet in which this RST flag is set signifies invalidation of communi-cation, by detecting this RST flag, a flow end can be deter-mined.")<br><br>Uchida at [0150] ("FIG. 16 is a first flow chart illustrating a flow end determination process using a RST flag. Since the basic operations are the same as those of the above specific example 1, the following description will be made with a focus on the difference.")<br><br>Uchida at [0151] ("In FIG. 16, upon receiving a notification that the packet has been received by the packet reception unit ClO, the end determination information extraction unit Cl 7 refers to the packet storage unit Cll, monitors a TCP RST flag, and finds a RST packet (step S401 in FIG. 16).")<br><br>Uchida at [0152] ("Since a RST flag is set, the end determination infor-mation extraction unit Cl 7 determines that the packet includes information necessary for determining a flow end. Thus, the end determination information extraction unit Cl 7 extracts information for identifying a process rule to be deleted (the ingress port is 2; the source address is 192.168. |



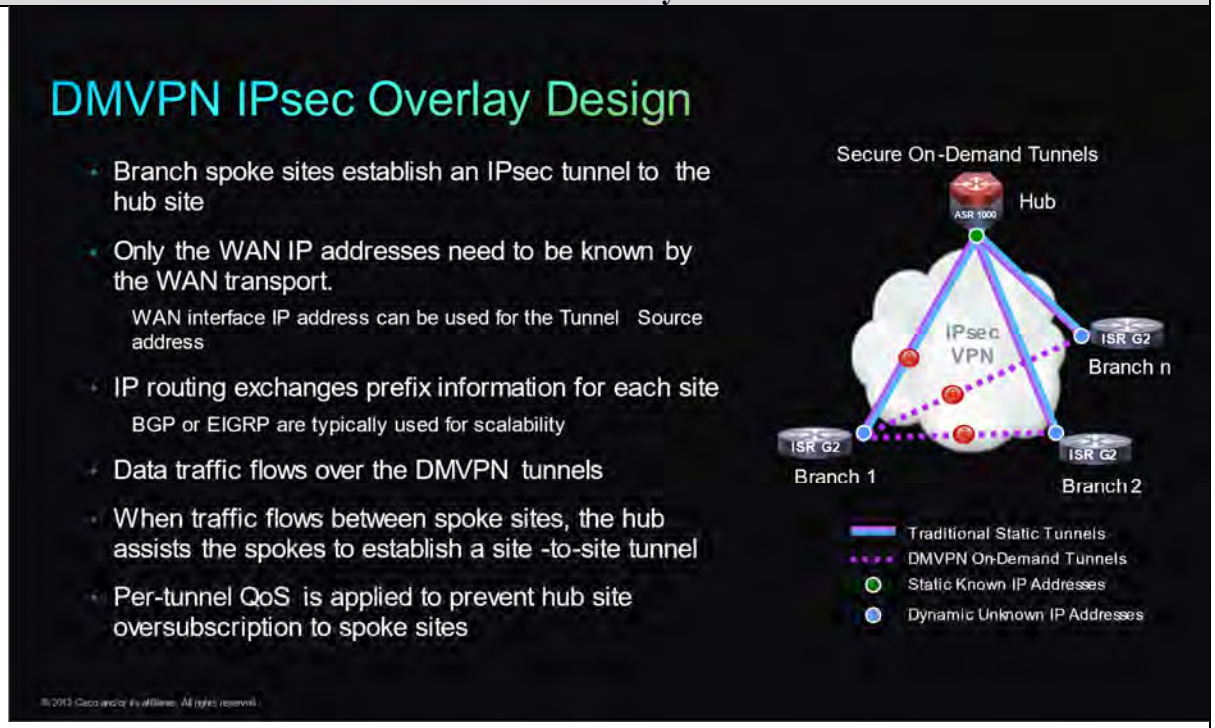
Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1096 of 1100

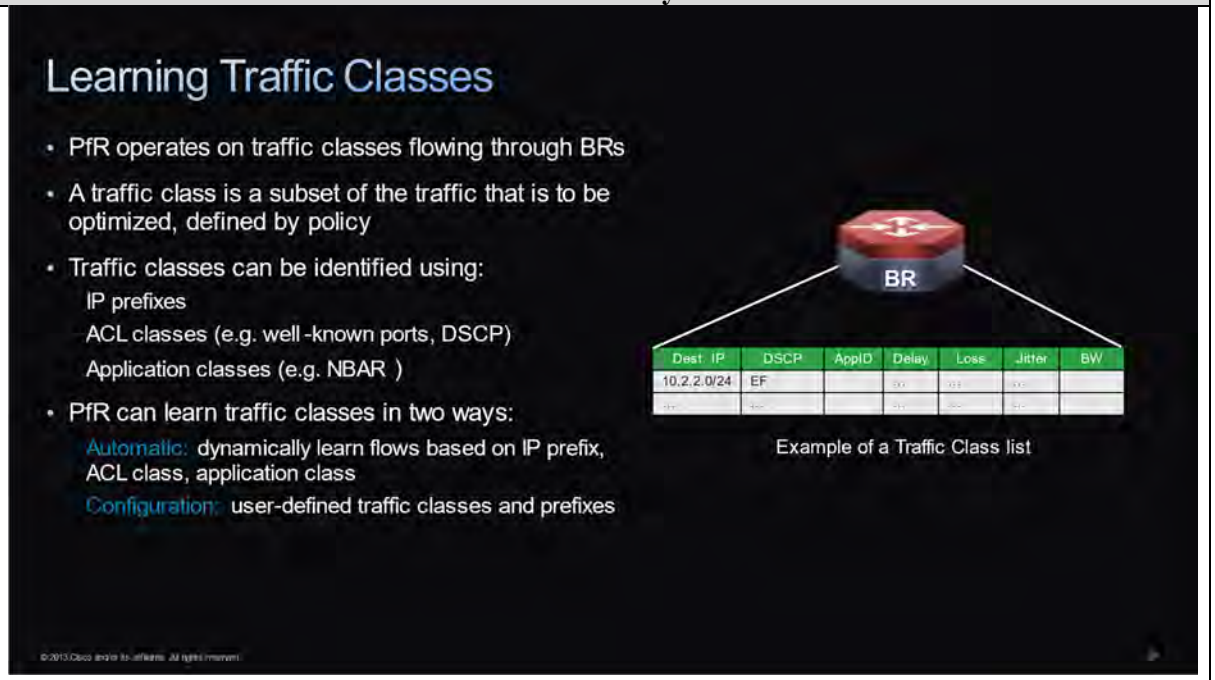
| No. | '111 Patent Claim 17 | Cisco IWAN System |
|---|---|---|
| | | 1.10; the destination is 192.168.1.1; and the protocol is TCP (the type is Ox0006)) and stands by until the packet is trans-mitted. Upon receiving a notification that the packet has been transmitted from the packet forwarding unit C16, the end determination information extraction unit Cl 7 notifies the flow end check unit C18 of the notification that the RST packet has been received and these items of information ( step S402 in FIG. 16).")<br><br>Uchida at [0164] ("For example, in a specific example of the present invention, certain TCP flags are monitored. A single packet forwarding apparatus can monitor these flags in a parallel fashion. For example, after a packet that triggers a flow end is detected, the above process may be allowed to branch to the above FIGS. 8, 14, and 16 (17) to realize parallel monitoring.") |

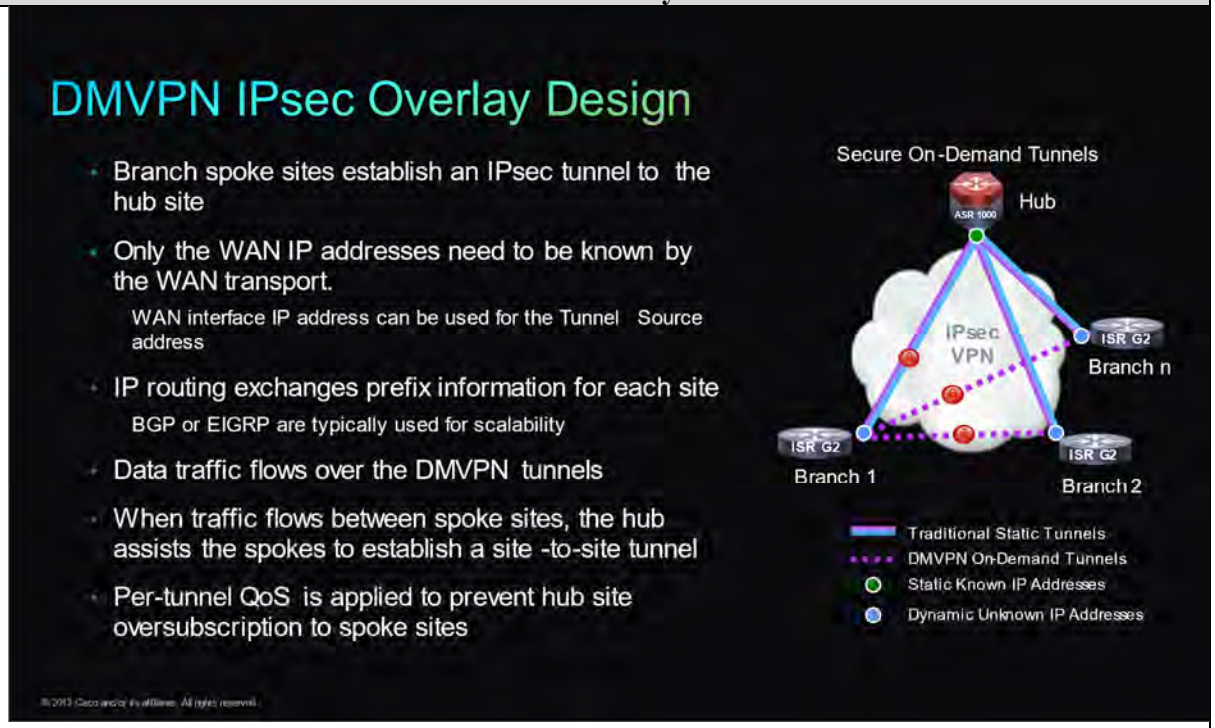
| No. | '111 Patent Claim 18 | Cisco IWAN System |
|---|---|---|
| 18[a] | The method according to claim 1, wherein the packet comprises distinct header and payload fields, | Cisco IWAN System discloses the method according to claim 1, wherein the packet comprises distinct header and payload fields.<br><br>*See supra* at Claim 1, 15[a]. |
| 18[b] | the header comprises at least the first and second entities addresses in the packet network, and | Cisco IWAN System discloses the header comprises at least the first and second entities addresses in the packet network.<br><br>For example, Cisco IWAN System discloses data traffic with IP addresses to forward the traffic flow.<br><br>Cisco IWAN |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1097 of 1100

| No. | '111 Patent Claim 18 | Cisco IWAN System |
|---|---|---|
| | |  <br>Cisco IWAN - Uncompromised Experience |

| No. | '111 Patent Claim 18 | Cisco IWAN System |
|---|---|---|
| | |  |
| 18[c] | wherein the packet-applicable criterion is that the first entity address, the second entity address, or both match a predetermined address or addresses. | Cisco IWAN System discloses wherein the packet-applicable criterion is that the first entity address, the second entity address, or both match a predetermined address or addresses.<br><br>For example, Cisco IWAN System discloses IP prefixes, such as IP addresses, as an identifying characteristic of traffic classes in a traffic policy.<br><br>Cisco IWAN |

| No. | '111 Patent Claim 18 | Cisco IWAN System |
|---|---|---|
| | |  <br>Cisco IWAN - Uncompromised Experience |



Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00714, Page 1100 of 1100