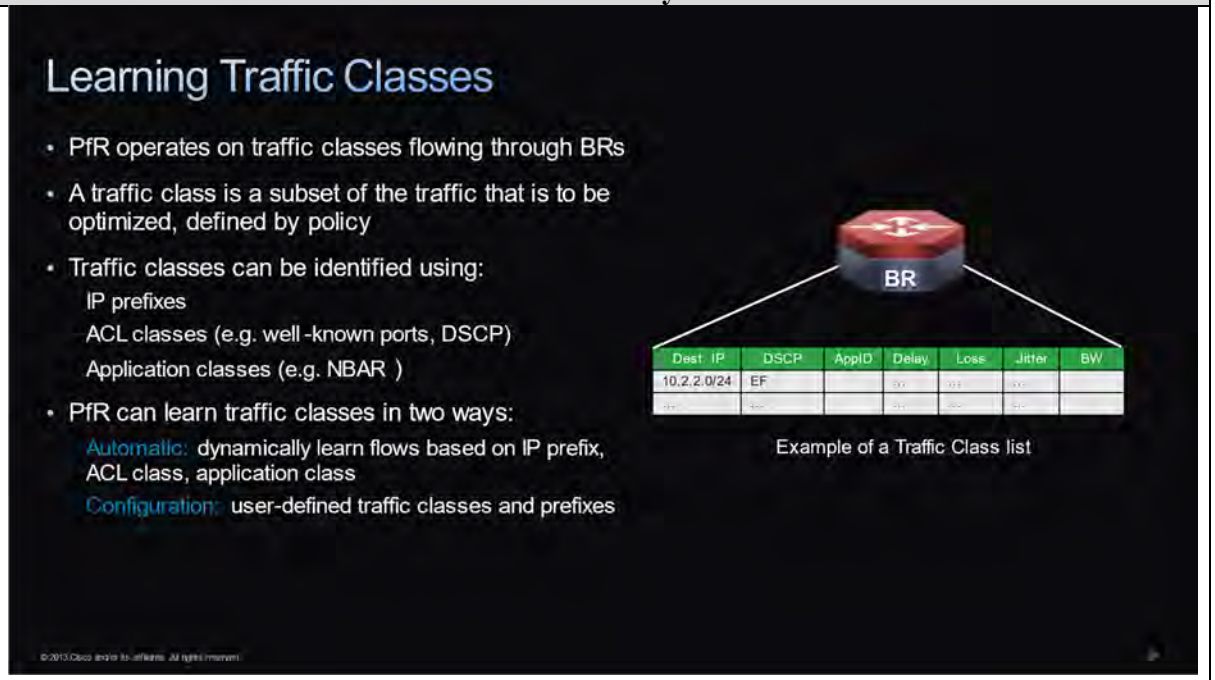
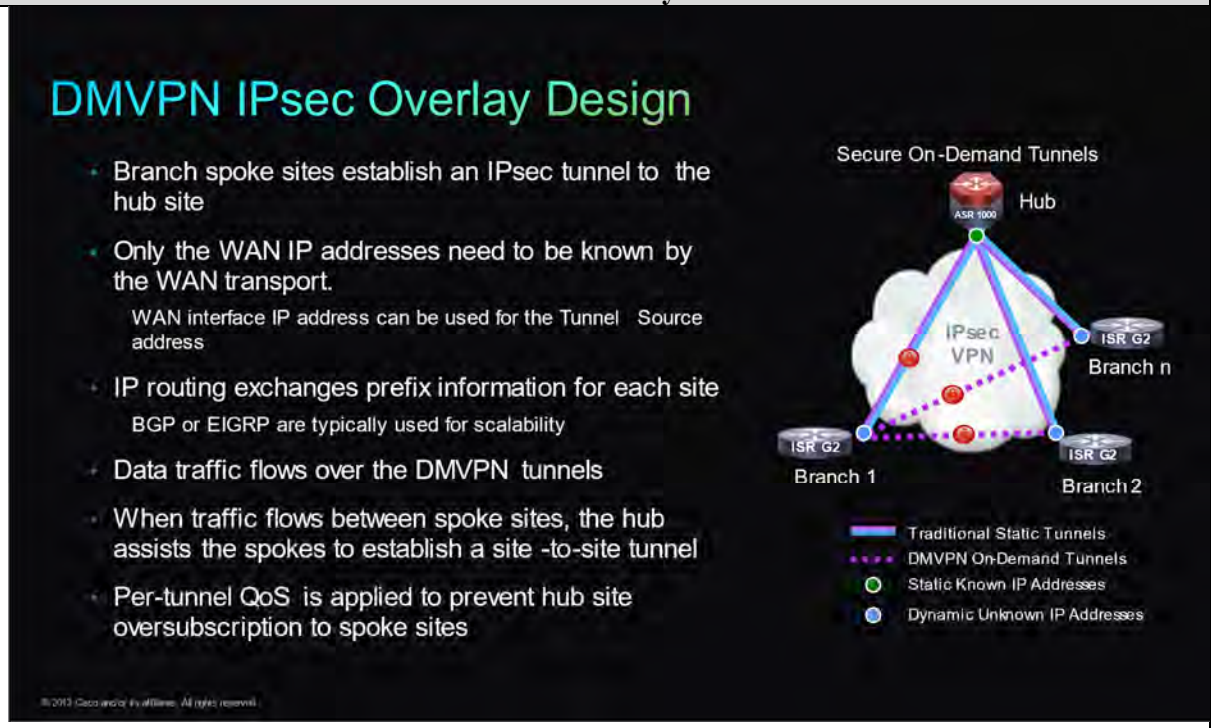
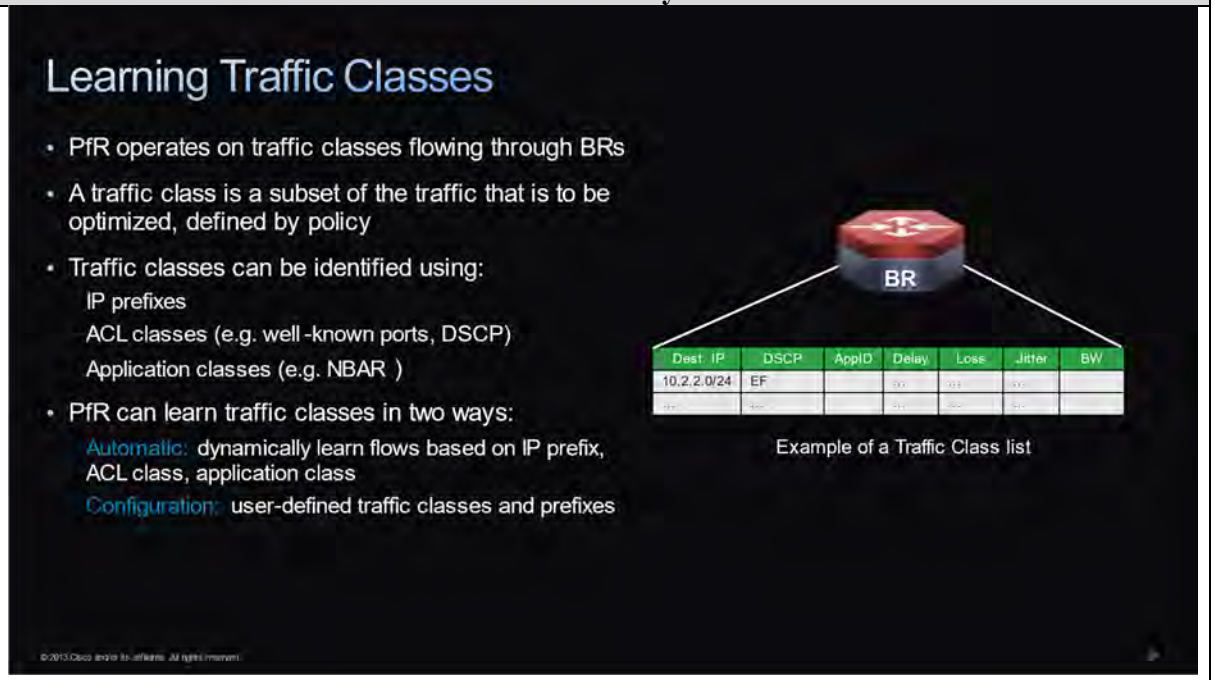
No.	'111 Patent Claim 18	Cisco IWAN System
		

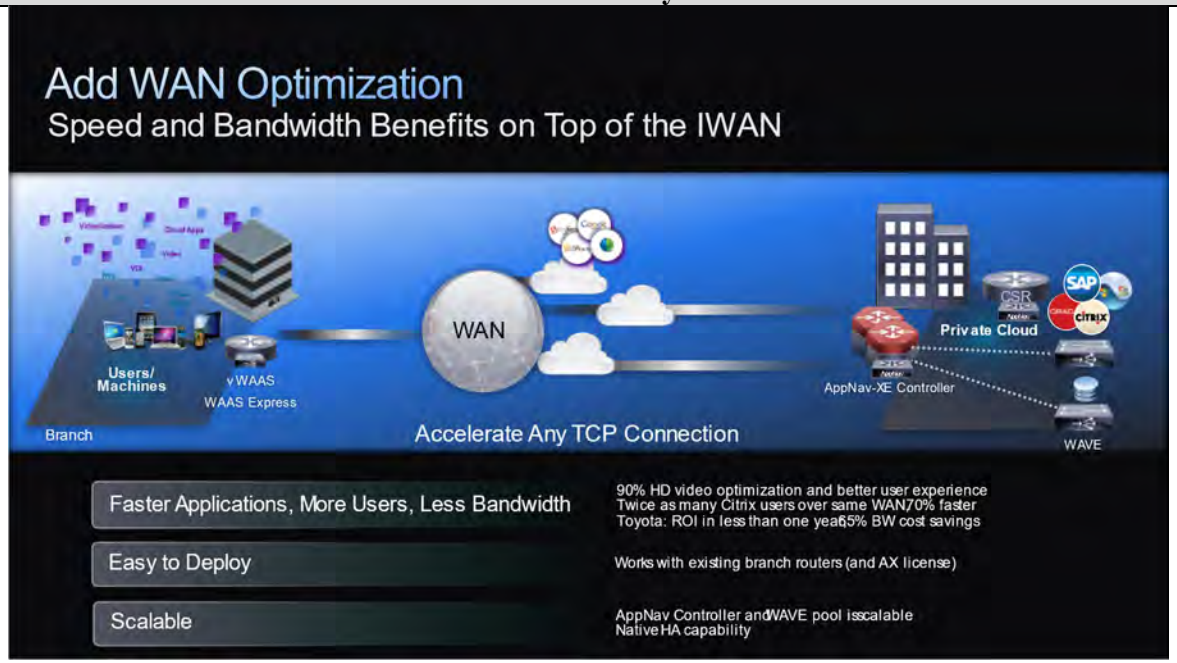
No.	'111 Patent Claim 19	Cisco IWAN System
19	The method according to claim 18, wherein the addresses are Internet Protocol (IP) addresses.	Cisco IWAN System discloses the method according to claim 18, wherein the addresses are Internet Protocol (IP) addresses. For example, Cisco IWAN System discloses IP addresses used for forwarding traffic flows. *See supra* at Claim 18. Cisco IWAN

No.	'111 Patent Claim 19	Cisco IWAN System
		 Cisco IWAN - Uncompromised Experience

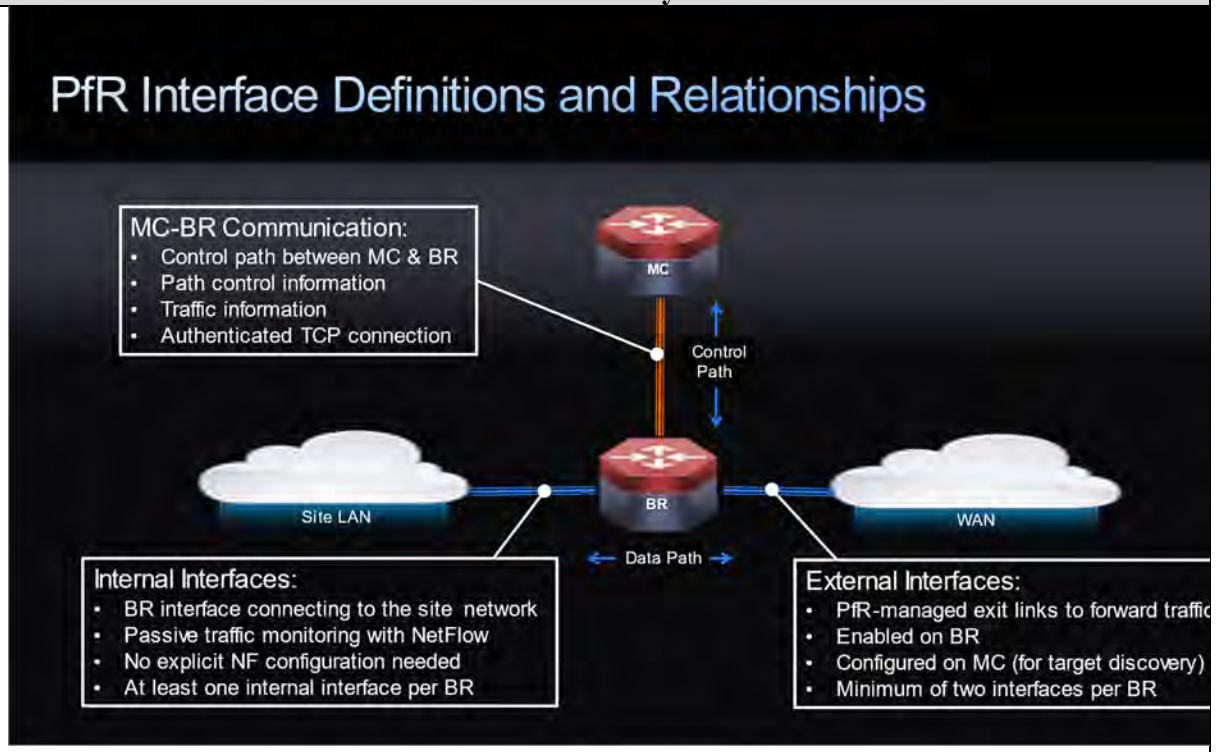
No.	'111 Patent Claim 19	Cisco IWAN System
		

No.	'111 Patent Claim 20	Cisco IWAN System
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	Cisco IWAN System 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. For examples, Cisco IWAN System discloses traffic flows sent over a TCP connection, which include TCP information and characteristics. *See supra* at Claim 1, 17[a]. Cisco IWAN

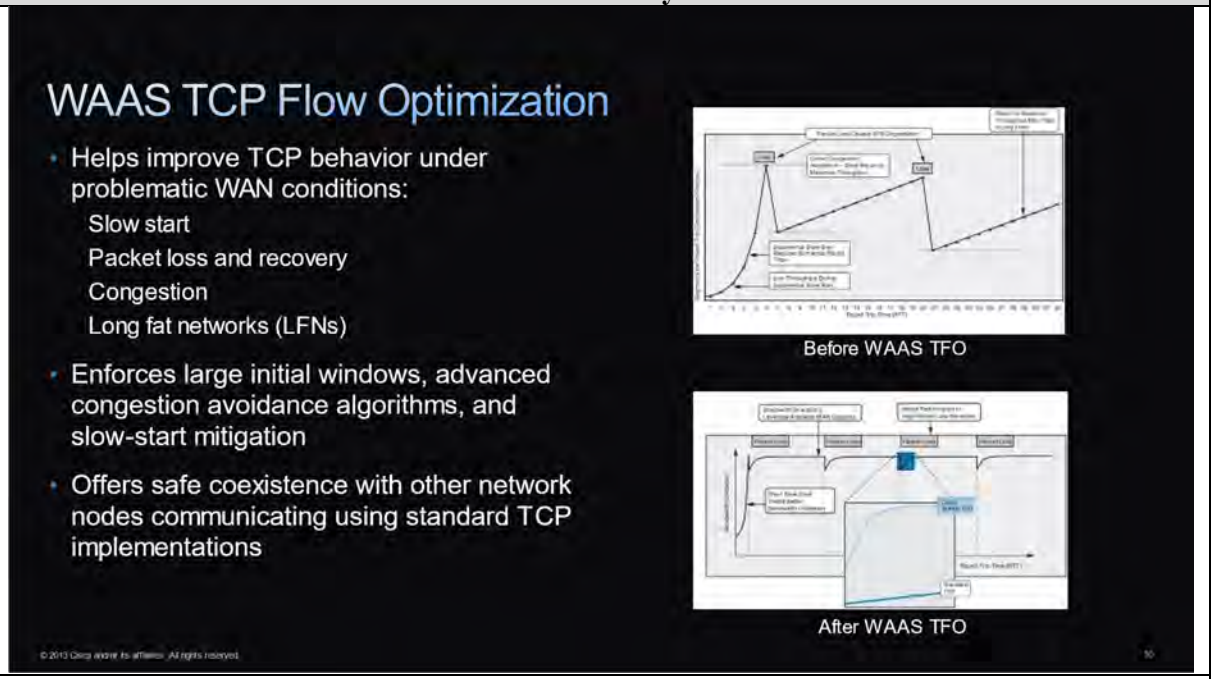
Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1103 of 2137

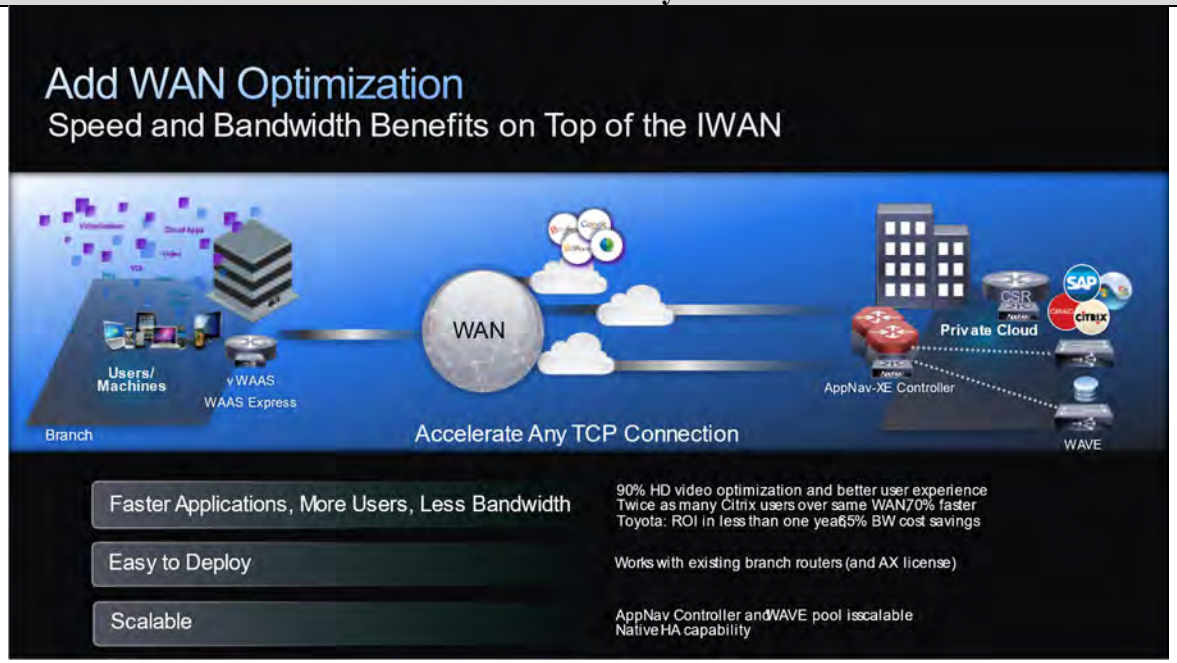
No.	'111 Patent Claim 20	Cisco IWAN System
		 Cisco IWAN - Uncompromised Experience

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1104 of 2137

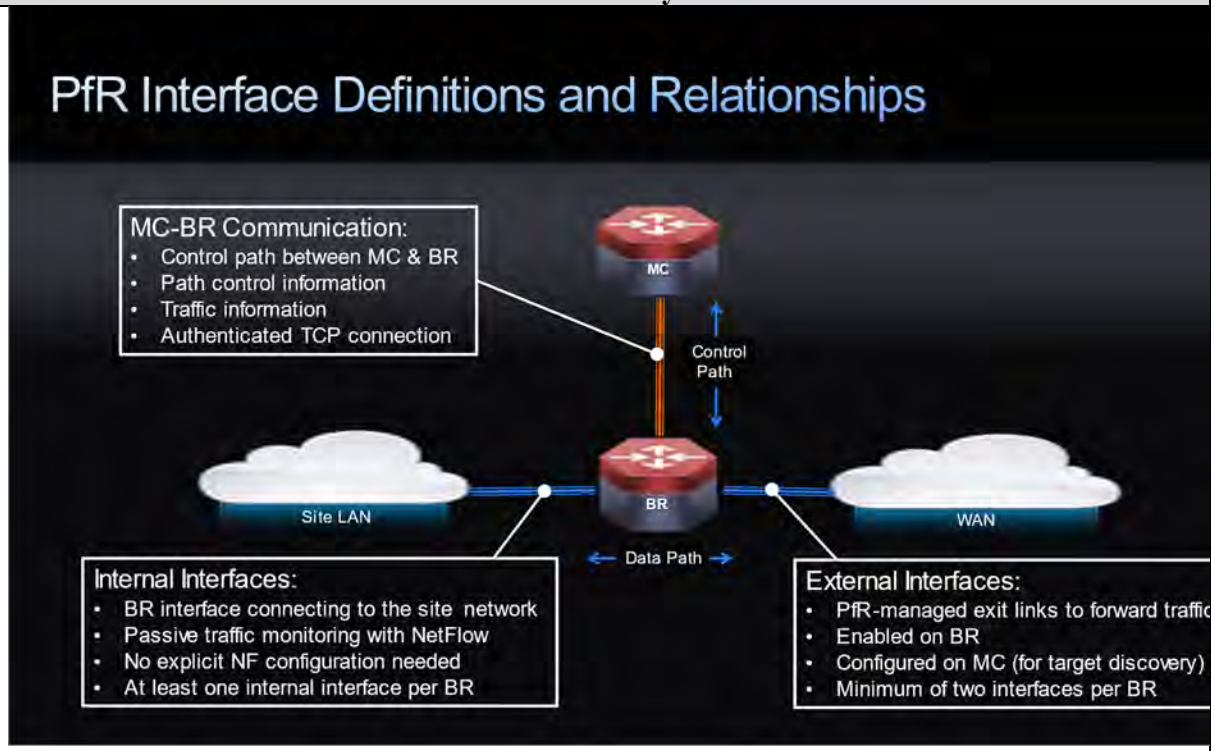
No.	'111 Patent Claim 20	Cisco IWAN System
		

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1105 of 2137

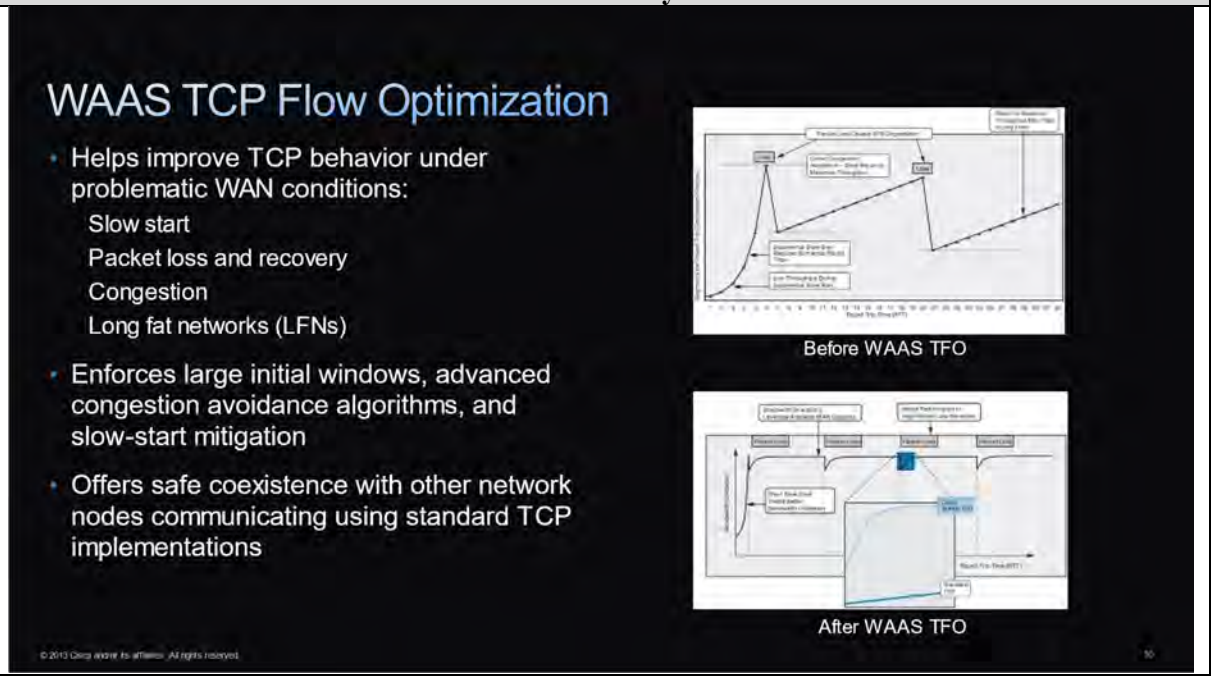
No.	'111 Patent Claim 20	Cisco IWAN System
		
20[b]	wherein the packet-applicable criterion is that the source TCP port, the destination TCP port, the TCP sequence number, the TCP sequence mask, or any combination thereof, matches a predetermined value or values.	Cisco IWAN System discloses wherein the 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. For example, Cisco IWAN System discloses identifying characteristics of traffic classes, including prefixes and well-known ports. A person of ordinary skill in the art would understand that these identifying characteristics may be of TCP traffic flows and include the source TCP port, the destination TCP port, the TCP sequence number, the TCP sequence mask, or any combination thereof, which may be defined in the traffic policy. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. Cisco IWAN

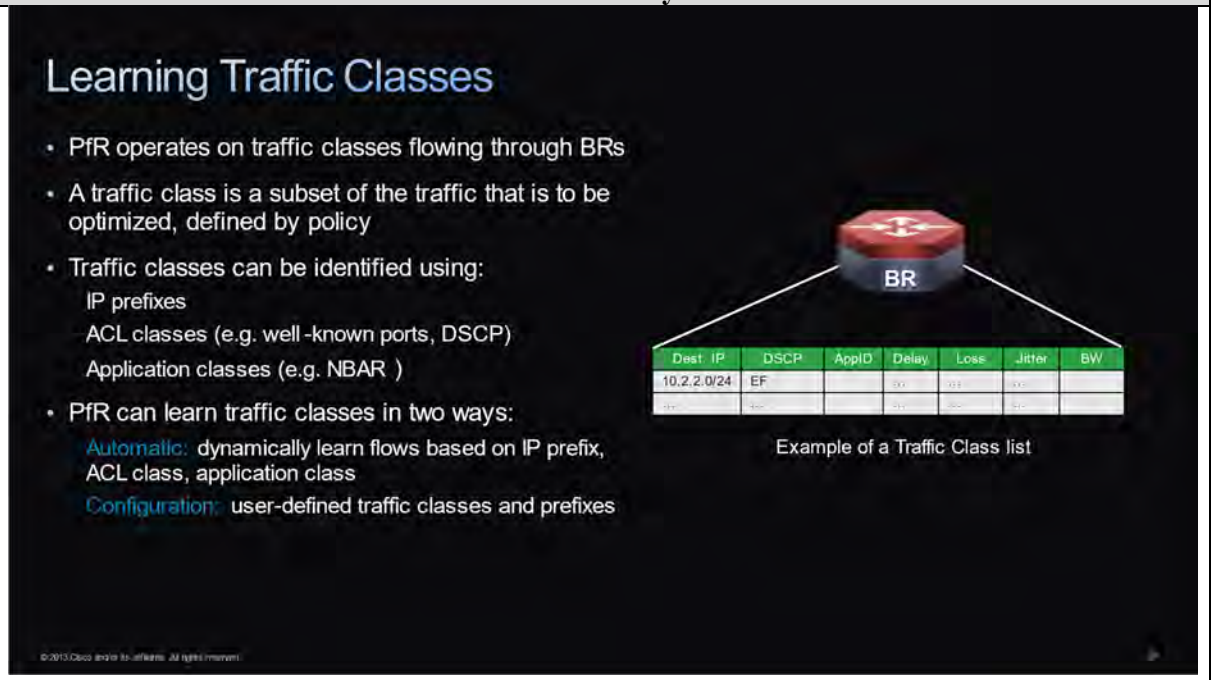
No.	'111 Patent Claim 20	Cisco IWAN System
		 Cisco IWAN - Uncompromised Experience

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1107 of 2137

No.	'111 Patent Claim 20	Cisco IWAN System
		

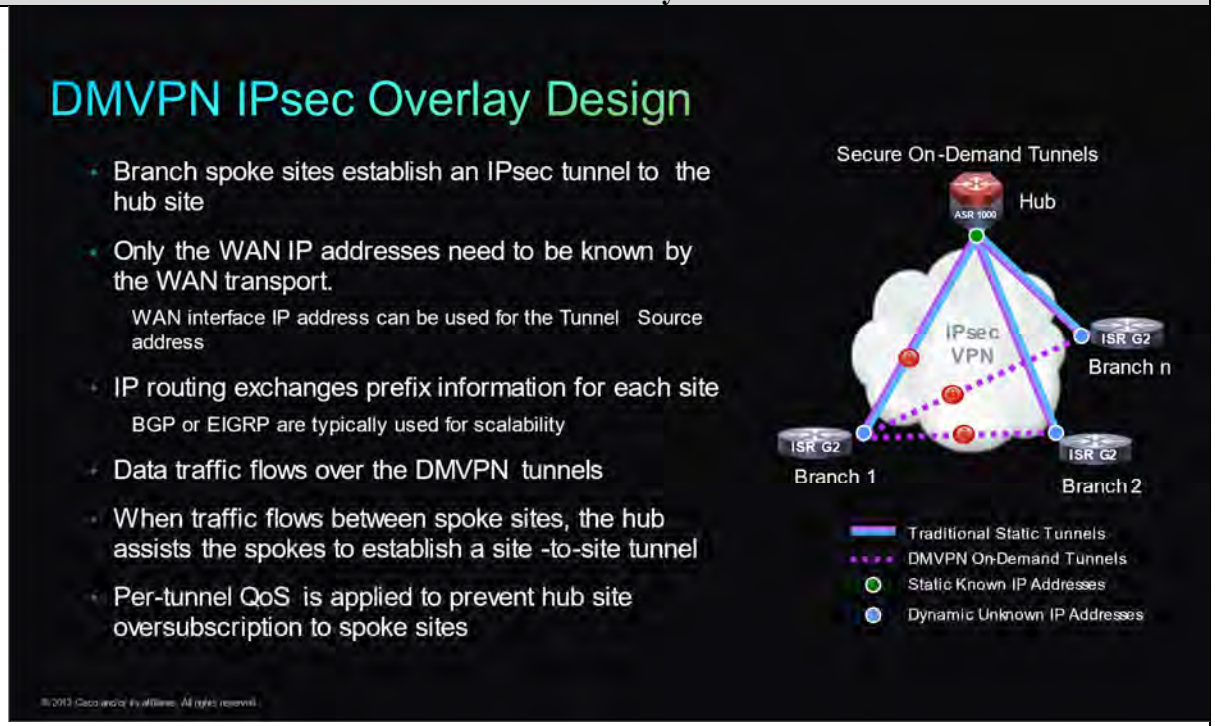
Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1108 of 2137

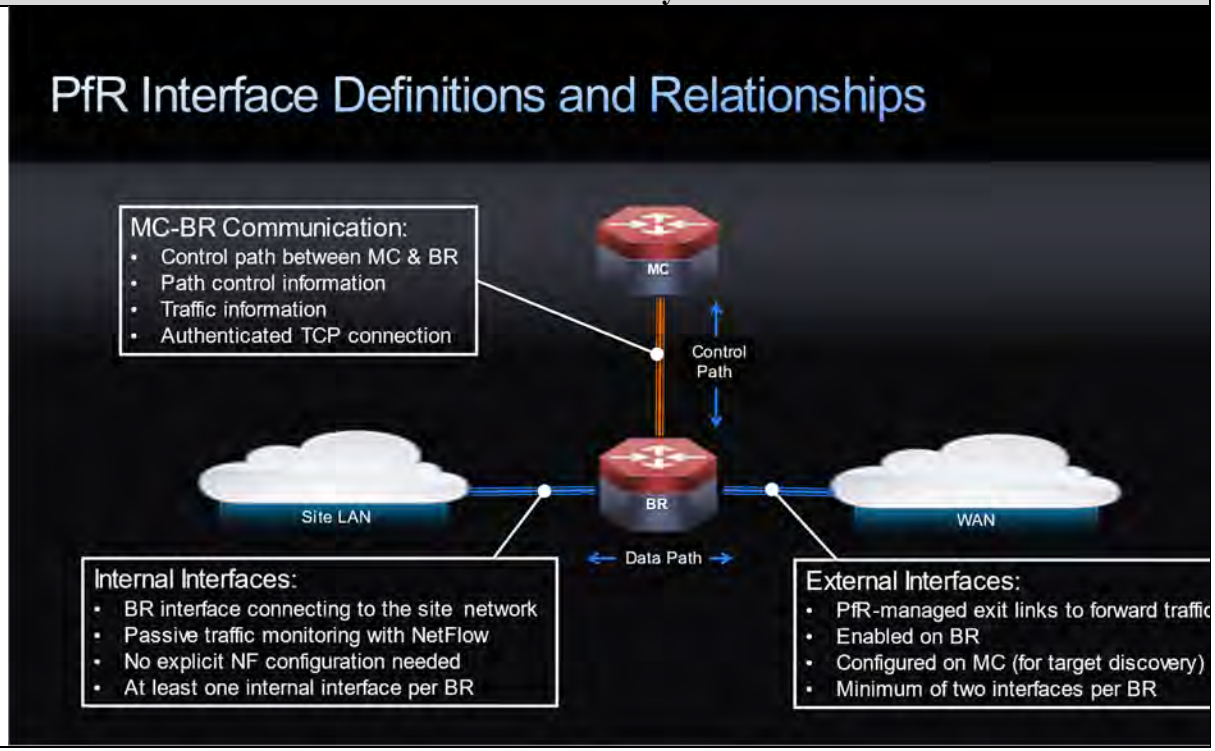
No.	'111 Patent Claim 20	Cisco IWAN System
		

No.	'111 Patent Claim 20	Cisco IWAN System
		

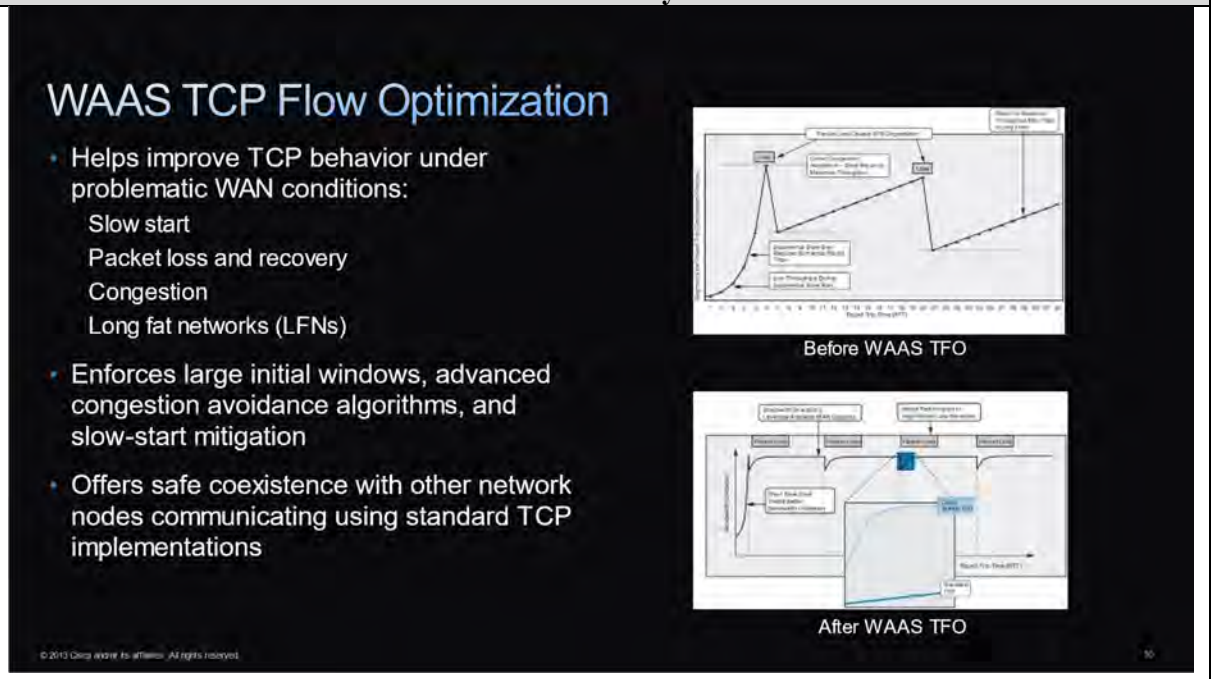
No.	'111 Patent Claim 21	Cisco IWAN System
21	The method according to claim 1, wherein the packet network comprises a Wide Area Network (WAN), Local Area Network (LAN), the Internet, Metropolitan Area Network (MAN), Internet Service Provider (ISP) backbone datacenter	Cisco IWAN System discloses the method 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. For example, Cisco IWAN System discloses communicating in a packet network that may be a wide area network, local area network, or data center network. *See supra* at Claim 1. Cisco IWAN

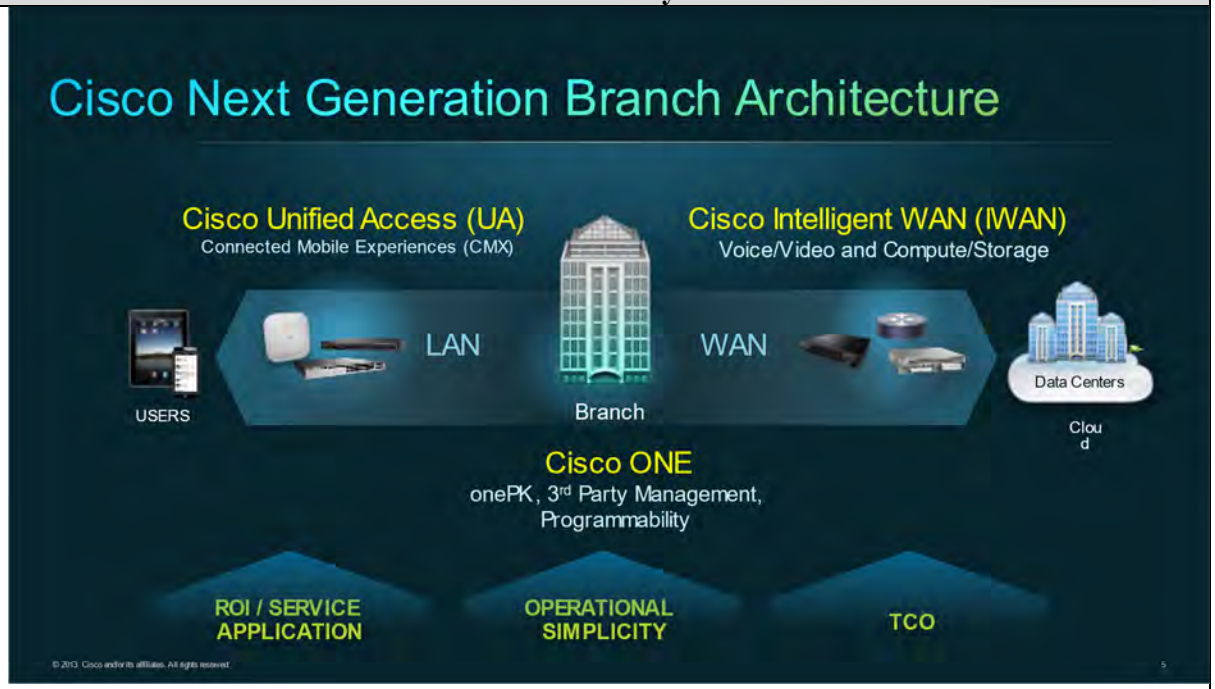
Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1110 of 2137

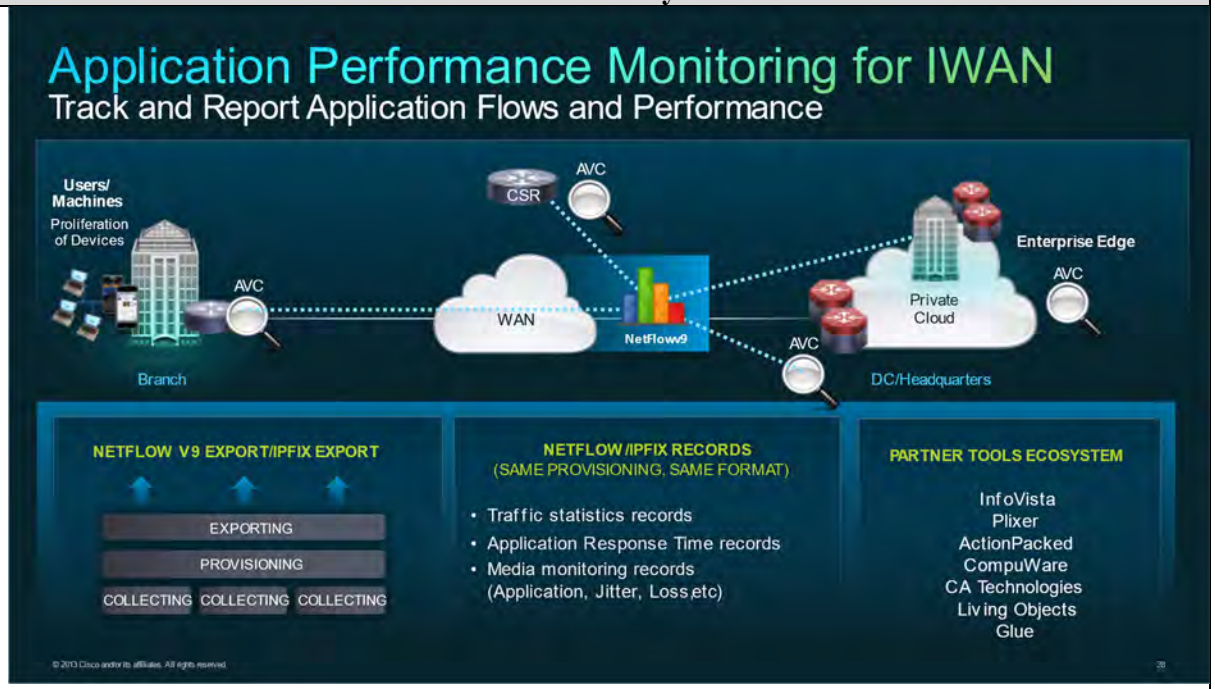
No.	'111 Patent Claim 21	Cisco IWAN System
	network, or inter - datacenter network.	 Cisco IWAN - Uncompromised Experience

No.	'111 Patent Claim 21	Cisco IWAN System
		

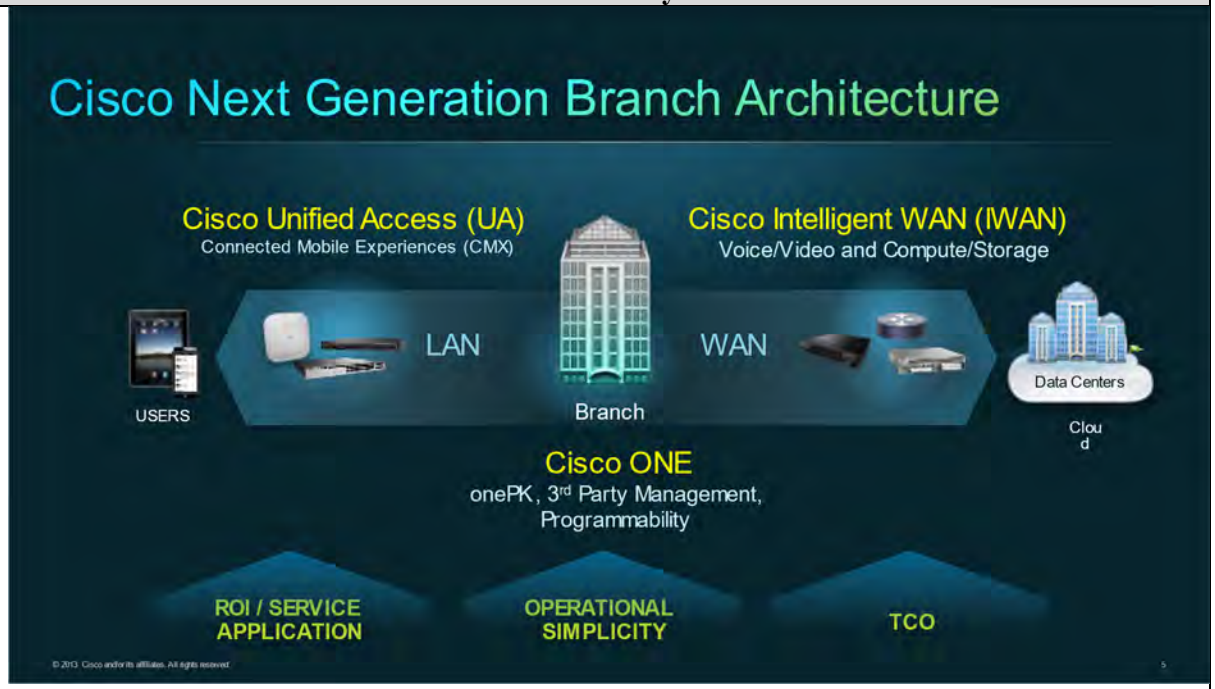
Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1112 of 2137

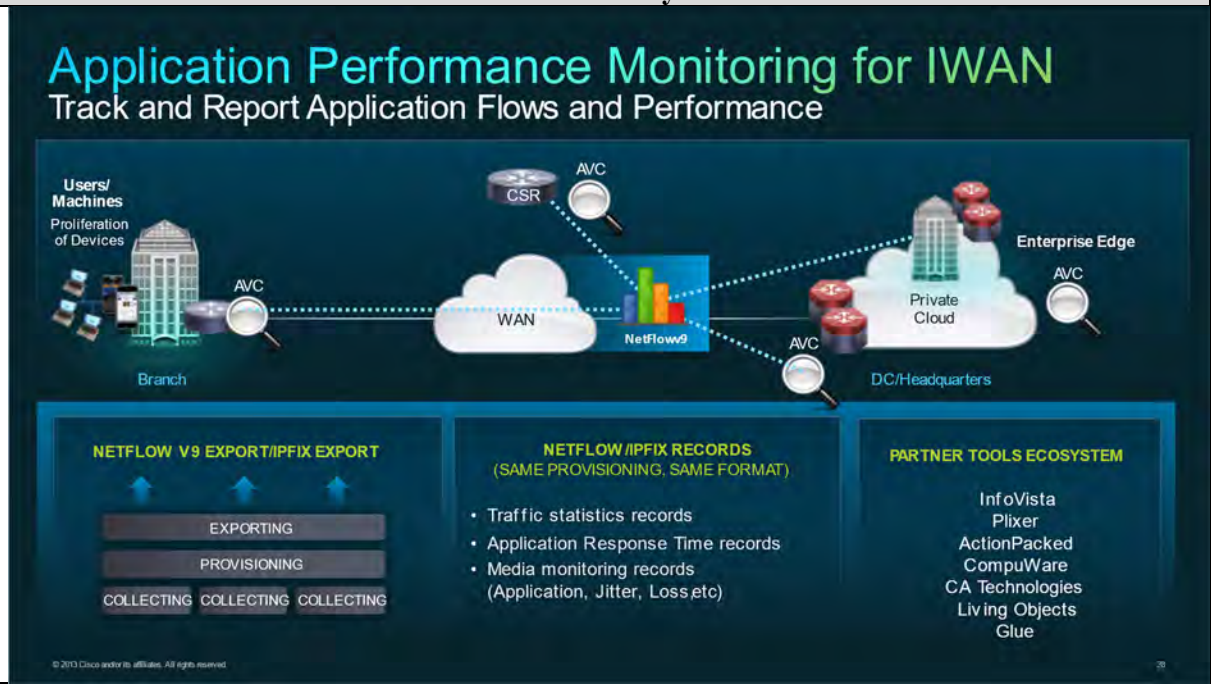
No.	'111 Patent Claim 21	Cisco IWAN System
		 IWAN At-A-Glance

No.	'111 Patent Claim 21	Cisco IWAN System
		

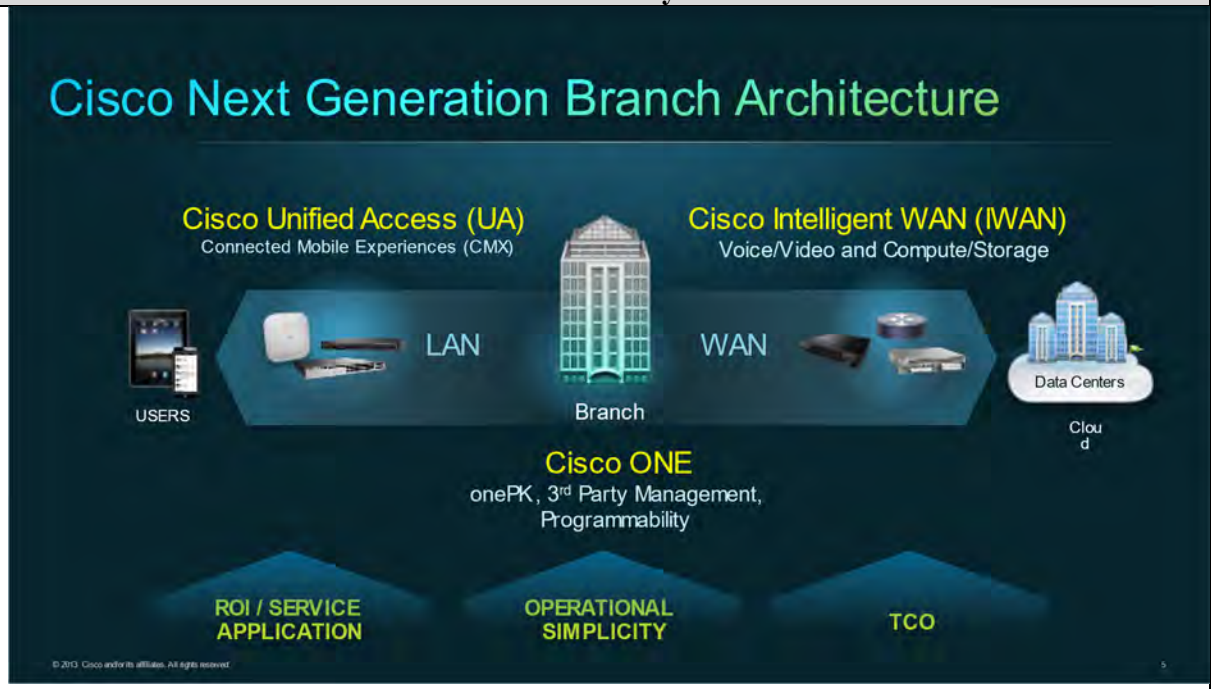
No.	'111 Patent Claim 21	Cisco IWAN System
		

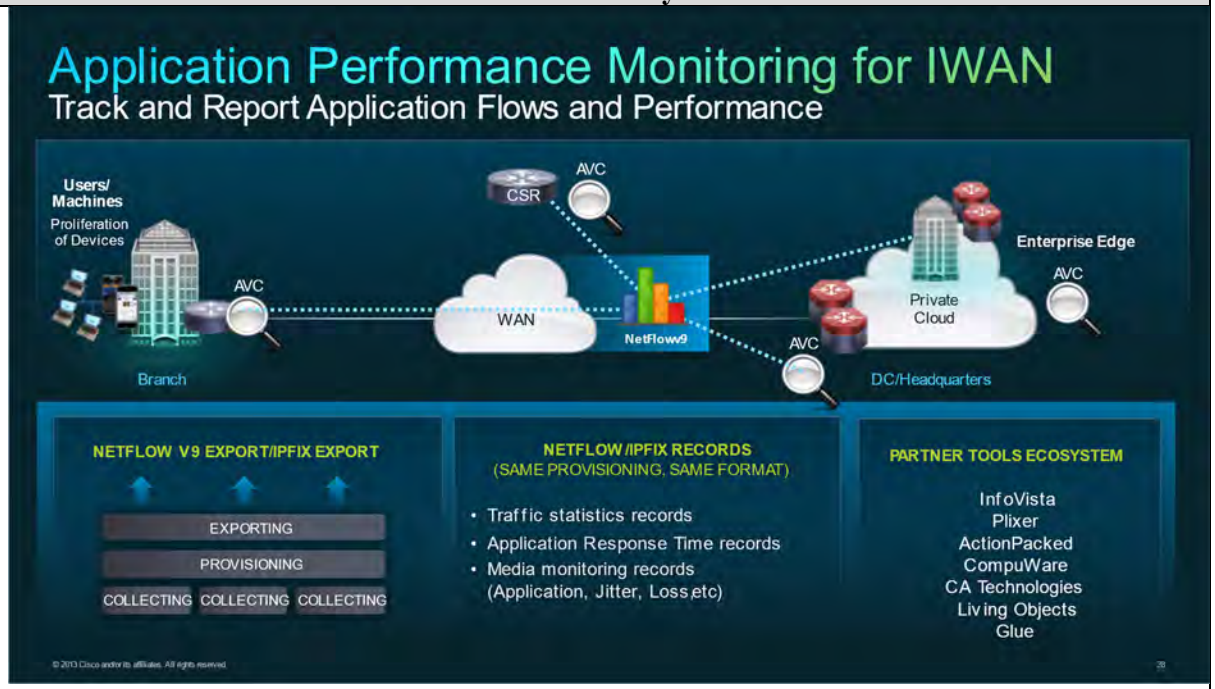
No.	'111 Patent Claim 22	Cisco IWAN System
22	The method according to claim 1, wherein the first entity is a server device and the second entity is a client device, or wherein the first entity is a client device and the second entity is a server device..	Cisco IWAN System discloses the method according 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. For example, Cisco IWAN System discloses data center devices, such as a server, and user devices, which may communicate traffic flows to each other. *See supra* at Claim 1. IWAN At-A-Glance

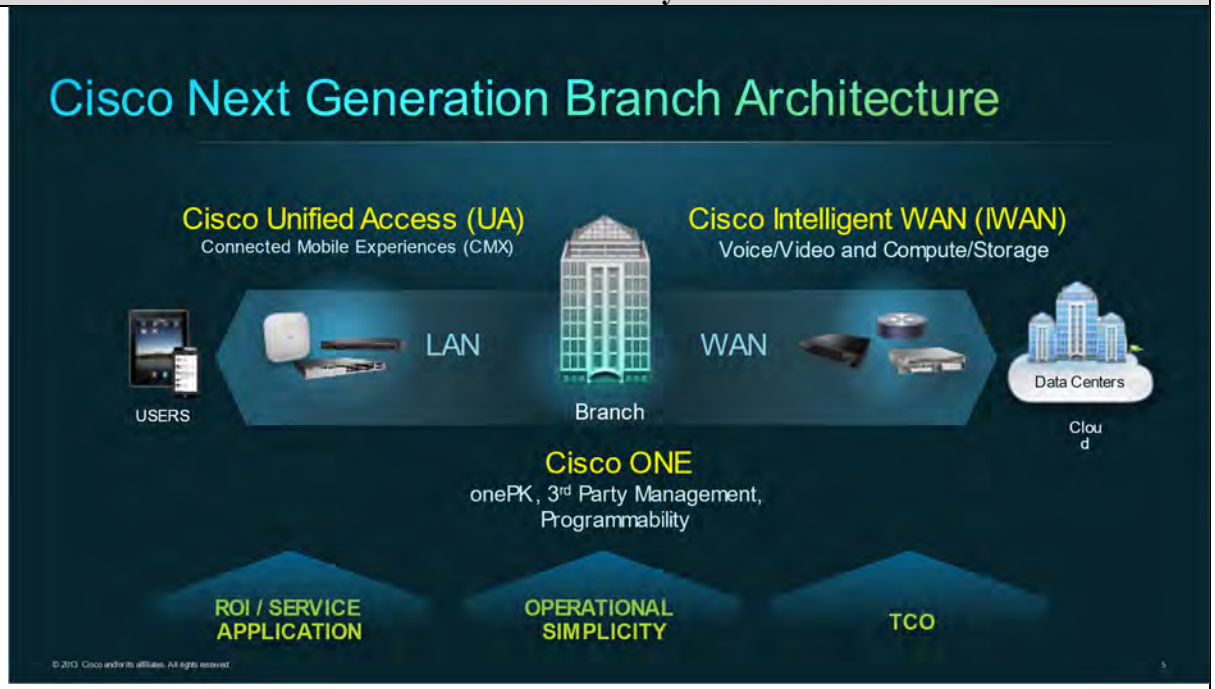
No.	'111 Patent Claim 22	Cisco IWAN System
		

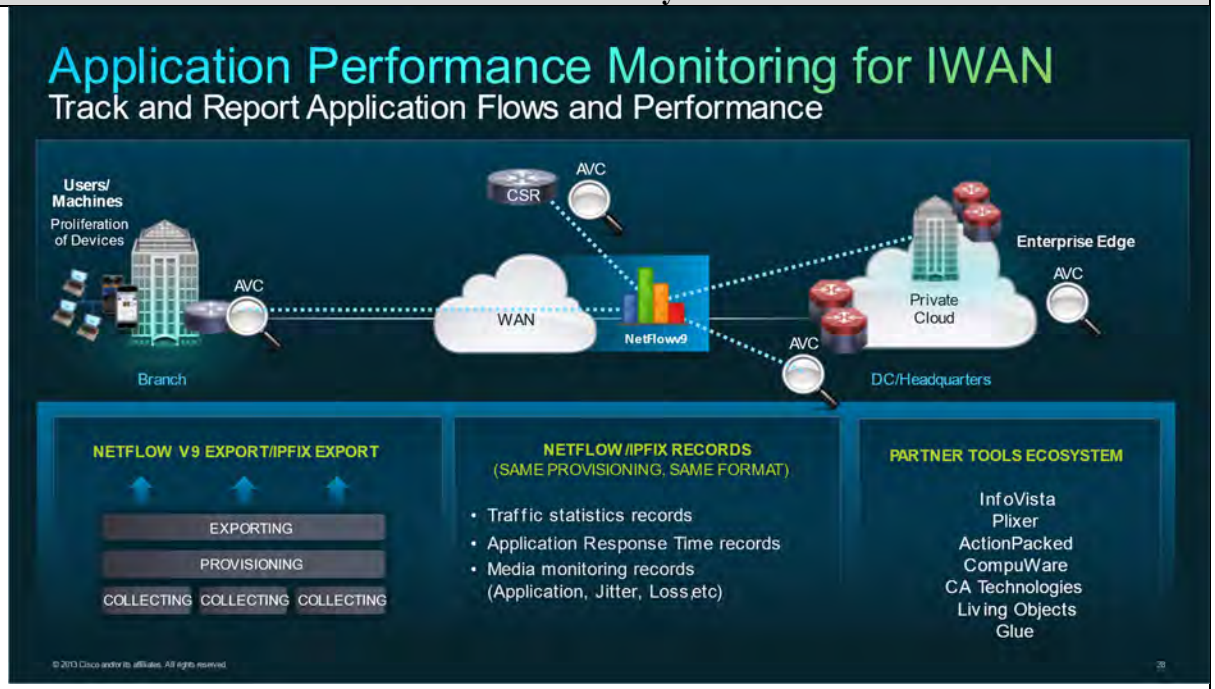
No.	'111 Patent Claim 22	Cisco IWAN System
		

No.	'111 Patent Claim 23	Cisco IWAN System
23[a]	The method according to claim 22, wherein the server device comprises a web server, and	Cisco IWAN System discloses the method according to claim 22, wherein the server device comprises a web server. For example, Cisco IWAN System discloses data center devices, such as a server, that may be a web server. *See supra* at Claim 22. IWAN At-A-Glance

No.	'111 Patent Claim 23	Cisco IWAN System
		

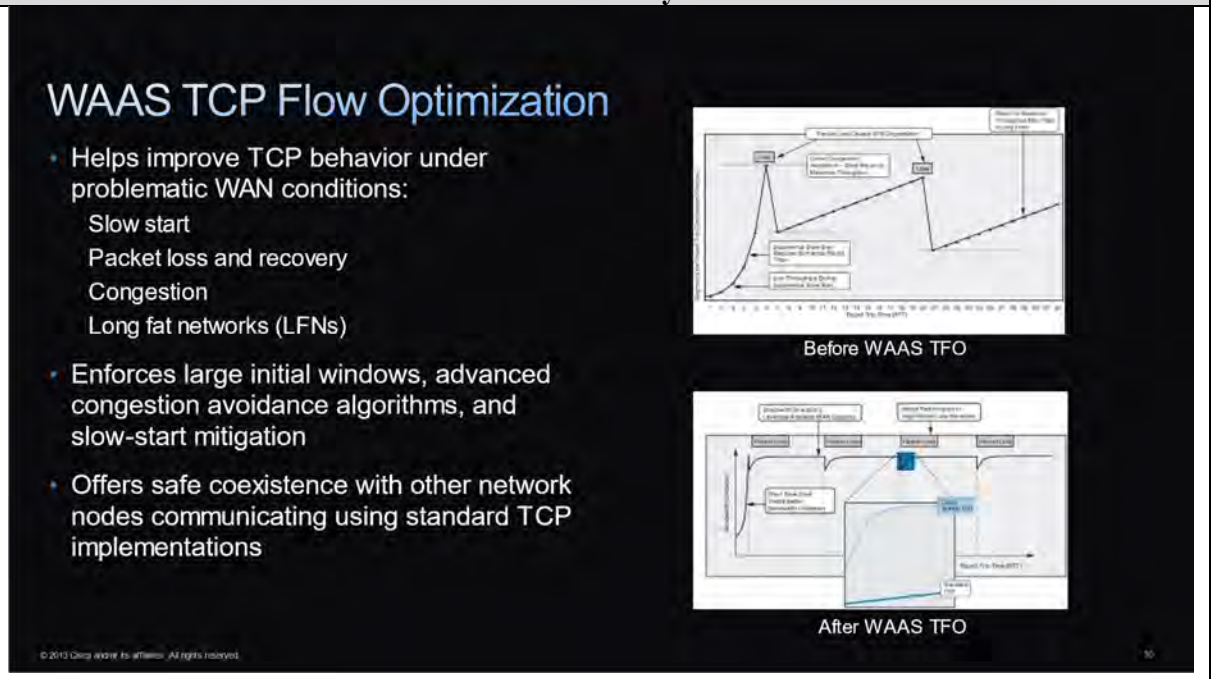
No.	'111 Patent Claim 23	Cisco IWAN System
		
23[b]	wherein the client device comprises a smartphone, a tablet computer, a personal computer, a laptop computer, or a wearable computing device.	Cisco IWAN System discloses wherein the client device comprises a smartphone, a tablet computer, a personal computer, a laptop computer, or a wearable computing device. For example, Cisco IWAN System discloses user devices that may be a mobile phone, tablet, or other personal device. IWAN At-A-Glance

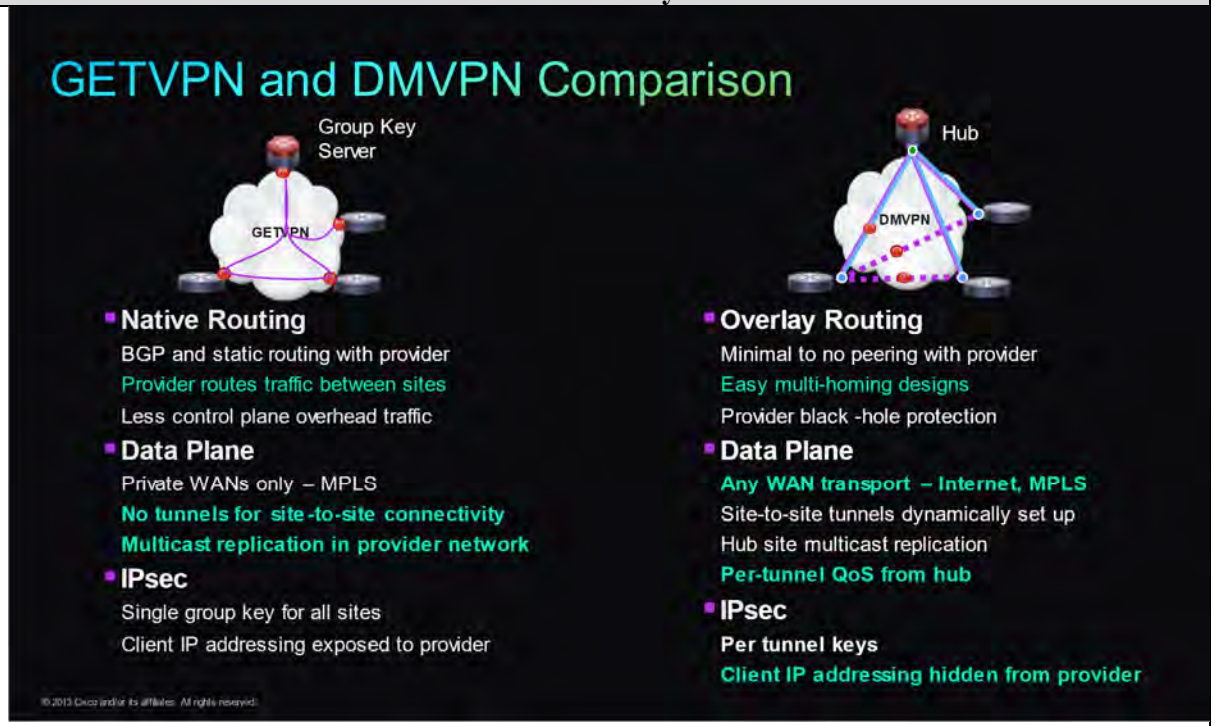
No.	'111 Patent Claim 23	Cisco IWAN System
		

No.	'111 Patent Claim 23	Cisco IWAN System
		

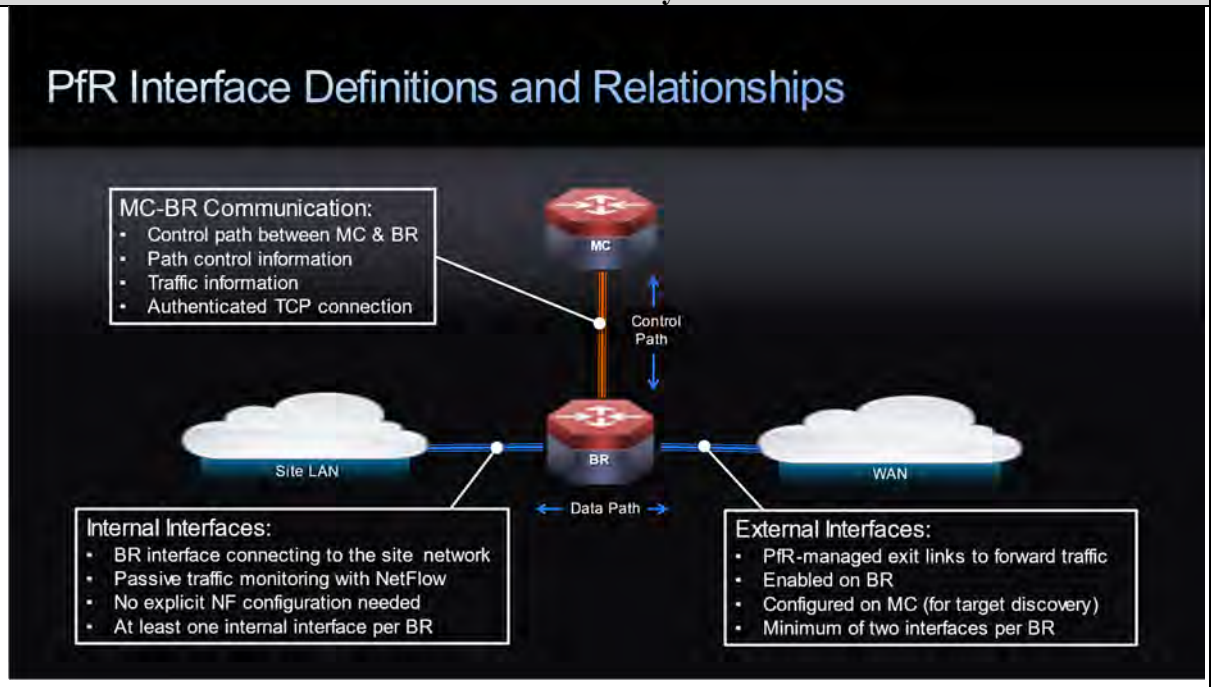
No.	'111 Patent Claim 24	Cisco IWAN System
24	The method according to claim 22, wherein the communication between the network node and the controller is based on, or uses, a standard protocol.	Cisco IWAN System discloses the method according to claim 22, wherein the communication between the network node and the controller is based on, or uses, a standard protocol. For example, Cisco IWAN System discloses communication connections between the branch devices/border routers and hub/master controller that is based on protocols, such as a standard TCP protocol and OpenFlow. *See supra* at Claim 22. Cisco IWAN - Uncompromised Experience

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1121 of 2137

No.	'111 Patent Claim 24	Cisco IWAN System
		 Cisco IWAN - Uncompromised Experience

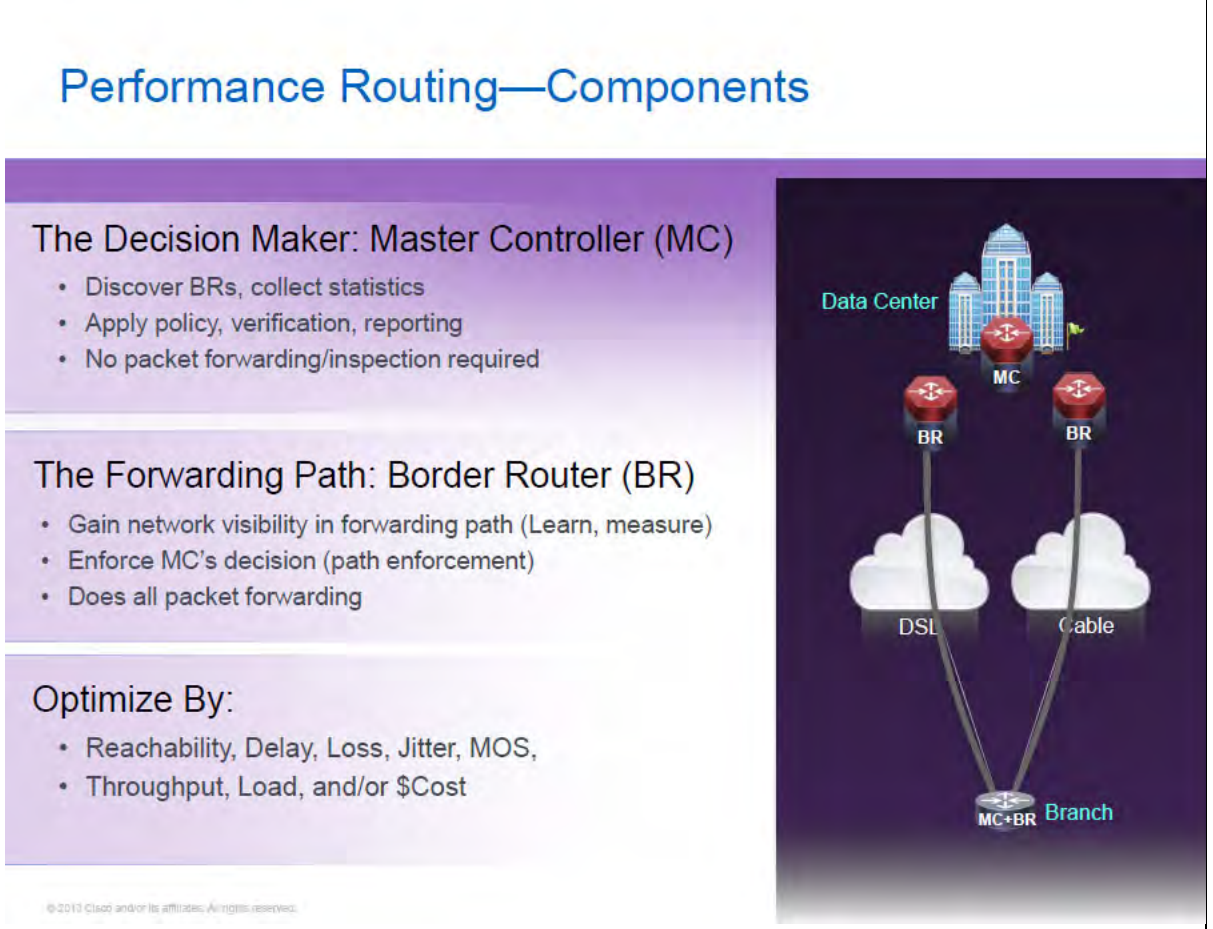
No.	'111 Patent Claim 24	Cisco IWAN System
		 Cisco IWAN - Uncompromised Experience

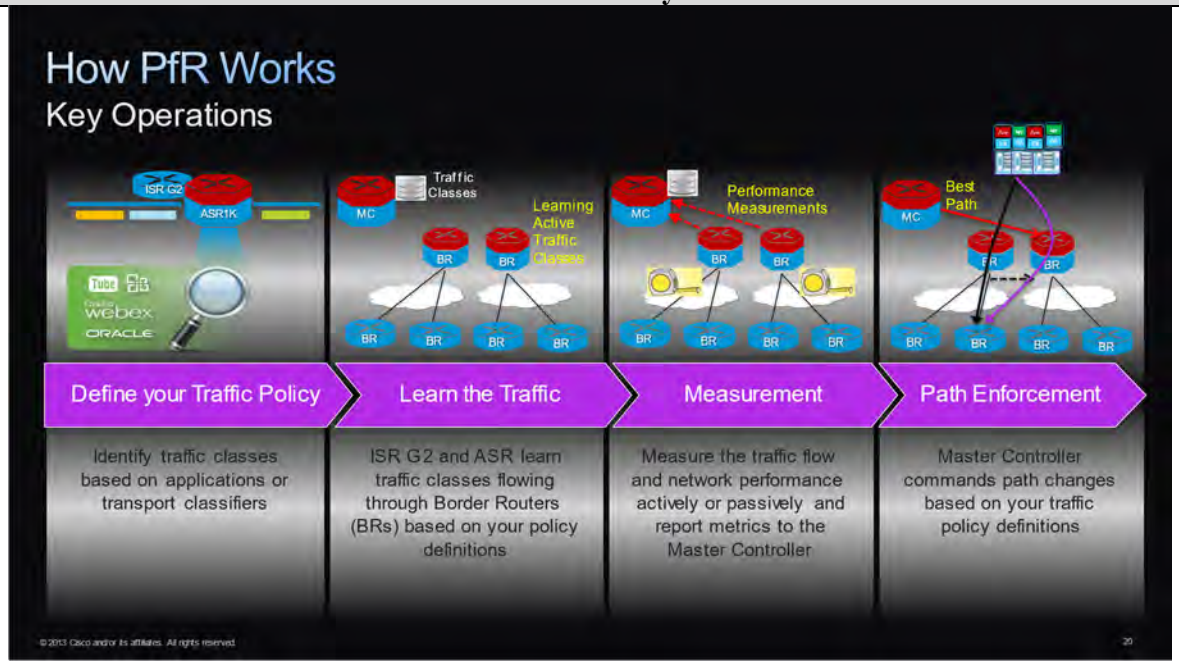
Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1123 of 2137

No.	'111 Patent Claim 24	Cisco IWAN System
		

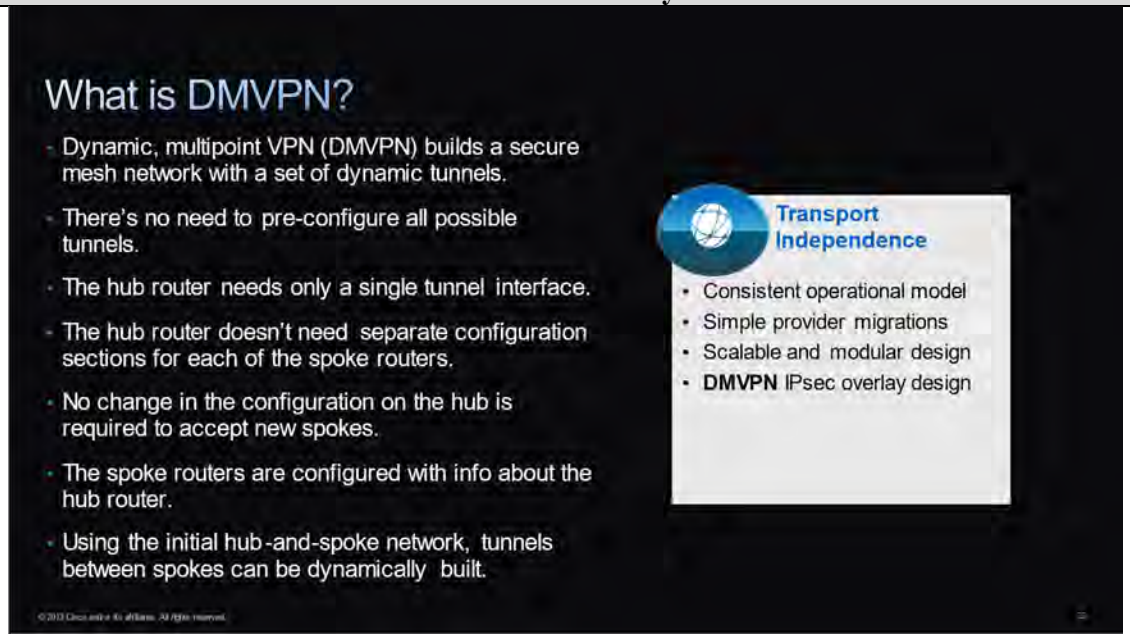
No.	'111 Patent Claim 27	Cisco IWAN System
27	The method according to claim 1, wherein the network node comprises a router, a switch, or a bridge.	Cisco IWAN System discloses the method according to claim 1, wherein the network node comprises a router, a switch, or a bridge. For example, Cisco IWAN System discloses branch devices/spokes that are border routers. *See supra* at Claim 1. IWAN Next Gen at 25

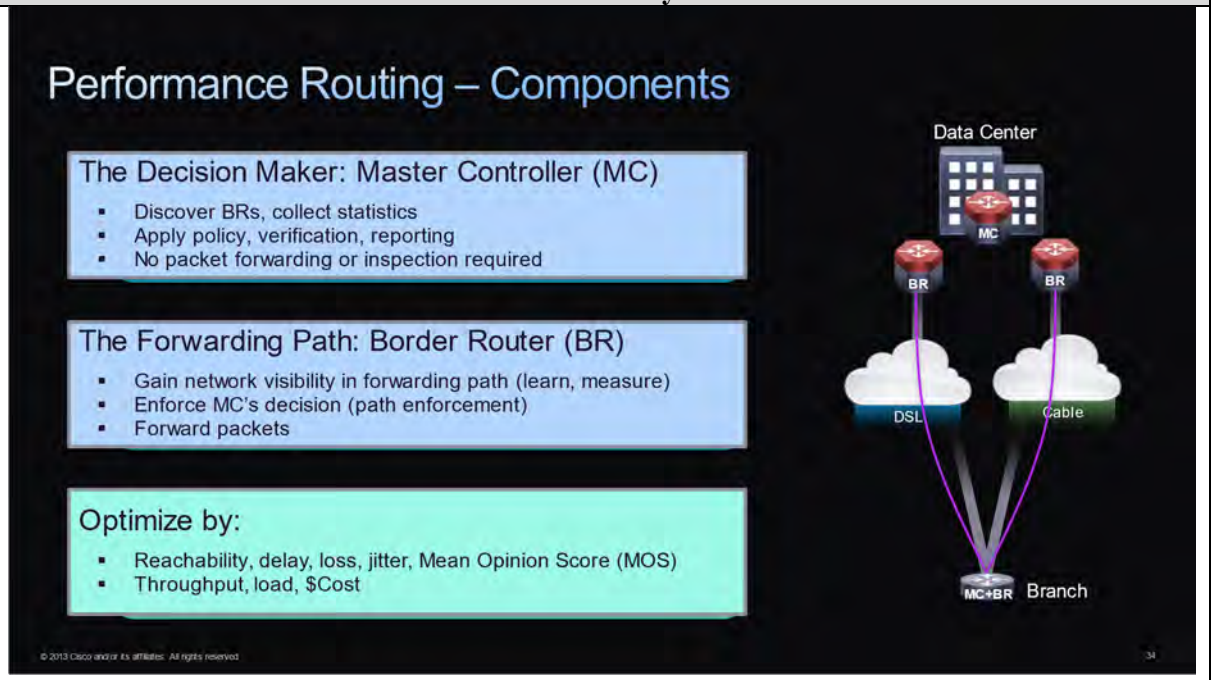
Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1124 of 2137

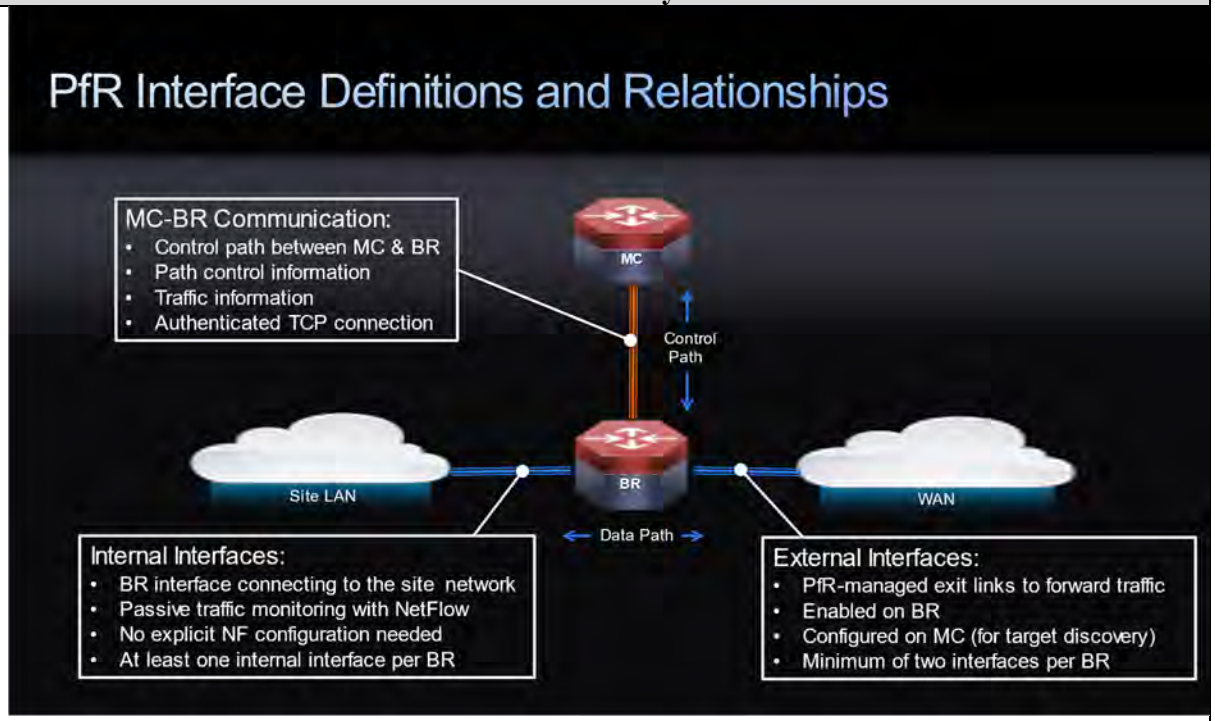
No.	'111 Patent Claim 27	Cisco IWAN System
		 Cisco IWAN

No.	'111 Patent Claim 27	Cisco IWAN System
		 Cisco IWAN - Uncompromised Experience

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1126 of 2137

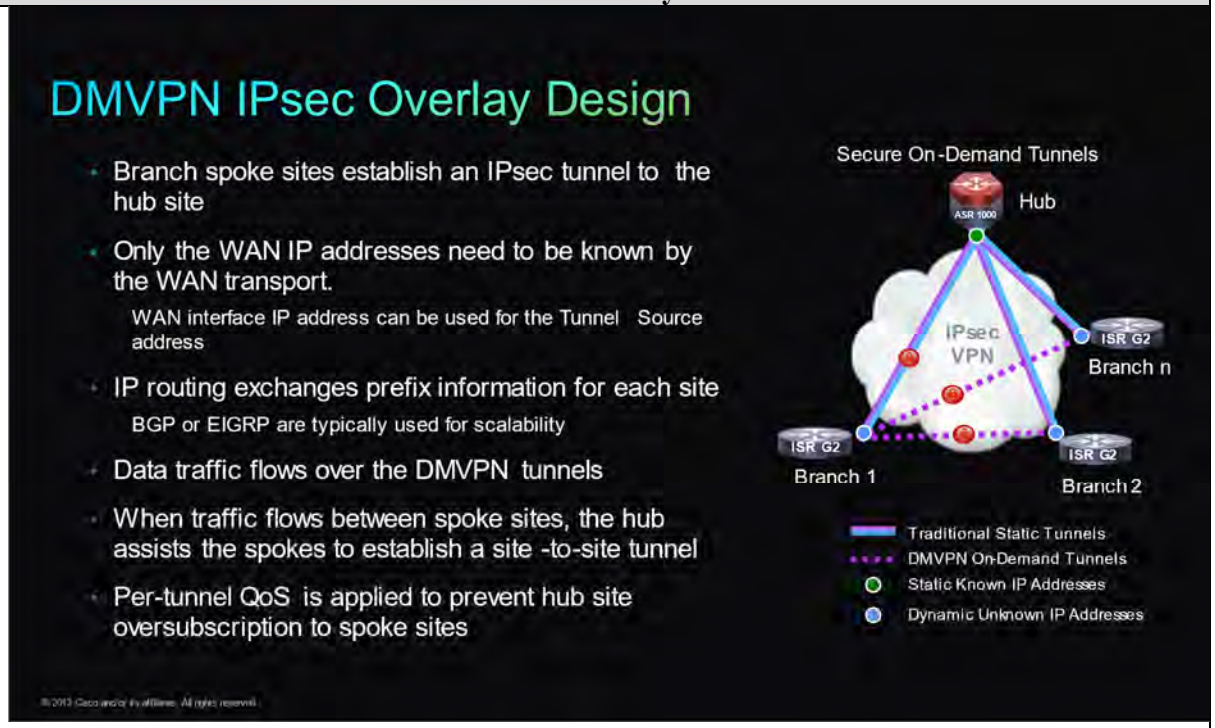
No.	'111 Patent Claim 27	Cisco IWAN System
		 Cisco IWAN - Uncompromised Experience

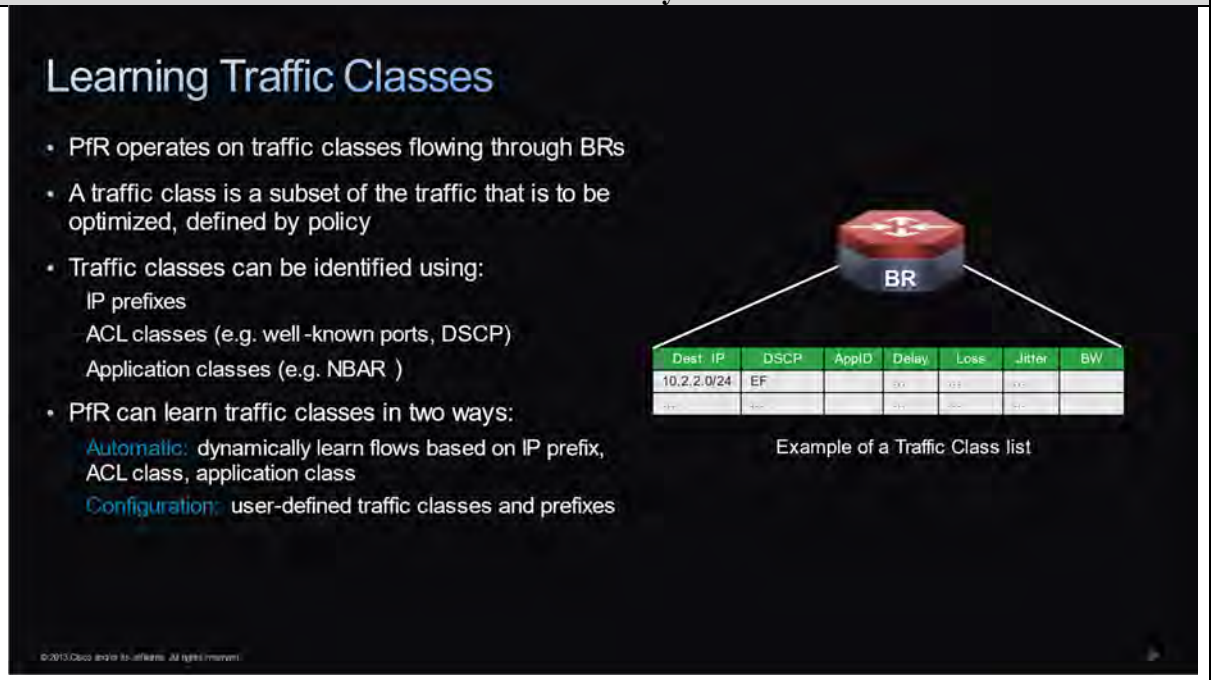
No.	'111 Patent Claim 27	Cisco IWAN System
		

No.	'111 Patent Claim 27	Cisco IWAN System
		

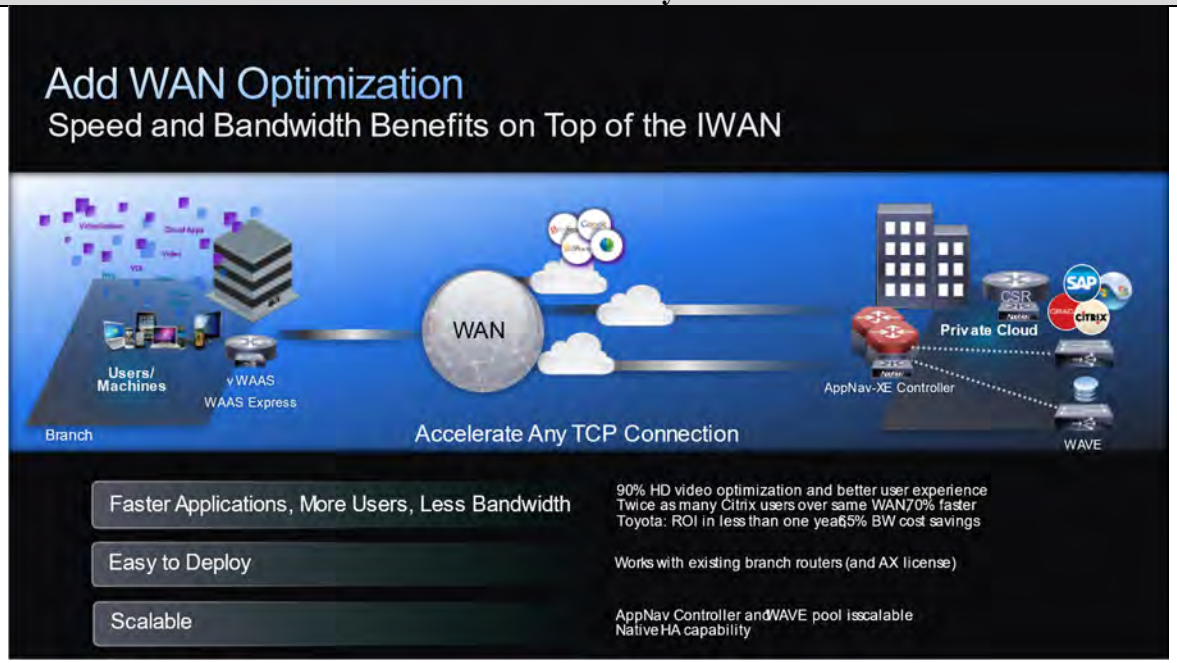
No.	'111 Patent Claim 28	Cisco IWAN System
28	The method according to claim 1, wherein the packet network is an Internet Protocol (IP) network, and the packet is an IP packet.	Cisco IWAN System discloses the method according to claim 1, wherein the packet network is an Internet Protocol (IP) network, and the packet is an IP packet. For example, Cisco IWAN System discloses communication of traffic flows with IP characteristics. *See supra* at Claim 1. Cisco IWAN

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1129 of 2137

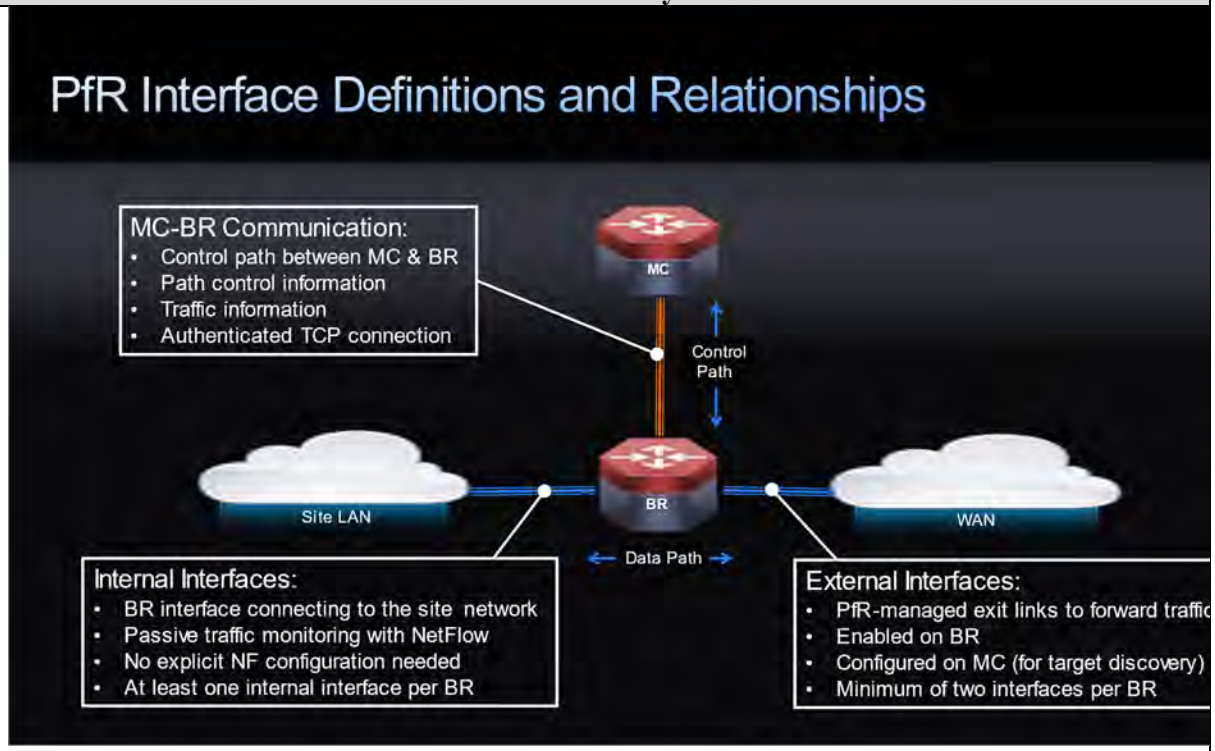
No.	'111 Patent Claim 28	Cisco IWAN System
		 Cisco IWAN - Uncompromised Experience

No.	'111 Patent Claim 28	Cisco IWAN System
		

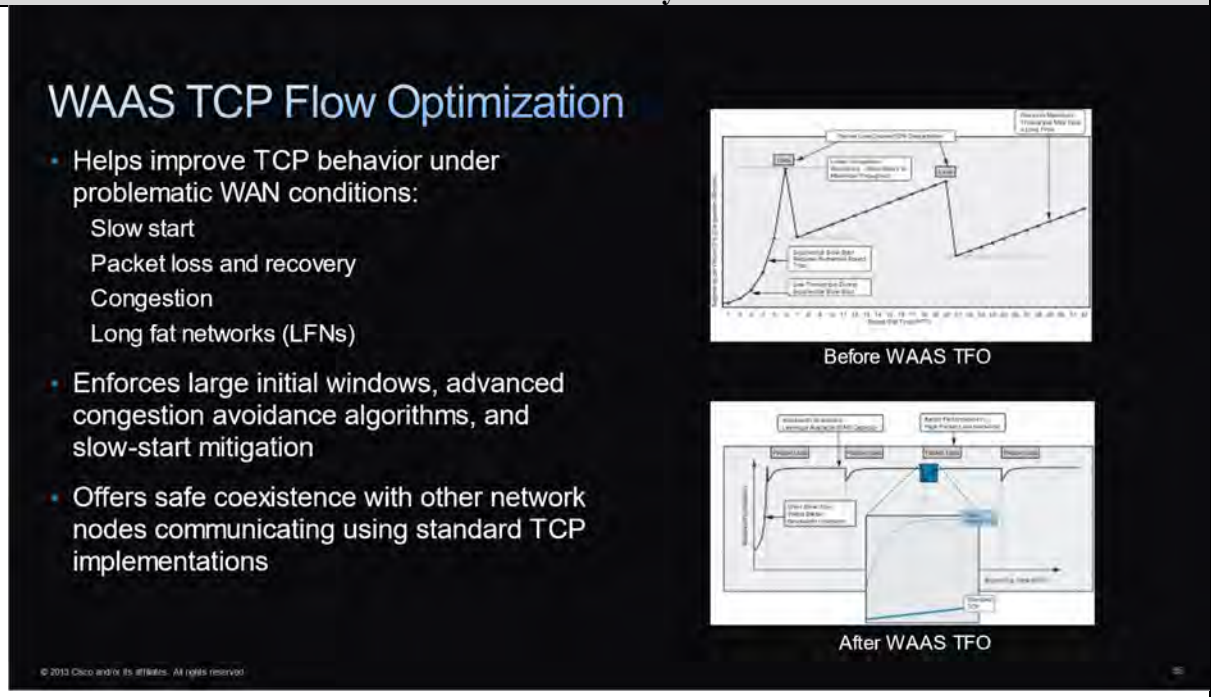
No.	'111 Patent Claim 29	Cisco IWAN System
29	The method according to claim 28, wherein the packet network is an Transmission Control Protocol (TCP) network, and the packet is an TCP packet.	Cisco IWAN System discloses the method according to claim 28, wherein the packet network is an Transmission Control Protocol (TCP) network, and the packet is an TCP packet. For example, Cisco IWAN System discloses communication of traffic flows with TCP characteristics. *See supra* at Claim 28. Cisco IWAN

No.	'111 Patent Claim 29	Cisco IWAN System
		 Cisco IWAN - Uncompromised Experience

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1132 of 2137

No.	'111 Patent Claim 29	Cisco IWAN System
		

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1133 of 2137

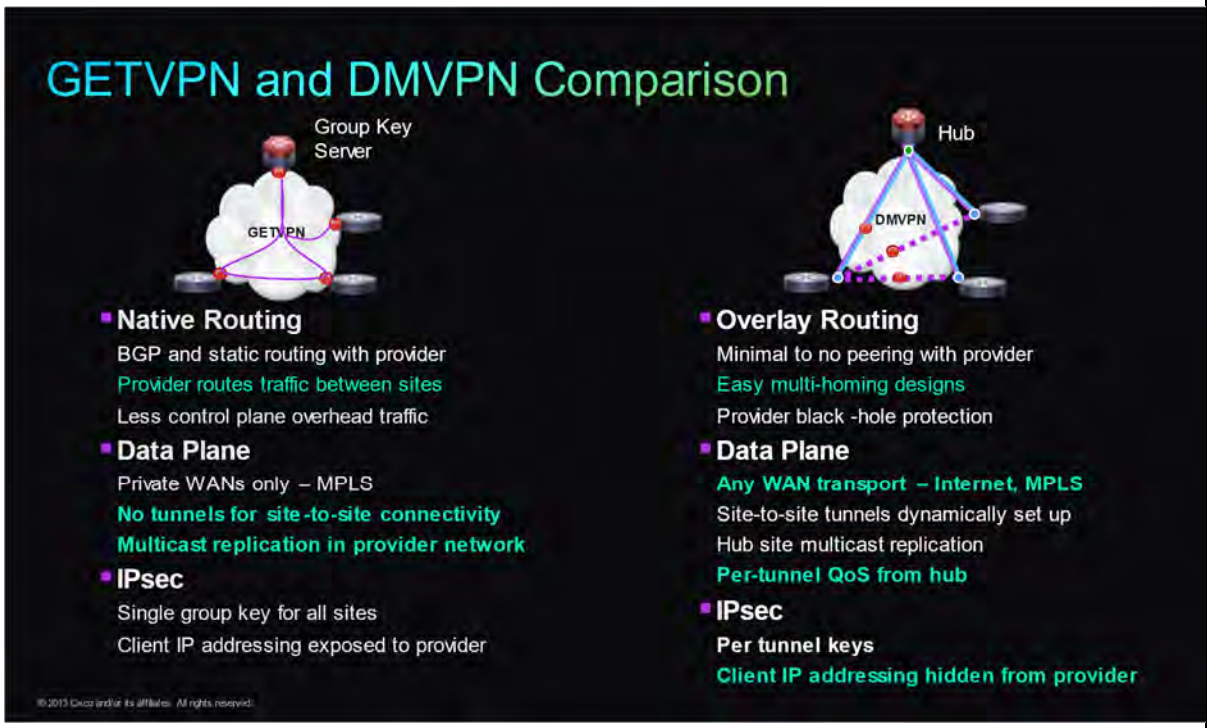
No.	'111 Patent Claim 29	Cisco IWAN System
		

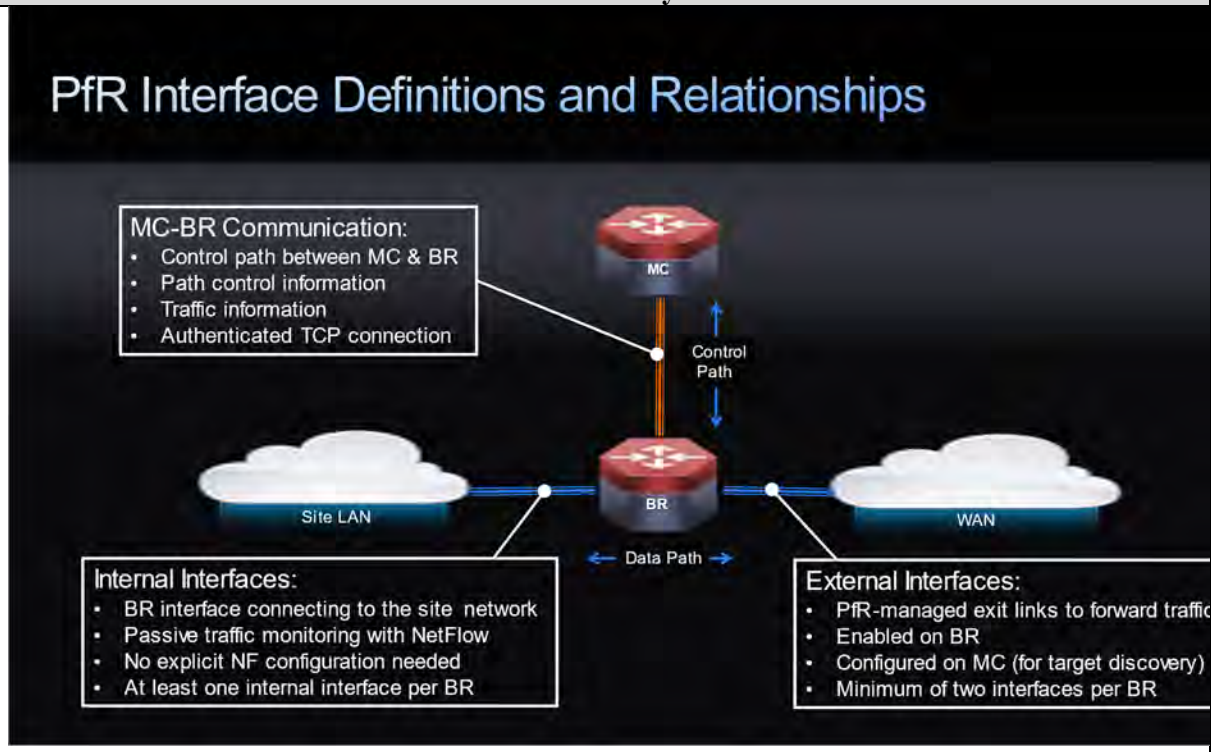
No.	'111 Patent Claim 30	Cisco IWAN System
30[a]	The method according to claim 1, further comprising: receiving, by the network node from the first entity over the packet network, one or more additional packets;	Cisco IWAN System discloses the method according to claim 1, further comprising: receiving, by the network node from the first entity over the packet network, one or more additional packets. *See supra at* Claim 1, 1[c].

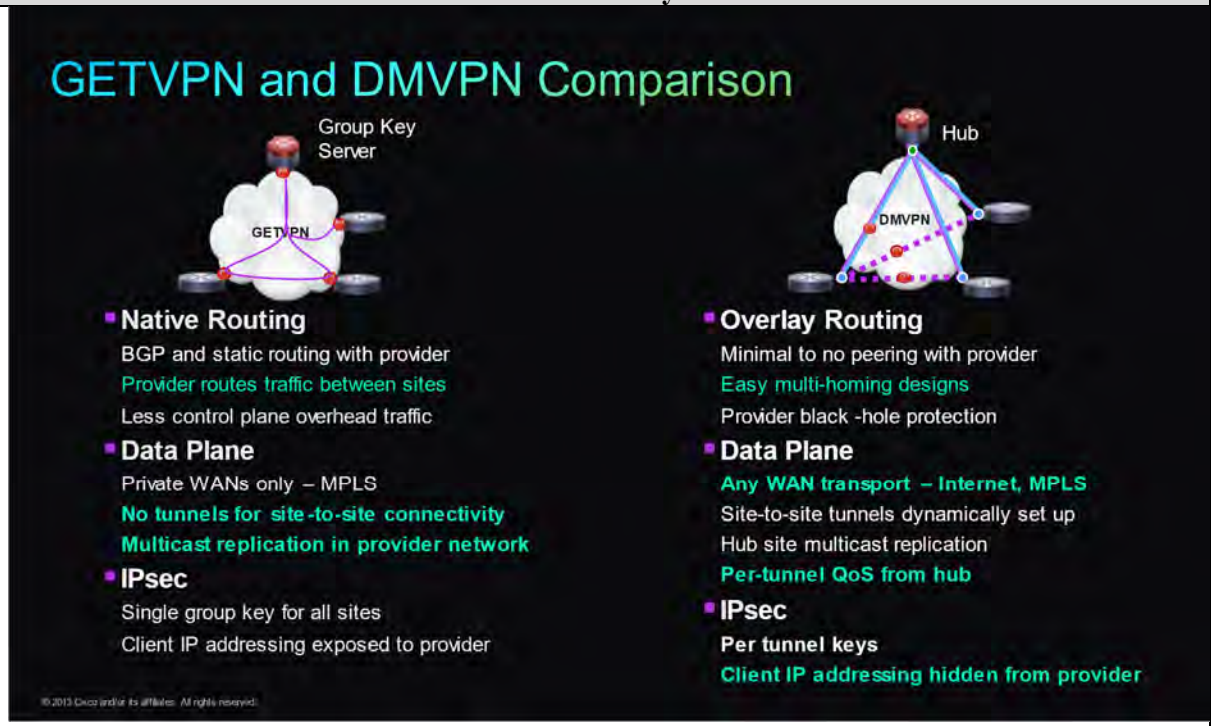
No.	'111 Patent Claim 30	Cisco IWAN System
30[b]	checking, by the network node, if any one of the one or more additional packets satisfies the criterion;	Cisco IWAN System discloses checking, by the network node, if any one of the one or more additional packets satisfies the criterion. *See supra at* 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	Cisco IWAN System 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. *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.	Cisco IWAN System 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. *See supra at* Claim 1[f], 30[a].

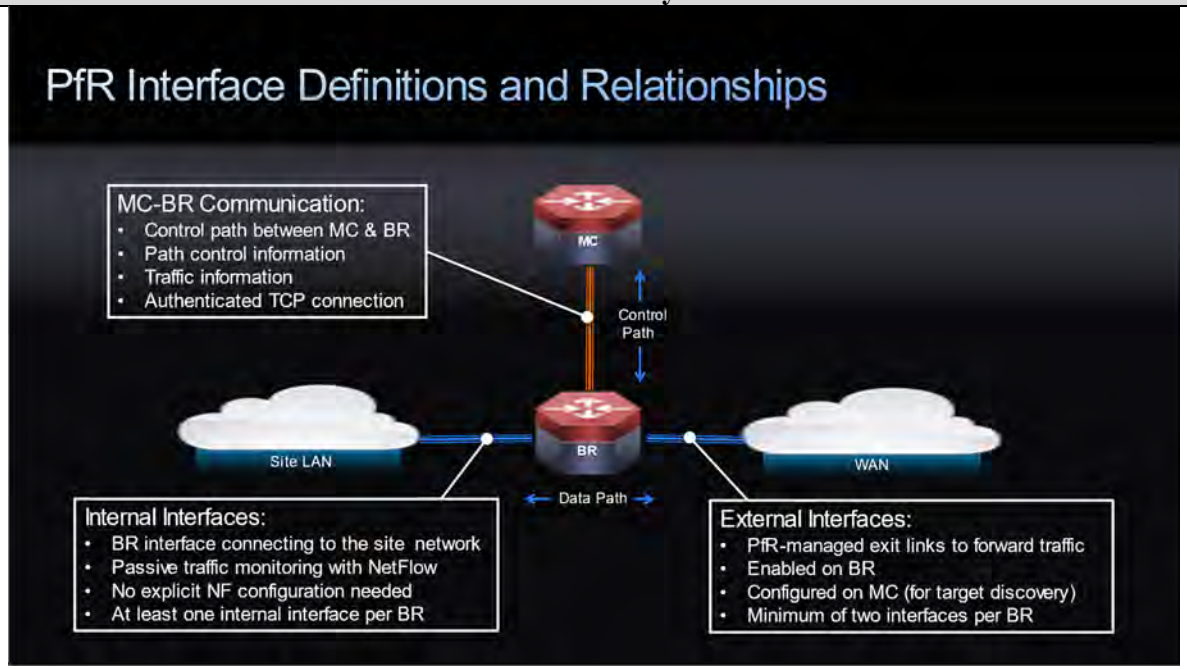
No.	'111 Patent Claim 31	Cisco IWAN System
31[a]	The method according to claim 1, wherein the packet network is a Software Defined Network (SDN),	Cisco IWAN System discloses the method according to claim 1, wherein the packet network is a Software Defined Network (SDN). For example, Cisco IWAN System discloses use of a software data plane in which OpenFlow, a Software Defined Network protocol. *See supra* at Claim 1. Cisco IWAN - Uncompromised Experience

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1135 of 2137

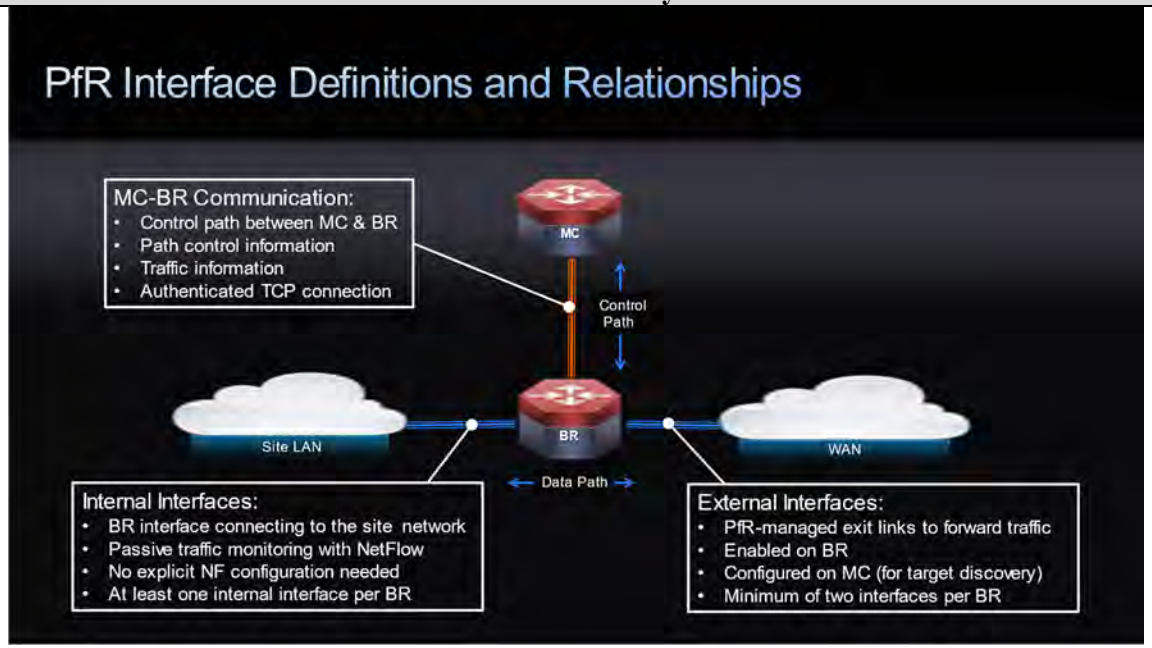
No.	'111 Patent Claim 31	Cisco IWAN System
		 Cisco IWAN - Uncompromised Experience

No.	'111 Patent Claim 31	Cisco IWAN System
		
31[b]	the packet is routed as part of a data plane and	Cisco IWAN System discloses the packet is routed as part of a data plane. For example, Cisco IWAN System discloses traffic flow forwarded by branch devices/border routers in a data plane or data path. Cisco IWAN - Uncompromised Experience

No.	'111 Patent Claim 31	Cisco IWAN System
		 Cisco IWAN - Uncompromised Experience

No.	'111 Patent Claim 31	Cisco IWAN System
		
31[c]	the network node communication with the controller serves as a control plane.	Cisco IWAN System discloses the network node communication with the controller serves as a control. For example, Cisco IWAN System discloses communication between branch devices/border routers and the hub/master controller in a control layer or control path. Cisco IWAN - Uncompromised Experience

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1139 of 2137

No.	'111 Patent Claim 31	Cisco IWAN System
		

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1140 of 2137

Chart for U.S. Patent 10,652,111 ("the '111 Patent")
U.S. Patent No. 9,276,877 to Chua '877 et al. ("Chua '877")

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 Chua '877 meets each element of those claims, and/or (2) 35 U.S.C. § 103 because Chua '877 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 Chua '877 titled "Data Model For Software Defined Networks", which was filed on March 15, 2013, and issued on March 1, 2016.

Motivations to combine the disclosures in Chua '877 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 Preliminary Infringement Disclosure, 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.

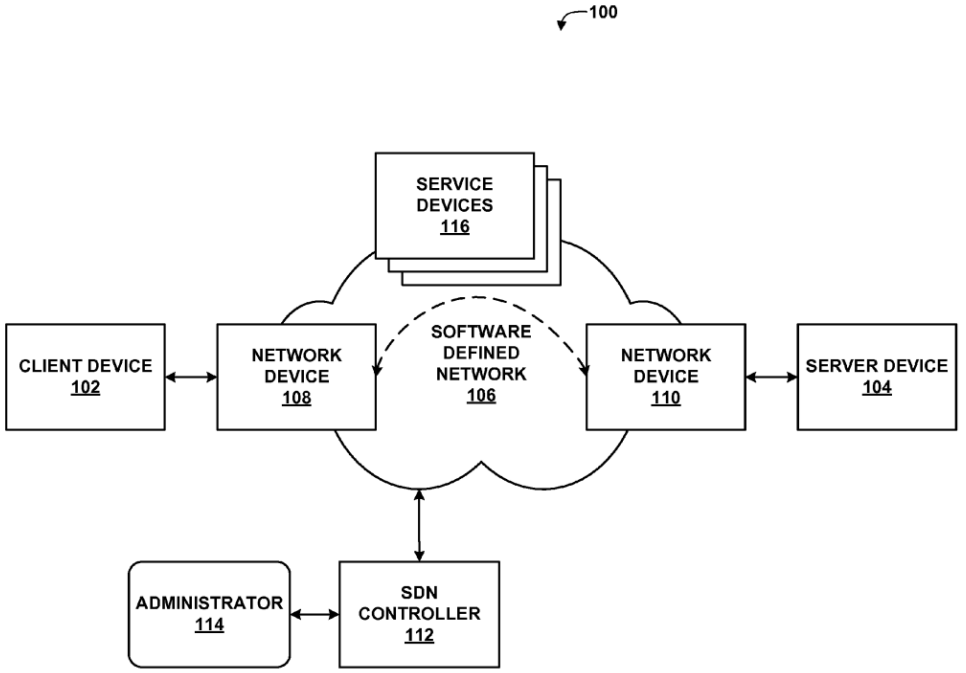
Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1141 of 2137

No.	'111 Patent Claim 1	Chua '877
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:	Chua '877 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. For example, Chua '877 discloses a method for use in a software defined network in which a remote controller device programs network and service devices that transport packets, to apply rules when sending packets between first and second devices. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. Chua '877 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.") Chua '877 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.") Chua '877 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.") 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,

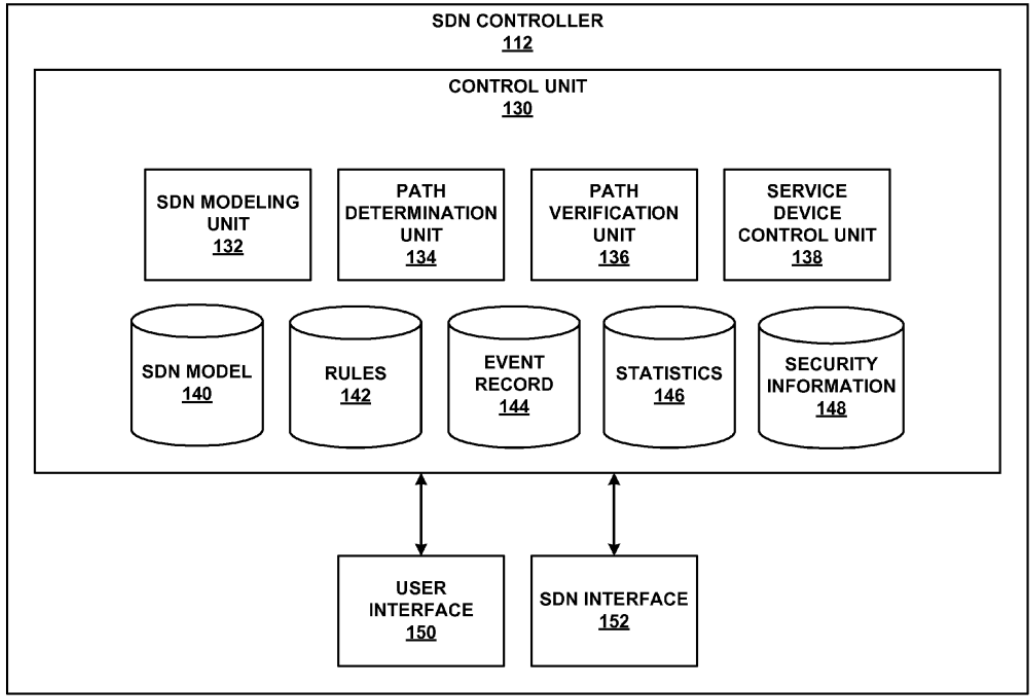
Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1142 of 2137

No.	'111 Patent Claim 1	Chua '877
		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.") Chua '877 at 20:5-18 ("As shown in FIG. 3, web clients 202 connect to Internet 204. Internet 204 provides access to system 200, which implements the techniques of this disclosure. For example, network devices of system 200 form an SDN controlled by SDN controller 218. SDN controller 218 may generally cor-respond to SDN controller 112 of FIGS. 1 and 2. In particular, SDN controller 218 (in this example) programs one or more switches (e.g., Pica8 3290 switches, IBM G8264), such as switch 210, to direct traffic through an SDN to other switches, service devices (e.g., web proxy 212, IDS 214, and/or webs erver 216). In general, SDN controller 218 may be config-ured to perform any or all of the techniques described above, alone or in any combination.") Chua '877 at 26:23-43 ("SDN controller 112 may then program the network devices to forward traffic along the determined paths (310). In con-ventional routers, control plane elements of a router program a forwarding information base (FIB) of a data plane of the router. However, SDN controller 112 represents a control plane element that can program data plane elements (e.g., switches) remotely, that is, without being part of or even directly coupled to the data plane elements. For example, SDN controller 112 may program any or all of network devices 108,110 and service devices 116 to forward network traffic having particular header data (e.g., source IP address, destination IP address, source port, destination port, protocol, MPLS label(s), VLAN tags, or the like) to a particular next hop (that is, next network device). Moreover, in accordance with the techniques of this disclosure, SDN controller 112 may

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1143 of 2137

No.	'111 Patent Claim 1	Chua '877
		program network devices of an SDN to forward traffic according to network paths that were determined based on one or more of connections determined between the network devices, service devices among the network devices, zones for packet flows through the network devices, and/or trusted or untrusted packet flows.") Chua '877 at Figure 1  FIG. 1 Chua '877 at Figure 2

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1144 of 2137

No.	'111 Patent Claim 1	Chua '877
		 FIG. 2
1[a]	sending, by the controller to the network node over the packet network, an instruction and a packet-applicable criterion;	Chua '877 discloses sending, by the controller to the network node over the packet network, an instruction and a packet-applicable criterion. For example, Chua '877 discloses a remote controller device programming network devices to perform a programmed action, based on packet data and values. Chua '877 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1145 of 2137

No.	'111 Patent Claim 1	Chua '877
		network devices based on the determination of the connections, and program the network devices to direct network traffic along the one or more paths") Chua '877 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.") Chua '877 at 2:49-61 ("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 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.") Chua '877 at 4:44-55 ("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 receive cre-dentials from a client device in accordance with a public key infrastructure (PKI)-based authentication protocol, deter-mine one or more policies that are applicable to the client device based on the received credentials, and program net-work devices of the SDN to enforce the determined policies on a per-packet-flow basis for packet flows including the client device.") Chua '877 at 5:32-49 ("FIG. 1 is a block diagram illustrating an example system 100 in which various techniques of this disclosure may be used. In this example, system 100 includes software defined network (SDN) 106, which includes network devices 108, 110 and service devices 116. Network devices 108, 110 may comprise switches, and other devices (not shown). These network devices can be physical instantiations or virtual instantiations. SDN 106 may also include other types of devices, such as routers, load balancers, various L4-L 7

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1146 of 2137

No.	'111 Patent Claim 1	Chua '877
		net-work devices, or even multi-tenant capable devices, among other network devices. Again, these network devices can be either of physical or virtual instantiations. In general, network devices of SDN 106, such as network devices 108,110, can be programmed to forward network traffic. System 100 also includes software defined network (SDN) controller 112. Administrator 114 uses SDN controller 112 to control (that is, program) network devices of SDN 106, such as network devices 108, 110.") 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 configured 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 programmed actions on packets of a packet flow based on this data. Such programmed actions may include: Allow—explicitly allow a certain network flow to proceed to its destination Block—explicitly block a certain flow from traversing SDN 106 Mirror—allow the traffic, but send a copy of the traffic for deeper inspection or recording to, e.g., one of service devices 116 Redirect—redirect the traffic to another network device (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. Transform—modify or translate values of headers of packets in the network flow Encapsulate—encapsulate packets in the network flow with a particular header") Chua '877 at 21:17-31 ("SDN controller 218 may be configured to program a switch, such as switch 210, to send packets of a particular packet flow to a service device, e.g., one of web proxy 212, IDS 214, and web server 216. For example, switch 210 is communicatively

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1147 of 2137

No.	'111 Patent Claim 1	Chua '877
		coupled to web proxy 210 by connection 222, which may represent ports 7 and 8. One of web clients 202 may request data from a web server, which may be internal to or external to system 200. Web proxy 212 may be configured to act as a proxy device to the web server. That is, SDN controller 218 may program switch 210 to direct net-work traffic destined for the web server to web proxy 212 via connection 222, instead. Web proxy 212 may then act on behalf of the web server, e.g., by caching resources stored by the web server and responding to requests for the resources on behalf of the web server.") 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. 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.") 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.")

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1148 of 2137

No.	'111 Patent Claim 1	Chua '877
		Chua '877 at 31:30-47 ("SDN controller 112 may then program network devices of the SDN to enforce the policies (400). For example, SDN controller 112 may program the network devices to direct packet flows including client device 102 along various paths, depending on the applicable policies for the packet flows. That is, certain packet flows originating from client device 102 or destined for client device 102 may be forwarded along different paths than other packet flows originating from or destined for client device 102, based on, e.g., the user of client device 102, an application, corresponding to the packet flow, that is being executed by client device 102, and/or a geographic location of client device 102. Likewise, packet flows for the same user using client device 102 and the same appli-cation may be directed along different paths through SDN 106 based on the geographic location of client device 102. Again, the policies may correspond to any combination of these factors, and may include additional or alternative fac-tors in other examples.") Chua '877 at Figure 5 (annotation added)

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1149 of 2137

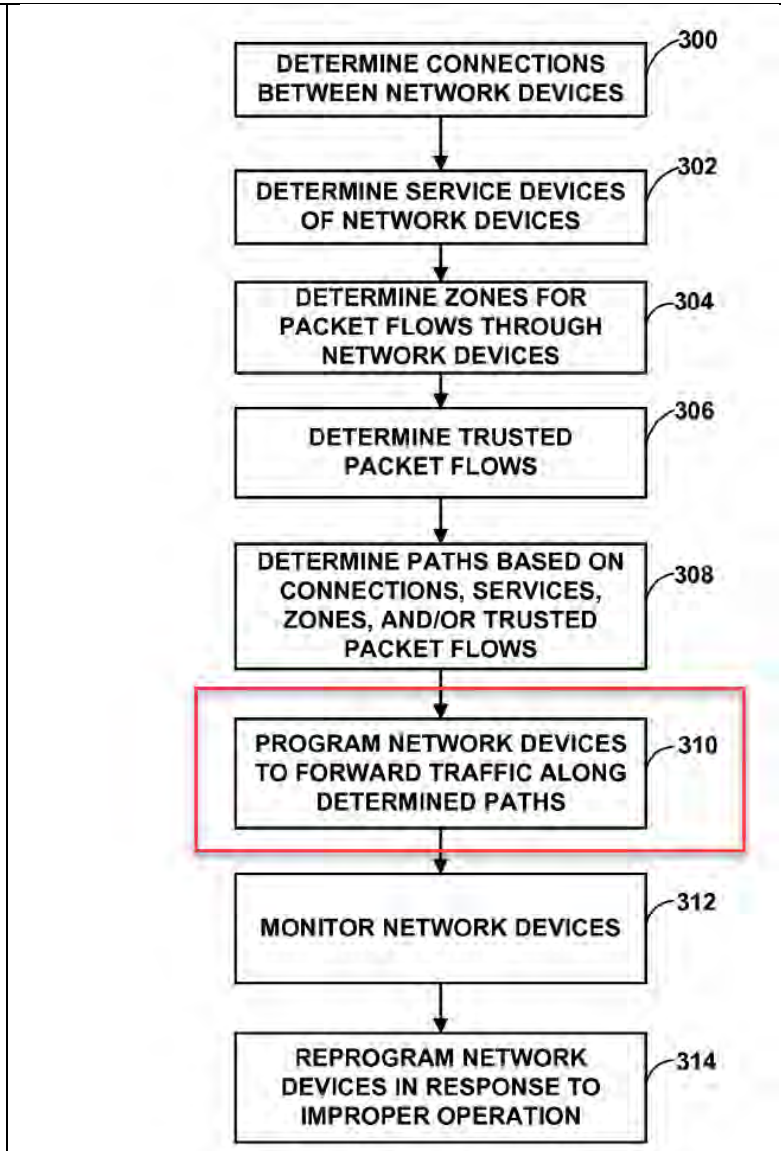

DETERMINE CONNECTIONS BETWEEN NETWORK DEVICES
300
DETERMINE SERVICE DEVICES OF NETWORK DEVICES
302
DETERMINE ZONES FOR PACKET FLOWS THROUGH NETWORK DEVICES
304
DETERMINE TRUSTED PACKET FLOWS
306
DETERMINE PATHS BASED ON CONNECTIONS, SERVICES, ZONES, AND/OR TRUSTED PACKET FLOWS
308
PROGRAM NETWORK DEVICES TO FORWARD TRAFFIC ALONG DETERMINED PATHS
310
MONITOR NETWORK DEVICES
312
REPROGRAM NETWORK DEVICES IN RESPONSE TO IMPROPER OPERATION
314

FIG. 5

No.	'111 Patent Claim 1	Chua '877
1[b]	receiving, by the network node from the controller, the instruction and the criterion;	Chua '877 discloses receiving, by the network node from the controller, the 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;	Chua '877 discloses receiving, by the network node from first entity over the packet network, a packet addressed to the second entity. For example, Chua '877 discloses network devices receiving packets from a client device or server device, that are intended to be forwarded to another client device or server device. Chua '877 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") Chua '877 at 5:32-49 ("FIG. 1 is a block diagram illustrating an example system 100 in which various techniques of this disclosure may be used. In this example, system 100 includes software defined network (SDN) 106, which includes network devices 108, 110 and service devices 116. Network devices 108, 110 may comprise switches, and other devices (not shown). These network devices can be physical instantiations or virtual instantiations. SDN 106 may also include other types of devices, such as routers, load balancers, various L4-L 7 net-work devices, or even multi-tenant capable devices, among other network devices. Again, these network devices can be either of physical or virtual instantiations. In general, network devices of SDN 106, such as network devices 108,110, can be programmed to forward network traffic. System 100 also includes software defined network (SDN) controller 112. Administrator 114 uses SDN controller 112 to control (that is, program) network devices of SDN 106, such as network devices 108, 110.") 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,

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1151 of 2137

No.	'111 Patent Claim 1	Chua '877
		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.") Chua '877 at 26:15-22 ("Determining the paths generally includes determining, from a first network device of the SDN, subsequent "next hop" network devices through the SDN. Thus, SDN control-ler 112 may analyze connections between the network devices (e.g., switches and service devices) and direct traffic from one device to another, assuming that the devices are communicatively coupled and cause the traffic to travel substantially in the direction of the destination of the traffic.") Chua '877 at 26:23-43 ("SDN controller 112 may then program the network devices to forward traffic along the determined paths (310). In conventional routers, control plane elements of a router program a forwarding information base (FIB) of a data plane of the router. However, SDN controller 112 represents a control plane element that can program data plane elements (e.g., switches) remotely, that is, without being part of or even directly coupled to the data plane elements. For example, SDN controller 112 may program any or all of network devices 108, 110 and service devices 116 to forward network traffic having particular header data (e.g., source IP address, destination IP address, source port, destination port, protocol, MPLS label(s), VLAN tags, or the like) to a particular next hop (that is, next network device). Moreover, in accordance with the techniques of this disclosure, SDN controller 112 may program network devices of an SDN to forward traffic according to network paths that were determined based on one or more of connections determined between the network devices, service devices among the network devices, zones for packet flows through the network

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1152 of 2137

No.	'111 Patent Claim 1	Chua '877
		devices, and/or trusted or untrusted packet flows.") Chua '877 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 packets with a particular header, or other such actions.") Chua '877 at 31:30-47 ("SDN controller 112 may then program network devices of the SDN to enforce the policies (400). For example, SDN controller 112 may program the network devices to direct packet flows including client device 102 along various paths, depending on the applicable policies for the packet flows. That is, certain packet flows originating from client device 102 or destined for client device 102 may be forwarded along different paths than other packet flows originating from or destined for client device 102, based on, e.g., the user of client device 102, an application, corresponding to the packet flow, that is being executed by client device 102, and/or a geographic location of client device 102. Likewise, packet flows for the same user using client device 102 and the same appli-cation may be directed along different paths through SDN 106 based on the geographic location of client device 102. Again, the policies may correspond to any combination of these factors, and may include additional or alternative factors in other examples.")

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1153 of 2137

No.	'111 Patent Claim 1	Chua '877
1[d]	checking, by the network node, if the packet satisfies the criterion;	Chua '877 discloses checking, by the network node, if the packet satisfies the criterion. For example, Chua '877 discloses network devices, that are programmed to enforce policies by the remote controller device based on a per-packet flow basis including by analyzing the particular header data of packets. Chua '877 at 4:44-55 ("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 receive credentials from a client device in accordance with a public key infrastructure (PKI)-based authentication protocol, determine one or more policies that are applicable to the client device based on the received credentials, and program network devices of the SDN to enforce the determined policies on a per-packet-flow basis for packet flows including the client device.") 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 configured 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 programmed actions on packets of a packet flow based on this data. Such programmed actions may include: Allow—explicitly allow a certain network flow to proceed to its destination Block—explicitly block a certain flow from traversing SDN 106 Mirror—allow the traffic, but send a copy of the traffic for deeper inspection or recording to, e.g., one of service devices 116 Redirect—redirect the traffic to another network device (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.
IPR2023-00401, Page 1154 of 2137

No.	'111 Patent Claim 1	Chua '877
		Transform—modify or translate values of headers of packets in the network flow Encapsulate—encapsulate packets in the network flow with a particular header") 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.") Chua '877 at 26:15-22 ("Determining the paths generally includes determining, from a first network device of the SDN, Subsequent "next hop' network devices through the SDN. Thus, SDN controller 112 may analyze connections between the network devices (e.g., Switches and service devices) and direct traffic from one device to another, assuming that the devices are communicatively coupled and cause the traffic to travel substantially in the direction of the destination of the traffic.") Chua '877 at 26:23-43 ("SDN controller 112 may then program the network devices to forward traffic along the determined paths (310). In con-ventional routers, control plane elements of a router program a forwarding information base (FIB) of a data plane of the router. However, SDN controller 112 represents a control plane element that can program data plane elements (e.g., switches) remotely, that is, without being part of or even directly coupled to the data plane elements. For example, SDN controller 112 may program any or all of network devices 108,110 and service devices 116 to forward network traffic having particular header

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1155 of 2137

No.	'111 Patent Claim 1	Chua '877
		data (e.g., source IP address, destination IP address, source port, destination port, protocol, MPLS label(s), VLAN tags, or the like) to a particular next hop (that is, next network device). Moreover, in accordance with the techniques of this disclosure, SDN controller 112 may program network devices of an SDN to forward traffic according to network paths that were determined based on one or more of connections determined between the network devices, service devices among the network devices, zones for packet flows through the network devices, and/or trusted or untrusted packet flows.") Chua '877 at 27:56-28:6 ("In addition, SDN controller 112 may program network devices to receive packets from the service device (332). More particularly, programming the network devices may include programming network devices to receive packets from a service device and forward the packets along the path that was determined for the packet flow corresponding to the packets. In some instances, the network device that forwards a packet to a service device may be the same network device that receives the packet from the service device. In such instances, SDN controller 112 may program a forwarding information base (FIB) of the network device such that the network device analyzes not only a packet header of the packet to determine a next hop for the packet, but also an input interface (e.g., an input port) by which the packet is received. That is, the FIB may include a forwarding table that includes data defining a correspondence from packet header information and an input interface to an output interface, for a packet flow.") Chua '877 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1156 of 2137

No.	'111 Patent Claim 1	Chua '877
		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.") Chua '877 at 31:30-47 ("SDN controller 112 may then program network devices of the SDN to enforce the policies (400). For example, SDN controller 112 may program the network devices to direct packet flows including client device 102 along various paths, depending on the applicable policies for the packet flows. That is, certain packet flows originating from client device 102 or destined for client device 102 may be forwarded along different paths than other packet flows originating from or destined for client device 102, based on, e.g., the user of client device 102, an application, corresponding to the packet flow, that is being executed by client device 102, and/or a geographic location of client device 102. Likewise, packet flows for the same user using client device 102 and the same appli-cation may be directed along different paths through SDN 106 based on the geographic location of client device 102. Again, the policies may correspond to any combination of these factors, and may include additional or alternative fac-tors in other examples.")
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	Chua '877 discloses responsive to the packet not satisfying the criterion, sending, by the network node over the packet network, the packet to the second entity. For example, Chua '877 discloses a network device sending a packet to the next hop identified by the destination address within the header of the packet, if the packet's header data does not match the programmed policy, *i.e,*, responsive to the packet not satisfying the criterion, sending, by the network node over the packet network, the packet to the second entity. 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 configured 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 programmed actions on packets of a packet flow based on this data. Such

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1157 of 2137

No.	'111 Patent Claim 1	Chua '877
		programmed actions may include: Allow—explicitly allow a certain network flow to proceed to its destination Block—explicitly block a certain flow from traversing SDN 106 Mirror—allow the traffic, but send a copy of the traffic for deeper inspection or recording to, e.g., one of service devices 116 Redirect—redirect the traffic to another network device (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. Transform—modify or translate values of headers of packets in the network flow Encapsulate—encapsulate packets in the network flow with a particular header") 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.") Chua '877 at 26:15-22 ("Determining the paths generally includes determining, from a first network device of the SDN, Subsequent "next hop' network devices through the SDN. Thus,

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1158 of 2137

No.	'111 Patent Claim 1	Chua '877
		SDN controller 112 may analyze connections between the network devices (e.g., Switches and service devices) and direct traffic from one device to another, assuming that the devices are communicatively coupled and cause the traffic to travel substantially in the direction of the destination of the traffic.") Chua '877 at 26:23-43 ("SDN controller 112 may then program the network devices to forward traffic along the determined paths (310). In con-ventional routers, control plane elements of a router program a forwarding information base (FIB) of a data plane of the router. However, SDN controller 112 represents a control plane element that can program data plane elements (e.g., switches) remotely, that is, without being part of or even directly coupled to the data plane elements. For example, SDN controller 112 may program any or all of network devices 108,110 and service devices 116 to forward network traffic having particular header data (e.g., source IP address, destination IP address, source port, destination port, protocol, MPLS label(s), VLAN tags, or the like) to a particular next hop (that is, next network device). Moreover, in accordance with the techniques of this disclosure, SDN controller 112 may program network devices of an SDN to forward traffic according to network paths that were determined based on one or more of connections determined between the network devices, service devices among the network devices, zones for packet flows through the network devices, and/or trusted or untrusted packet flows.") Chua '877 at 27:56-28:6 ("In addition, SDN controller 112 may program network devices to receive packets from the service device (332). More particularly, programming the network devices may include programming network devices to receive packets from a service device and forward the packets along the path that was determined for the packet flow corresponding to the packets. In some instances, the network device that forwards a packet to a service device may be the same network device that receives the packet from the service device. In such instances, SDN controller 112 may program a forwarding information base (FIB) of the network device such that the network device analyzes not only a packet header of the packet to determine a next hop for the packet, but also an input interface (e.g., an input port) by which the packet is received. That is, the FIB may include a forwarding table that includes data defining a correspondence from packet header information and an input interface to an output interface, for a packet flow.")

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1159 of 2137

No.	'111 Patent Claim 1	Chua '877
		Chua '877 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.") Chua '877 at 31:30-47 ("SDN controller 112 may then program network devices of the SDN to enforce the policies (400). For example, SDN controller 112 may program the network devices to direct packet flows including client device 102 along various paths, depending on the applicable policies for the packet flows. That is, certain packet flows originating from client device 102 or destined for client device 102 may be forwarded along different paths than other packet flows originating from or destined for client device 102, based on, e.g., the user of client device 102, an application, corresponding to the packet flow, that is being executed by client device 102, and/or a geographic location of client device 102. Likewise, packet flows for the same user using client device 102 and the same appli-cation may be directed along different paths through SDN 106 based on the geographic location of client device 102. Again, the policies may correspond to any combination of these factors, and may include additional or alternative fac-tors in other examples.")

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1160 of 2137

No.	'111 Patent Claim 1	Chua '877
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.	Chua '877 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. For example, Chua '877 discloses a network device sending a packet to a service device, if the packet's header data matches the program policy. Chua '877 at 2:49-61 ("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 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.") 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: Allow-explicitly allow a certain network flow to proceed to its destination Block-explicitly block a certain flow from traversing SDN 106 Mirror-allow the traffic, but send a copy of the traffic for deeper inspection or recording to, e.g., one of service devices 116 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. Transform-modify or translate values of headers of packets in the network flow

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1161 of 2137

No.	'111 Patent Claim 1	Chua '877
		Encapsulate-encapsulate packets in the network flow with a particular header") 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.") Chua '877 at 26:23-43 ("SDN controller 112 may then program the network devices to forward traffic along the determined paths (310). In con-ventional routers, control plane elements of a router program a forwarding information base (FIB) of a data plane of the router. However, SDN controller 112 represents a control plane element that can program data plane elements (e.g., switches) remotely, that is, without being part of or even directly coupled to the data plane elements. For example, SDN controller 112 may program any or all of network devices 108,110 and service devices 116 to forward network traffic having particular header data (e.g., source IP address, destination IP address, source port, destination port, protocol, MPLS label(s), VLAN tags, or the like) to a particular next hop (that is, next network device). Moreover, in accordance with the techniques of this disclosure, SDN controller 112 may program network devices of an SDN to forward traffic according to network paths that were determined based on one or more of connections determined between the network devices, service devices among the network devices, zones for packet flows through the network devices, and/or trusted or untrusted packet flows.") Chua '877 at 27:56-28:6 ("In addition, SDN controller 112 may program network devices to receive packets from the service device (332). More particularly, programming the network

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1162 of 2137

No.	'111 Patent Claim 1	Chua '877
		devices may include programming network devices to receive packets from a service device and forward the packets along the path that was determined for the packet flow corresponding to the packets. In some instances, the network device that forwards a packet to a service device may be the same network device that receives the packet from the service device. In such instances, SDN controller 112 may program a forwarding information base (FIB) of the network device such that the network device analyzes not only a packet header of the packet to determine a next hop for the packet, but also an input interface (e.g., an input port) by which the packet is received. That is, the FIB may include a forwarding table that includes data defining a correspondence from packet header information and an input interface to an output interface, for a packet flow.") Chua '877 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.
IPR2023-00401, Page 1163 of 2137

No.	'111 Patent Claim 2	Chua '877
2[a]	The method according to claim 1, wherein the instruction is 'probe', 'mirror', or 'terminate' instruction, and	Chua '877 discloses the method according to claim 1, wherein the instruction is 'probe', 'mirror', or 'terminate' instruction. For example, Chua '877 discloses programming network nodes with redirecting, mirroring, and blocking programmed actions. 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: Allow-explicitly allow a certain network flow to proceed to its destination Block-explicitly block a certain flow from traversing SDN 106 Mirror-allow the traffic, but send a copy of the traffic for deeper inspection or recording to, e.g., one of service devices 116 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. Transform-modify or translate values of headers of packets in the network flow Encapsulate-encapsulate packets in the network flow with a particular header") Chua '877 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1164 of 2137

No.	'111 Patent Claim 2	Chua '877
		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.")
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.	Chua '877 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. For example, Chua '877 discloses a blocking program in which the network device explicitly blocks a packet flow from traversing SDN 106. 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: Allow-explicitly allow a certain network flow to proceed to its destination Block-explicitly block a certain flow from traversing SDN 106 Mirror-allow the traffic, but send a copy of the traffic for deeper inspection or recording to, e.g., one of service devices 116 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.
IPR2023-00401, Page 1165 of 2137

No.	'111 Patent Claim 2	Chua '877
		Transform-modify or translate values of headers of packets in the network flow Encapsulate-encapsulate packets in the network flow with a particular header") 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.") 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.")

~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1166 of 2137

No.	'111 Patent Claim 2	Chua '877
		Chua '877 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.")

No.	'111 Patent Claim 3	Chua '877
3[a]	The method according to claim 1, wherein the instruction is a 'probe', a 'mirror', or a 'terminate' instruction, and	Chua '877 discloses the method according to claim 1, wherein the instruction is a 'probe', a 'mirror', or a 'terminate' instruction. *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	Chua '877 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. For example, Chua '877 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.
IPR2023-00401, Page 1167 of 2137

No.	'111 Patent Claim 3	Chua '877
	sending the packet, by the network node, to the second entity and to the controller.	to a second service device while forwarding the packets of the packet flow to the destination of the packet flow. 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: Allow-explicitly allow a certain network flow to proceed to its destination Block-explicitly block a certain flow from traversing SDN 106 Mirror-allow the traffic, but send a copy of the traffic for deeper inspection or recording to, e.g., one of service devices 116 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. Transform-modify or translate values of headers of packets in the network flow Encapsulate-encapsulate packets in the network flow with a particular header") 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1168 of 2137

No.	'111 Patent Claim 3	Chua '877
		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.") 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.") Chua '877 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1169 of 2137

No.	'111 Patent Claim 3	Chua '877
		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.")

No.	'111 Patent Claim 4	Chua '877
4[a]	The method according to claim 1, wherein the instruction is 'probe', 'mirror', or 'terminate' instruction, and	Chua '877 discloses the method according to claim 1, wherein the instruction is 'probe', 'mirror', or 'terminate' instruction. *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;	Chua '877 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. For example, Chua '877 discloses a redirect program in which responsive to analysis of the packet by a network device, the network device sends certain packets to the controller. 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: Allow-explicitly allow a certain network flow to proceed to its destination Block-explicitly block a certain flow from traversing SDN 106 Mirror-allow the traffic, but send a copy of the traffic for deeper inspection or recording to, e.g., one of service devices 116 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.
IPR2023-00401, Page 1170 of 2137

No.	'111 Patent Claim 4	Chua '877
		Transform-modify or translate values of headers of packets in the network flow Encapsulate-encapsulate packets in the network flow with a particular header") 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.") Chua '877 at 24:48-63 ("Initially, SDN controller 112 determines connections between network devices of an SDN (300). For example, network devices of the SDN (such as network devices 108, 110, which may correspond to network switches) may be configured to send acknowledgements to SDN controller 112 in response to receiving packets from SDN controller 112, as well as to forward the packets to other network devices to which the network devices are communicatively coupled. By analyzing the responses from the network devices, SDN con-troller 112 may determine connections between network devices of an SDN. For example, all ports on one network device 110 might be programmed to send all incoming pack-ets of a certain type to the SDN Controller 112. By sending an appropriately crafted packet (of the same type) on all ports of 108, the SDN Controller can determine how Network Device 108 is connected to Network Device 110.") Chua '877 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1171 of 2137

No.	'111 Patent Claim 4	Chua '877
		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.") 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.")
4[c]	responsive to receiving the packet, analyzing the packet, by the controller;	Chua '877 discloses responsive to receiving the packet, analyzing the packet, by the controller. For example, Chua '877 discloses analyzing, including deep packet inspection, performed by the controller on packets received., i.e, responsive to receiving the packet, analyzing the packet, by the controller. 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1172 of 2137

No.	'111 Patent Claim 4	Chua '877
		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.") Chua '877 at 24:48-63 ("Initially, SDN controller 112 determines connections between network devices of an SDN (300). For example, network devices of the SDN (such as network devices 108, 110, which may correspond to network switches) may be configured to send acknowledgements to SDN controller 112 in response to receiving packets from SDN controller 112, as well as to forward the packets to other network devices to which the network devices are communicatively coupled. By analyzing the responses from the network devices, SDN con-troller 112 may determine connections between network devices of an SDN. For example, all ports on one network device 110 might be programmed to send all incoming pack-ets of a certain type to the SDN Controller 112. By sending an appropriately crafted packet (of the same type) on all ports of 108, the SDN Controller can determine how Network Device 108 is connected to Network Device 110.") 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1173 of 2137

No.	'111 Patent Claim 4	Chua '877
		any or all of determination of service devices, determination of zones, and/or determination of trusted packet flows.") Chua '877 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.") 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.")

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1174 of 2137

No.	'111 Patent Claim 4	Chua '877
4[d]	sending the packet, by the controller, to the network node; and	Chua '877 discloses sending the packet, by the controller, to the network node. For example, Chua '877 discloses the controller sending the packet back to the network device to forward according to the determined packet flow. A person of ordinary skill in the art would understand that a packet sent to the controller for analysis is then sent back to the network node. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Chua '151 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. 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.") Chua '877 at 24:48-63 ("Initially, SDN controller 112 determines connections between network devices of an SDN (300). For example, network devices of the SDN (such as network devices 108, 110, which may correspond to network switches) may be configured to send acknowledgements to SDN controller 112 in response to receiving packets from SDN controller 112, as well as to forward the packets to other network devices to which the network devices are communicatively coupled. By analyzing the responses from the network devices, SDN con-troller 112 may determine connections between network devices of an SDN. For example, all ports on one network device 110 might be programmed to send all incoming

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1175 of 2137

No.	'111 Patent Claim 4	Chua '877
		pack-ets of a certain type to the SDN Controller 112. By sending an appropriately crafted packet (of the same type) on all ports of 108, the SDN Controller can determine how Network Device 108 is connected to Network Device 110.") 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.") Chua '877 at 26:23-43 ("In addition, SDN controller 112 may program network devices to receive packets from the service device (332). More particularly, programming the network devices may include programming network devices to receive packets from a service device and forward the packets along the path that was determined for the packet flow corresponding to the packets. In some instances, the network device that forwards a packet to a service device may be the same network device that receives the packet from the service device. In such instances, SDN controller 112 may program a forwarding information base (FIB) of the network device such that the network device analyzes not only a packet header of the packet to determine a next hop for the packet, but also an input interface (e.g., an input port) by which the packet is received. That is, the FIB may include a forwarding table that includes data defining a correspondence from packet header information and an input interface to an output interface, for a packet flow.") 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1176 of 2137

No.	'111 Patent Claim 4	Chua '877
		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.") Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Chua '877 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 are examples of two such references. 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 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.") 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

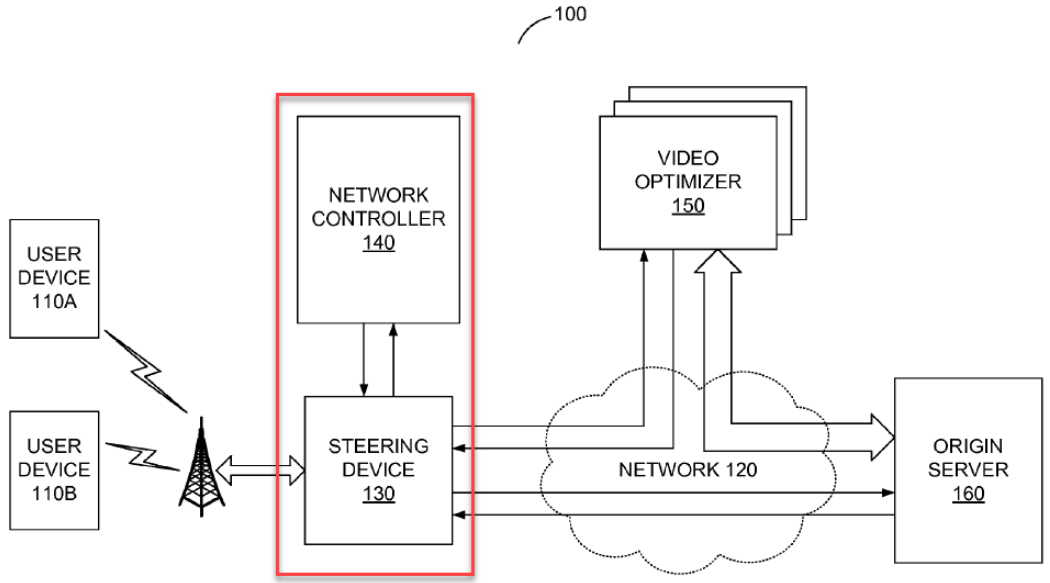
Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1177 of 2137

No.	'111 Patent Claim 4	Chua '877
		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.") 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.") 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,

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1178 of 2137

No.	'111 Patent Claim 4	Chua '877
		processing infrastructure (e.g., optimization services and network controllers) may be scaled independent from the steering devices handling raw HTTP throughput.") 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.") 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.") 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1179 of 2137

No.	'111 Patent Claim 4	Chua '877
		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.") Swenson at Figure 1 (annotation added)  FIG. 1
4[e]	responsive to receiving the packet, sending the packet, by the network	Chua '877 discloses responsive to receiving the packet, sending the packet, by the network node, to the second entity.

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1180 of 2137

No.	'111 Patent Claim 4	Chua '877
	node, to the second entity.	For example, Chua '877 discloses the controller sending the packet back to the network device to forward according to the determined packet flow. A person having ordinary skill in the art would understand that the packet would then be sent to its originally intended destination, 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 Chua '877 is found to not meet this limitation, responsive to receiving the packet, sending the packet, by the network node, to the second entity would have been obvious to a person having ordinary skill in the art, as explained below. 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.") Chua '877 at 24:48-63 ("Initially, SDN controller 112 determines connections between network devices of an SDN (300). For example, network devices of the SDN (such as network devices 108, 110, which may correspond to network switches) may be configured to send acknowledgements to SDN controller 112 in response to receiving packets from SDN controller 112, as well as to forward the packets to other network devices to which the network devices are communicatively coupled. By analyzing the responses from the network devices, SDN con-troller 112 may determine connections between network devices of an SDN. For example, all ports on one network device 110 might be programmed to send all incoming pack-ets of a certain type to the SDN Controller 112. By sending an appropriately crafted

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1181 of 2137

No.	'111 Patent Claim 4	Chua '877
		packet (of the same type) on all ports of 108, the SDN Controller can determine how Network Device 108 is connected to Network Device 110.") 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.") Chua '877 at 26:23-43 ("In addition, SDN controller 112 may program network devices to receive packets from the service device (332). More particularly, programming the network devices may include programming network devices to receive packets from a service device and forward the packets along the path that was determined for the packet flow corresponding to the packets. In some instances, the network device that forwards a packet to a service device may be the same network device that receives the packet from the service device. In such instances, SDN controller 112 may program a forwarding information base (FIB) of the network device such that the network device analyzes not only a packet header of the packet to determine a next hop for the packet, but also an input interface (e.g., an input port) by which the packet is received. That is, the FIB may include a forwarding table that includes data defining a correspondence from packet header information and an input interface to an output interface, for a packet flow.") 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.")

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1182 of 2137

No.	'111 Patent Claim 4	Chua '877
		Chua '877 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 are examples of two such references. Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, the Chua '877 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. 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. 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.") 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.
IPR2023-00401, Page 1183 of 2137

No.	'111 Patent Claim 4	Chua '877
		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.") 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.") 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.

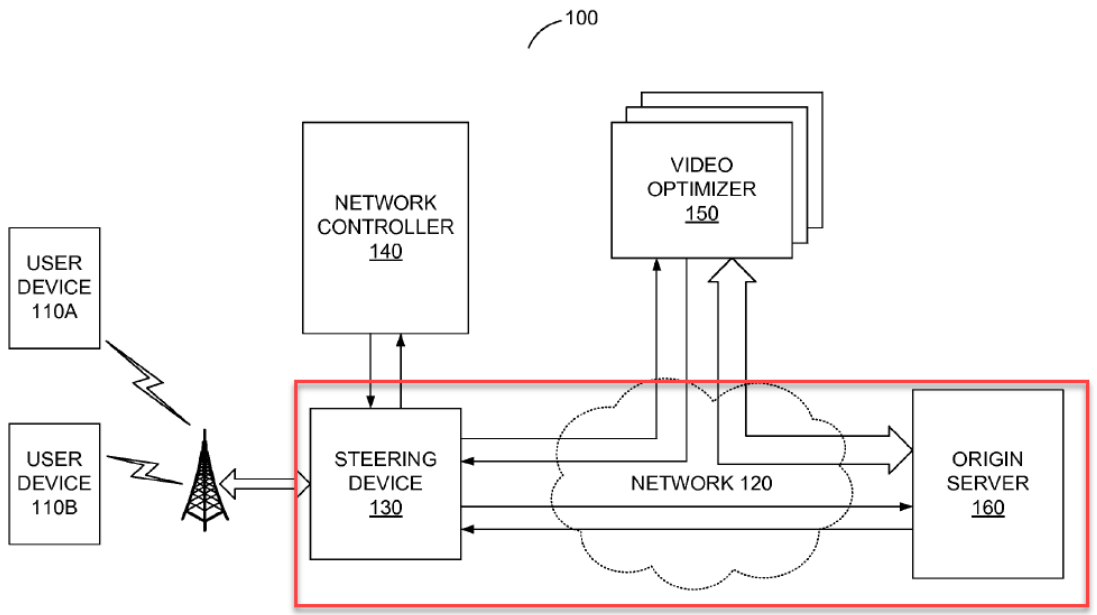
Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1184 of 2137

No.	'111 Patent Claim 4	Chua '877
		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.") 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.") 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.
IPR2023-00401, Page 1185 of 2137

No.	'111 Patent Claim 4	Chua '877
		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.") Swenson at Figure 1 (annotation added)

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1186 of 2137

No.	'111 Patent Claim 4	Chua '877
		 FIG. 1

No.	'111 Patent Claim 5	Chua '877
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.	Chua '877 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. For example, Chua '877 discloses in response to a program enforced by the network device on particular packet header data, sending the packet or data representative of the packet to the controller. *See supra* at Claim 1.

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1187 of 2137

No.	'111 Patent Claim 5	Chua '877
		Chua '877 at 2:38-47 ("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.") Chua '877 at 2:49-61 ("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 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.") Chua '877 at 2:62-3:6 ("In another example, a computer-readable storage medium has stored thereon instructions that, when executed, cause a processor of a controller device for a software defined network (SDN) to program a set of network devices of the SDN to form a path through the SDN and to send data representa-tive of packets sent along the path to the controller device, send 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.") 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1188 of 2137

No.	'111 Patent Claim 5	Chua '877
		perform pro-grammed actions on packets of a packet flow based on this data. Such programmed actions may include: Allow-explicitly allow a certain network flow to proceed to its destination Block-explicitly block a certain flow from traversing SDN 106 Mirror-allow the traffic, but send a copy of the traffic for deeper inspection or recording to, e.g., one of service devices 116 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. Transform-modify or translate values of headers of packets in the network flow Encapsulate-encapsulate packets in the network flow with a particular header") 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.") Chua '877 at 29:28-39 ("In this manner, the method of FIG. 7 represents an example of a method including 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1189 of 2137

No.	'111 Patent Claim 5	Chua '877
		forwarding the packets of the packet flow along the path based on data received from the set of network devices, and presenting a report representative of the determination.")

No.	'111 Patent Claim 6	Chua '877
6	The method according to claim 5, further comprising storing the received packet or a portion thereof, by the controller, in a memory.	Chua '877 discloses the method according to claim 5, further comprising storing the received packet or a portion thereof, by the controller, in a memory. For example, Chua '877 discloses logging and storing the packets, representative data of the packets, paths, and programs. *See supra* at Claim 5. 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: Allow-explicitly allow a certain network flow to proceed to its destination Block-explicitly block a certain flow from traversing SDN 106 Mirror-allow the traffic, but send a copy of the traffic for deeper inspection or recording to, e.g., one of service devices 116

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1190 of 2137

No.	'111 Patent Claim 6	Chua '877
		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. Transform-modify or translate values of headers of packets in the network flow Encapsulate-encapsulate packets in the network flow with a particular header") 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.") 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.") 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1191 of 2137

No.	'111 Patent Claim 6	Chua '877
		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.")

No.	'111 Patent Claim 7	Chua '877
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.	Chua '877 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. For example, Chua '877 discloses in response to a program enforced by the network device on particular packet header data, sending data representative of the packet to the controller. *See supra* at Claim 5. Chua '877 at 2:38-47 ("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.") Chua '877 at 2:49-61 ("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 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1192 of 2137

No.	'111 Patent Claim 7	Chua '877
		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.") Chua '877 at 2:62-3:6 ("In another example, a computer-readable storage medium has stored thereon instructions that, when executed, cause a processor of a controller device for a software defined network (SDN) to program a set of network devices of the SDN to form a path through the SDN and to send data representa-tive of packets sent along the path to the controller device, send 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.") 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: Allow-explicitly allow a certain network flow to proceed to its destination Block-explicitly block a certain flow from traversing SDN 106 Mirror-allow the traffic, but send a copy of the traffic for deeper inspection or recording to, e.g., one of service devices 116 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. Transform-modify or translate values of headers of packets in the network flow Encapsulate-encapsulate packets in the network flow with a particular header") 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1193 of 2137

No.	'111 Patent Claim 7	Chua '877
		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.") 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.") Chua '877 at 29:28-39 ("In this manner, the method of FIG. 7 represents an example of a method including 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,

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1194 of 2137

No.	'111 Patent Claim 7	Chua '877
		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 presenting a report representative of the determination.")

No.	'111 Patent Claim 8	Chua '877
8[a]	The method according to claim 7, wherein the portion of the packet consists of multiple consecutive bytes, and	Chua '877 discloses the method according to claim 7, wherein the portion of the packet consists of multiple consecutive bytes. For example, Chua '877 discloses representative data of packets of various byte sizes and counts. A person of ordinary skill in the art would understand that packets consist of multiple consecutive bytes comprising a portion of a packet. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Chua at '877 is found to not meet this limitation, wherein the portion of the packet consists of multiple consecutive byte would have been obvious to a person having ordinary skill in the art, as explained below. *See supra* at Claim 7. Chua '877 at 13:42-53 ("Control unit 130 may also maintain data representative of a history of statistics for elements of SDN 106, such as switches, switch ports, flows, and flow groups. Statistics 146 of FIG. 2 represents this data. The statistics entry for each instance in time may be dynamically extensible to allow additional counters and other values to be added as the system is extended. Statistics 146 may include a set of data entries, where each entry includes date and time of the last update, packet count, byte count, received packets count, received byte count, transmitted packet count, transmitted byte count, receive errors, transmit errors, received frame errors, transmit frame errors, and collisions.") Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Chua '877 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.

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1195 of 2137

No.	'111 Patent Claim 8	Chua '877
		For example, Kempf discloses consecutive bytes of a packet header field. 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.") 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.") 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1196 of 2137

No.	'111 Patent Claim 8	Chua '877
		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.") 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.") 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.") 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1197 of 2137

No.	'111 Patent Claim 8	Chua '877
		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.") 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: Write-Metadata (GTP-TEID, 0x FFFFFFFF) Apply-Actions (Set-Output-Port LOCAL_GTP _U_ENCAP)") 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. 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.	Chua '877 discloses wherein the instruction comprises identification of the consecutive bytes in the packet. For example, Chua '877 discloses programming network devices by the controller based on information gathered from the byte size and counts of representative data of packets. A

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1198 of 2137

No.	'111 Patent Claim 8	Chua '877
		person of ordinary skill in the art would understand that the consecutive bytes of the packet are identified according to an instruction. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Chua '877 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. Chua '877 at 13:12-41 ("A Source object may define the parameters needed to specify the network traffic source. Each Source object may include data representative of an identifier (e.g., a text string name) for the source, and a set of one or more zones. In particular, one or more flows may be grouped into zones. The zones may represent administrative or security areas that ensure that network managers or customers carmot interfere with each other from a switch control or data plane perspec-tive. For example, a test or research SDN environment may operate entirely within a single zone, and a simple production deployment may have a secure administrative zone used for management purposes and an open operational zone used for dynamic network programming. Multi-tenant or multi-appli-cation deployments may utilize unique zones for each cus-tomer or function or application. There is no fundamental restriction to how zones can be partitioned. In some examples, zones are partitioned on a per IP address, per port, per VLAN, or possibly per subnet basis, but this concept can be usefully extended on a per protocol basis or to any other packet header field or per application or per user. SDN con-troller 112 may ensure that a flow rule written for one zone incorporates sufficient additional match rules to prevent flows from other zones from being processed inappropriately. Zones may be ordered by numerical priority to avoid confu-sion that may otherwise occur when overlapping partitions are created. Each zone may support OpenFlow, normal dis-tributed, hybrid, or even alternate protocol types indepen-dently. Zones can also be used to partition off areas where no traffic is expected, allowing illegitimate traffic to be black holed, spanned, or redirected elsewhere for further action.") Chua '877 at 13:42-53 ("Control unit 130 may also maintain data representative of a history of statistics for elements of SDN 106, such as switches, switch ports, flows, and flow groups. Statistics 146 of FIG. 2 represents this data. The statistics entry for each instance in time may be dynamically extensible to allow additional counters and other values to be added as the system is extended. Statistics 146 may include a set of data entries, where each entry includes date and time of the last update, packet count, byte count, received packets count, received

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1199 of 2137

No.	'111 Patent Claim 8	Chua '877
		byte count, transmitted packet count, transmitted byte count, receive errors, transmit errors, received frame errors, transmit frame errors, and collisions.") Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Chua '877 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. For example, Kempf discloses rules in which the flow table includes matching to the consecutive bytes of a packet header. 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.") 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.
IPR2023-00401, Page 1200 of 2137

No.	'111 Patent Claim 8	Chua '877
		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.") 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.") 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.
IPR2023-00401, Page 1201 of 2137

No.	'111 Patent Claim 8	Chua '877
		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.") 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.") 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: Write-Metadata (GTP-TEID, 0x FFFFFFFF) Apply-Actions (Set-Output-Port LOCAL_GTP _U_ENCAP)") For example, Copeland discloses identifying the sampled packet headers comprised of fragmented packets of smaller byte sizes. 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.
IPR2023-00401, Page 1202 of 2137

No.	'111 Patent Claim 8	Chua '877
		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.") 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.") 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	Chua '877
9	The method according to claim 5, further comprising responsive to receiving the packet, analyzing the packet, by the controller.	Chua '877 discloses the method according to claim 5, further comprising responsive to receiving the packet, analyzing the packet, by the controller. For example, Chua '877 discloses analyzing, including deep packet inspection, performed by the controller on packets received, i.e, responsive to receiving the packet, analyzing the packet, by the controller. *See supra* at Claim 5. 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1203 of 2137

No.	'111 Patent Claim 9	Chua '877
		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.") 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 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.") 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).") 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.")

No.	'111 Patent Claim 9	Chua '877
		Chua '877 at 15:4-17 ("Another example use case for SDN controller 112 includes establishing a perimeter DMZ interconnection. Traffic entering a data center must often be distributed to a variety of appliances or server-based application-aware devices based on potentially deep inspection of incoming (and for certain flow types, outgoing) packets. Rather than route all traffic through L4-L 7 firewalls or application load balancers, or utilize distributed protocols such as policy based routing with limited capability and complex network interactions, SDN controller 112 may allow customers (e.g., administrator 114) to directly sort traffic on a per-packet or per-flow basis for individualized processing, saving cost, improving performance, limiting risk, improving availability, and increasing control.") Chua '877 at 15:49-63 ("In addition or in the alternative to the above, control unit 130 may be configured to utilize external triggers and information to change rules within programmable flow devices, e.g., switches of SDN 106. In particular, path determination unit 134 may be configured to utilize security information 148 to develop paths through SDN 106 to provide protection to SDN 106. SDN controller 112 may develop rules, stored in rules 142, that define triggers associated with events that cause control unit 130 to program network devices of SDN 106 via SDN interface 152 in response to the triggers. SDN controller 112 may program the network devices of SDN 106 such that, in response to one of the triggers "firing," that is, becoming activated, the network devices modify forwarding information for a packet flow related to the external trigger that fired.") 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. 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1205 of 2137

No.	'111 Patent Claim 9	Chua '877
		packet flow are prevented from reaching an intended destination. As noted above, zones may be ordered based on priority values when overlap occurs.") 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.")

No.	'111 Patent Claim 12	Chua '877
12	The method according to claim 9, wherein the analyzing comprises applying security or data analytic application.	Chua '877 discloses the method according to claim 9, wherein the analyzing comprises applying security or data analytic application. For example, Chua '877 discloses analyzing, including determining trusted packet flows for security purposes, performed by the controller on packets received., i.e, applying security or data analytic application. *See supra* at Claim 9. 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.")

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1206 of 2137

No.	'111 Patent Claim 12	Chua '877
		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. 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.") 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.") 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1207 of 2137

No.	'111 Patent Claim 12	Chua '877
		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.") 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.") 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.") 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1208 of 2137

No.	'111 Patent Claim 12	Chua '877
		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 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.") 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).") 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.") 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1209 of 2137

No.	'111 Patent Claim 12	Chua '877
		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.")

No.	'111 Patent Claim 13	Chua '877
13	The method according to claim 9, wherein the analyzing comprises applying security application that comprises firewall or intrusion detection functionality.	Chua '877 discloses the method according to claim 9, wherein the analyzing comprises applying security application that comprises firewall or intrusion detection functionality. For example, Chua '877 discloses security determinations and analysis that involve the use of firewalls or intrusion detection services, i.e, applying security application that comprises firewall or intrusion detection functionality. *See supra* at Claim 9. 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.") 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. 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1210 of 2137

No.	'111 Patent Claim 13	Chua '877
		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.") 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.") 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 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.
IPR2023-00401, Page 1211 of 2137

No.	'111 Patent Claim 13	Chua '877
		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.") 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.") 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 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1212 of 2137

No.	'111 Patent Claim 13	Chua '877
		trusted, whereas connections to network devices outside the enterprise network cannot be trusted and, therefore, should be monitored by a security device.") 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).") 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.") 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.
IPR2023-00401, Page 1213 of 2137

No.	'111 Patent Claim 13	Chua '877
		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	Chua '877
14	The method according to claim 9, wherein the analyzing comprises performing Deep Packet Inspection (DPI) or using a DPI engine on the packet.	Chua '877 discloses the method according to claim 9, wherein the analyzing comprises performing Deep Packet Inspection (DPI) or using a DPI engine on the packet. For example, Chua '877 discloses analyzing, including deep packet inspection, performed by the controller on packets received, i.e, analyzing comprises performing Deep Packet Inspection (DPI) or using a DPI engine on the packet. *See supra* at Claim 9. 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.") 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.")

No.	'111 Patent Claim 14	Chua '877
		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.") 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.")

No.	'111 Patent Claim 15	Chua '877
15[a]	The method according to claim 9, wherein the packet comprises distinct header and payload fields, and	Chua '877 discloses the method according to claim 9, wherein the packet comprises distinct header and payload fields. For example, Chua '877 discloses the network traffic having particular header data (e.g., source IP address, destination IP address, source port, destination port, protocol, MPLS label(s), VLAN tags, or the like). A person of ordinary skill in the art would understand that packets also have distinct payload fields to be forwarded to the intended destination.

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1215 of 2137

No.	'111 Patent Claim 15	Chua '877
		See supra at Claim 9. Chua '877 at 13:12-41 ("A Source object may define the parameters needed to specify the network traffic source. Each Source object may include data representative of an identifier (e.g., a text string name) for the source, and a set of one or more zones. In particular, one or more flows may be grouped into zones. The zones may represent administrative or security areas that ensure that network managers or customers cannot interfere with each other from a switch control or data plane perspec-tive. For example, a test or research SDN environment may operate entirely within a single zone, and a simple production deployment may have a secure administrative zone used for management purposes and an open operational zone used for dynamic network programming. Multi-tenant or multi-appli-cation deployments may utilize unique zones for each cus-tomer or function or application. There is no fundamental restriction to how zones can be partitioned. In some examples, zones are partitioned on a per IP address, per port, per VLAN, or possibly per subnet basis, but this concept can be usefully extended on a per protocol basis or to any other packet header field or per application or per user. SDN con-troller 112 may ensure that a flow rule written for one zone incorporates sufficient additional match rules to prevent flows from other zones from being processed inappropriately. Zones may be ordered by numerical priority to avoid confu-sion that may otherwise occur when overlapping partitions are created. Each zone may support OpenFlow, normal dis-tributed, hybrid, or even alternate protocol types indepen-dently. Zones can also be used to partition off areas where no traffic is expected, allowing illegitimate traffic to be black holed, spanned, or redirected elsewhere for further action.") 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 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1216 of 2137

No.	'111 Patent Claim 15	Chua '877
		trusted, whereas connections to network devices outside the enterprise network cannot be trusted and, therefore, should be monitored by a security device.") 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).") 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.") 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. 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.")

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1217 of 2137

No.	'111 Patent Claim 15	Chua '877
		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.") Chua '877 at 26:23-43 ("SDN controller 112 may then program the network devices to forward traffic along the determined paths (310). In conventional routers, control plane elements of a router program a forwarding information base (FIB) of a data plane of the router. However, SDN controller 112 represents a control plane element that can program data plane elements (e.g., switches) remotely, that is, without being part of or even directly coupled to the data plane elements. For example, SDN controller 112 may program any or all of network devices 108, 110 and service devices 116 to forward network traffic having particular header data (e.g., source IP address, destination IP address, source port, destination port, protocol, MPLS label(s), VLAN tags, or the like) to a particular next hop (that is, next network device). Moreover, in accordance with the techniques of this disclosure, SDN controller 112 may program network devices of an SDN to forward traffic according to network paths that were determined based on one or more of connections determined between the network devices, service devices among the network devices, zones for packet flows through the network devices, and/or trusted or untrusted packet flows.")
15[b]	wherein the analyzing comprises checking part of, or whole of, the payload field.	Chua '877 discloses wherein the analyzing comprises checking part of, or whole of, the payload field. For example, Chua '877 discloses analysis of packets in which the controller inspects packets of packet flows . A person of ordinary skill in the art would understand that analysis of a packet, including by deep packet inspect, checks the payload field of a packet to determine whether the content of a packet is malicious.

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1218 of 2137

No.	'111 Patent Claim 15	Chua '877
		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.")

No.	'111 Patent Claim 16	Chua '877
16[a]	The method according to claim 1, wherein the packet comprises distinct header and payload fields,	Chua '877 discloses the method according to claim 1, wherein the packet comprises distinct header and payload fields. *See supra* at Claim 1, 15[a].
16[b]	the header comprises one or more flag bits, and	Chua '877 discloses the header comprises one or more flag bits. For example, Chua '877 discloses packets with packet header field values represented in various ways. A person of ordinary skill in the art would understand that a header can 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 Chua '877 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. Chua '877 at 12:24-39 ("A FlowChain object may define the configuration data needed to configure and manage a set of flows (that is, packet flows) through the network infrastructure (e.g., through SDN 106). Each FlowChain object may include an identifier (e.g., a name), a Source object, one or more FlowLink objects, a flag to indicate whether the FlowChain is enabled or disabled, a flag to indicate whether the FlowChain is bi-directional, and a set of statistics for the FlowChain. The FlowChain may be defined using a set of configuration

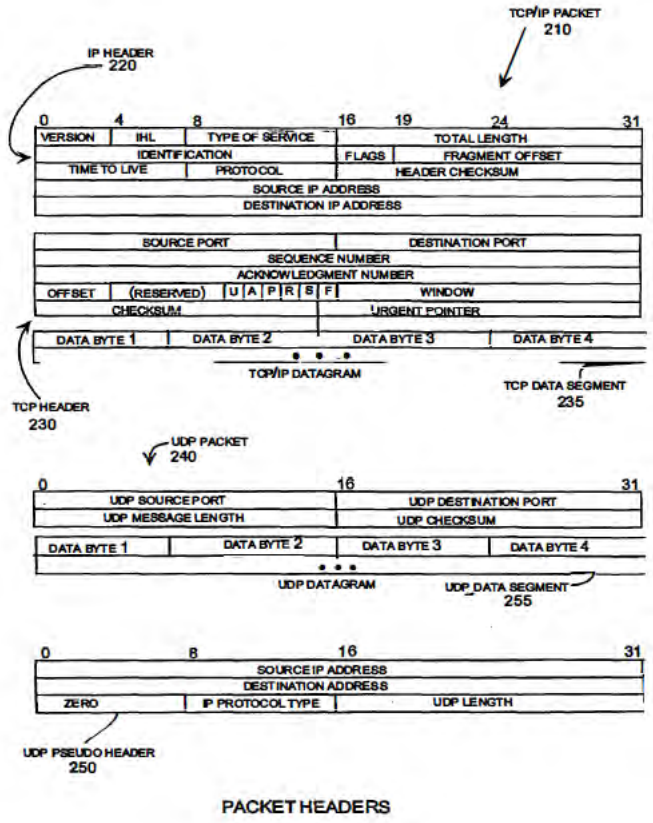
Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1219 of 2137

No.	'111 Patent Claim 16	Chua '877
		commands, e.g., extensible markup language (XML) commands, JavaScript Object Notation (JSON) commands, or other commands. In some examples, the commands may be natural language friendly. That is, the commands may be easily read and interpreted by a human operator. Such commands may benefit users in interacting with a device for creating a FlowChain, as well as other users, e.g., for audits and validation.") 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. 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.") Chua '877 at 26:23-43 ("SDN controller 112 may then program the network devices to forward traffic along the determined paths (310). In con-ventional routers, control plane elements of a router program a forwarding information base (FIB) of a data plane of the router. However, SDN controller 112 represents a control plane element that can program data plane elements (e.g., switches) remotely, that is, without being part of or even directly coupled to the data plane elements. For example, SDN controller 112 may program any or all of network devices 108,110 and service devices 116 to forward network traffic having particular header data (e.g., source IP address, destination IP address, source port, destination port, protocol, MPLS label(s), VLAN tags, or the like) to a particular next hop (that is, next network device). Moreover, in accordance with the techniques of this disclosure, SDN controller 112 may program network devices of an SDN to forward traffic according to network paths that were determined based on one or more of connections determined between the network devices,

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1220 of 2137

No.	'111 Patent Claim 16	Chua '877
		service devices among the network devices, zones for packet flows through the network devices, and/or trusted or untrusted packet flows.") Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Chua '877 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. For example, Copeland discloses packet headers with flag bits. Copeland at Figure 2

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1221 of 2137

No.	'111 Patent Claim 16	Chua '877
		 FIG. 2 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.
IPR2023-00401, Page 1222 of 2137

No.	'111 Patent Claim 16	Chua '877
		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.") 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.") 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 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.") 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.
IPR2023-00401, Page 1223 of 2137

No.	'111 Patent Claim 16	Chua '877
		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.") 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.") As another example, Kempf discloses packet headers with flag bits. 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.") 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.
IPR2023-00401, Page 1224 of 2137

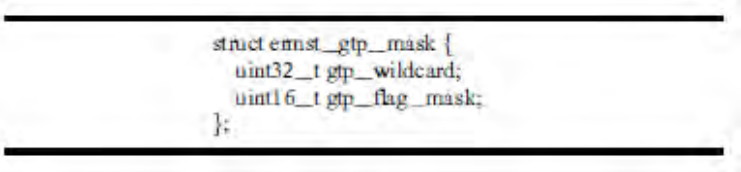
No.	'111 Patent Claim 16	Chua '877
		that they can be implemented in hardware or firmware. For this reason, GTP virtual ports may have the following restrictions on traffic they will handle: Protocol Type (PT) field= 1, where GTP encapsulation ports only sup-port GTP, not GTP' (PT field=0); Extension 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.") 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.") 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.") 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.
IPR2023-00401, Page 1225 of 2137

No.	'111 Patent Claim 16	Chua '877
		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.") 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).") 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.") 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.
IPR2023-00401, Page 1226 of 2137

No.	'111 Patent Claim 16	Chua '877

		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:
		 
		struct ermst_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.")

16[c]	wherein the packet-applicable criterion is that one or more of the flag bits is set.	Chua '877 discloses wherein the packet-applicable criterion is that one or more of the flag bits is set. For example, Chua '877 discloses specific packet header data used by the network devices to enforce programs, included flag bits of the packet header. A person of ordinary skill in the art would understand that whether flag bits are set of a packet header may constitute the packet header data used by network devices to determine enforcement of a program or perform an action. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Chua '877 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.
IPR2023-00401, Page 1227 of 2137

No.	'111 Patent Claim 16	Chua '877
		Chua '877 at 26:23-43 ("SDN controller 112 may then program the network devices to forward traffic along the determined paths (310). In con-ventional routers, control plane elements of a router program a forwarding information base (FIB) of a data plane of the router. However, SDN controller 112 represents a control plane element that can program data plane elements (e.g., switches) remotely, that is, without being part of or even directly coupled to the data plane elements. For example, SDN controller 112 may program any or all of network devices 108,110 and service devices 116 to forward network traffic having particular header data (e.g., source IP address, destination IP address, source port, destination port, protocol, MPLS label(s), VLAN tags, or the like) to a particular next hop (that is, next network device). Moreover, in accordance with the techniques of this disclosure, SDN controller 112 may program network devices of an SDN to forward traffic according to network paths that were determined based on one or more of connections determined between the network devices, service devices among the network devices, zones for packet flows through the network devices, and/or trusted or untrusted packet flows.") Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Chua '877 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. For example, Copeland discloses packet specific characteristics including flag bits that are set. 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1228 of 2137

No.	'111 Patent Claim 16	Chua '877
		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.") 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.") 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.") 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).") 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.
IPR2023-00401, Page 1229 of 2137

No.	'111 Patent Claim 16	Chua '877
		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.") As another example, Kempf flow table matches in which the flag bits is set. 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.") 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.") 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)

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1230 of 2137

No.	'111 Patent Claim 16	Chua '877
		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.") 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.") 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.") 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1231 of 2137

No.	'111 Patent Claim 16	Chua '877
		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).") 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.") 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.") 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.
IPR2023-00401, Page 1232 of 2137

No.	'111 Patent Claim 16	Chua '877

```
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.
IPR2023-00401, Page 1233 of 2137

No.	'111 Patent Claim 17	Chua '877
17[a]	The method according to claim 16, wherein the packet is an Transmission Control Protocol (TCP) packet, and	Chua '877 discloses the method according to claim 16, wherein the packet is an Transmission Control Protocol (TCP) packet. For example, Chua '877 discloses packet traffic using TCP services, i,e., the packet is an Transmission Control Protocol (TCP) packet. *See supra* at Claim 16. Chua '877 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. 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: 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 Create either UDP packet flows or TCP sessions that confirm to the traffic type inferred or discovered form the flowchain configuration 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) 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1234 of 2137

No.	'111 Patent Claim 17	Chua '877
		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 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.	Chua '877 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. For example, Chua '877 discloses specific packet header data used by the network devices, which may include flag bits. A person of ordinary skill in the art would understand that flag bits could 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 Chua '877 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. Chua '877 at 26:23-43 ("SDN controller 112 may then program the network devices to forward traffic along the determined paths (310). In con-ventional routers, control plane elements of a router program a forwarding information base (FIB) of a data plane of the router. However, SDN controller 112 represents a control plane element that can program data plane elements (e.g., switches) remotely, that is, without being part of or even directly coupled to the data plane elements. For example, SDN controller 112 may program any or all of network devices 108,110 and service devices 116 to forward network traffic having particular header data (e.g., source IP address, destination IP address, source port, destination port, protocol, MPLS label(s), VLAN tags, or the like) to a particular next hop (that is, next network device). Moreover, in accordance with the techniques of this disclosure, SDN controller 112 may program network devices of an SDN to forward traffic according to network paths that were determined based on one or more of connections determined between the network devices, service devices among the network devices, zones for packet flows through the network devices, and/or trusted or untrusted packet flows.")

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1235 of 2137

No.	'111 Patent Claim 17	Chua '877
		Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Chua '877 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 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.") 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1236 of 2137

No.	'111 Patent Claim 17	Chua '877
		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.") 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.") 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.") 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.") 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.") 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.") 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.
IPR2023-00401, Page 1237 of 2137

No.	'111 Patent Claim 17	Chua '877
		ACK packet providing to Host2 an acknowledgment number one greater than the sequence number provided in the FIN-ACK packet of Host2.") Uchida 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. As another example, Uchida discloses the TCP (Transmission Control Protocol) FIN flag, RST flag, and SYN flag. 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.") 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.") 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.") 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.")

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1238 of 2137

No.	'111 Patent Claim 17	Chua '877
		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.") 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.") 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,

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1239 of 2137

No.	'111 Patent Claim 17	Chua '877
		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.") 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).") 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.") 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).") 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1240 of 2137

No.	'111 Patent Claim 17	Chua '877
		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).") 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.") 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.") 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).") 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1241 of 2137

No.	'111 Patent Claim 17	Chua '877
		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).") 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.") 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.") 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.") 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.") 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.") 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1242 of 2137

No.	'111 Patent Claim 17	Chua '877
		Cl 7 refers to the packet storage, unit Cll, monitors a TCP SYN flag, and finds a SYN packet (step S301 in FIG. 14).") 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).") 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.") 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.") 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.")

No.	'111 Patent Claim 17	Chua '877
		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.") Uchida at [0877] ("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).") 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).") 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	Chua '877
18[a]	The method according to claim 1, wherein the packet comprises distinct header and payload fields,	Chua '877 discloses the method according to claim 1, wherein the packet comprises distinct header and payload fields. *See supra* at Claim 1, 15[a].

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1244 of 2137

No.	'111 Patent Claim 18	Chua '877
18[b]	the header comprises at least the first and second entities addresses in the packet network, and	Chua '877 discloses the header comprises at least the first and second entities addresses in the packet network. For example, Chua '877 discloses packet header data including source and destination address. Chua '877 at 22:53-23:4 ("The switch description objects may include data describing a primary controller for the respective switch, a switch identifier, a switch media access control (MAC) address for the respective switch, an Internet protocol (IP) address for the respective switch, and an array of port description objects for each physical port on the respective switch. For example, for switch 210, a switch description object may include data identifying SDN controller 218 as the primary controller for switch 210, an identifier for switch 210, a MAC address for switch 210, an IP address for switch 210, and an array of port description objects that describe ports of switch 210, such as the ports forming connections 222, 224, 226, and 228. Each of the port description objects may include data describing a port number for the respective port, a port identifier for the respective port, a MAC address of the respective port, and a description of capabilities for the respective port (e.g., a media type for the respective port, a speed for the respective port, whether the respective port is half duplex or full duplex, whether the port supports auto-negotiation, or the like).") Chua '877 at 26:23-43 ("SDN controller 112 may then program the network devices to forward traffic along the determined paths (310). In con-ventional routers, control plane elements of a router program a forwarding information base (FIB) of a data plane of the router. However, SDN controller 112 represents a control plane element that can program data plane elements (e.g., switches) remotely, that is, without being part of or even directly coupled to the data plane elements. For example, SDN controller 112 may program any or all of network devices 108,110 and service devices 116 to forward network traffic having particular header data (e.g., source IP address, destination IP address, source port, destination port, protocol, MPLS label(s), VLAN tags, or the like) to a particular next hop (that is, next network device). Moreover, in accordance with the techniques of this disclosure, SDN controller 112 may program network devices of an SDN to forward traffic according to network paths that were determined based on one or more of connections determined between the network devices, service devices among the network devices, zones for packet flows through the network devices, and/or trusted or untrusted packet flows.")

~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1245 of 2137

No.	'111 Patent Claim 18	Chua '877
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.	Chua '877 discloses wherein the packet-applicable criterion is that the first entity address, the second entity address, or both match a predetermined address or addresses. For example, Chua '877 discloses packet header data used by the network devices, including source and destination addresses, to enforce policies programmed on the network devices for forwarding the packet. Chua '877 at 22:53-23:4 ("The switch description objects may include data describing a primary controller for the respective switch, a switch identifier, a switch media access control (MAC) address for the respective switch, an Internet protocol (IP) address for the respective switch, and an array of port description objects for each physical port on the respective switch. For example, for switch 210, a switch description object may include data identifying SDN controller 218 as the primary controller for switch 210, an identifier for switch 210, a MAC address for switch 210, an IP address for switch 210, and an array of port description objects that describe ports of switch 210, such as the ports forming connections 222, 224, 226, and 228. Each of the port description objects may include data describing a port number for the respective port, a port identifier for the respective port, a MAC address of the respective port, and a description of capabilities for the respective port (e.g., a media type for the respective port, a speed for the respective port, whether the respective port is half duplex or full duplex, whether the port supports auto-negotiation, or the like).") 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1246 of 2137

No.	'111 Patent Claim 18	Chua '877
		any or all of determination of service devices, determination of zones, and/or determination of trusted packet flows.") Chua '877 at 26:23-43 ("SDN controller 112 may then program the network devices to forward traffic along the determined paths (310). In con-ventional routers, control plane elements of a router program a forwarding information base (FIB) of a data plane of the router. However, SDN controller 112 represents a control plane element that can program data plane elements (e.g., switches) remotely, that is, without being part of or even directly coupled to the data plane elements. For example, SDN controller 112 may program any or all of network devices 108,110 and service devices 116 to forward network traffic having particular header data (e.g., source IP address, destination IP address, source port, destination port, protocol, MPLS label(s), VLAN tags, or the like) to a particular next hop (that is, next network device). Moreover, in accordance with the techniques of this disclosure, SDN controller 112 may program network devices of an SDN to forward traffic according to network paths that were determined based on one or more of connections determined between the network devices, service devices among the network devices, zones for packet flows through the network devices, and/or trusted or untrusted packet flows.")

No.	'111 Patent Claim 19	Chua '877
19	The method according to claim 18, wherein the addresses are Internet Protocol (IP) addresses.	Chua '877 discloses the method according to claim 18, wherein the addresses are Internet Protocol (IP) addresses. For example, Chua '877 discloses source IP and destination IP addresses. *See supra* at Claim 18. Chua '877 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.

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1247 of 2137

No.	'111 Patent Claim 19	Chua '877
		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: 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 Create either UDP packet flows or TCP sessions that confirm to the traffic type inferred or discovered form the flowchain configuration 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) 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 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.") Chua '877 at 22:53-23:4 ("The switch description objects may include data describing a primary controller for the respective switch, a switch identifier, a switch media access control (MAC) address for the respective switch, an Internet protocol (IP) address for the respective switch, and an array of port description objects for each physical port on the respective switch. For example, for switch 210, a switch description object may include data identifying SDN controller 218 as the primary controller for switch 210, an identifier for switch 210, a MAC address for switch 210, an IP address for switch 210, and an array of port description objects that describe ports of switch 210, such as the ports forming connections 222, 224, 226, and 228. Each of the port description objects may include data describing a port number for the respective port, a port identifier for the respective port, a MAC address of the respective port,

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1248 of 2137

No.	'111 Patent Claim 19	Chua '877
		and a description of capabilities for the respective port (e.g., a media type for the respective port, a speed for the respective port, whether the respective port is half duplex or full duplex, whether the port supports auto-negotiation, or the like).") 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. 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.") Chua '877 at 26:23-43 ("SDN controller 112 may then program the network devices to forward traffic along the determined paths (310). In con-ventional routers, control plane elements of a router program a forwarding information base (FIB) of a data plane of the router. However, SDN controller 112 represents a control plane element that can program data plane elements (e.g., switches) remotely, that is, without being part of or even directly coupled to the data plane elements. For example, SDN controller 112 may program any or all of network devices 108,110 and service devices 116 to forward network traffic having particular header data (e.g., source IP address, destination IP address, source port, destination port, protocol, MPLS label(s), VLAN tags, or the like) to a particular next hop (that is, next network device). Moreover, in accordance with the techniques of this disclosure, SDN controller 112 may program network devices of an SDN to forward traffic according to network paths that were determined based on one or more of connections determined between the network devices, service devices among the network devices, zones for packet flows through the network devices, and/or trusted or untrusted packet flows.")

~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1249 of 2137

No.	'111 Patent Claim 20	Chua '877
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	Chua '877 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. *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 predetermined value or values.	Chua '877 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. For example, Chua '877 discloses packet header data used by the network devices, including TCP values, to enforce policies programmed on the network devices for forwarding the packet, i,e., 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. A person of ordinary skill in the art would understand that TCP packet header data could include 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. 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1250 of 2137

No.	'111 Patent Claim 20	Chua '877
		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.") Chua '877 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. 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: 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 Create either UDP packet flows or TCP sessions that confirm to the traffic type inferred or discovered form the flowchain configuration 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) 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.
IPR2023-00401, Page 1251 of 2137

No.	'111 Patent Claim 20	Chua '877
		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.") 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. 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.") 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.") Chua '877 at 26:23-43 ("SDN controller 112 may then program the network devices to forward traffic along the determined paths (310). In con-ventional routers, control plane elements of a router program a forwarding information base (FIB) of a data plane of the router. However, SDN controller 112 represents a control plane element that can program data plane elements (e.g., switches) remotely, that is, without being part of or even directly coupled to

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1252 of 2137

No.	'111 Patent Claim 20	Chua '877
		the data plane elements. For example, SDN controller 112 may program any or all of network devices 108,110 and service devices 116 to forward network traffic having particular header data (e.g., source IP address, destination IP address, source port, destination port, protocol, MPLS label(s), VLAN tags, or the like) to a particular next hop (that is, next network device). Moreover, in accordance with the techniques of this disclosure, SDN controller 112 may program network devices of an SDN to forward traffic according to network paths that were determined based on one or more of connections determined between the network devices, service devices among the network devices, zones for packet flows through the network devices, and/or trusted or untrusted packet flows.")

No.	'111 Patent Claim 21	Chua '877
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.	Chua '877 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. For example, Chua '877 discloses transmitting packets in a network including a Local Area Network (LAN) and Internet service provider (ISP) network, 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. *See supra* at Claim 1. Chua '877 at 10:17-30 ("Using the OpenFlow or other SDN protocol, it is possible to set up the recognition of Extensible Authentication Proto-col (EAP) over Local Area Network (EAPO L) packets (transport mechanism for 802. lX protocol) and to create a conversation with the control platform to emulate an 802.lX exchange. In this manner, system 100 can perform authenti-cation via EAP methods (using certificates or passwords) to identify the end-point (e.g., user and client device, e.g., client device 102). Unlike 802.lX port-based security, however, the use of 802 .1 X in an SD N framework can be used to lock down specific MAC source addresses, and can apply per-flow (that is, per-packet-flow) admission

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1253 of 2137

No.	'111 Patent Claim 21	Chua '877
		controls, as well as per-flow service orchestration and insertion to scan the traffic for malicious or sensitive content, as dictated by a policy engine.") Chua '877 at 12:10-23 ("SDN model 140 may further include data representing a network topology for SDN 106. Rules 142 corresponds to data representing rules for generating flow rules. The network topology and rules of rules 142 may be defined by software-based objects, such as FlowChain objects, Flow Link objects, SubDevice objects, Device objects, DevicePort objects, Por-tRange objects, Source objects, Zone objects, SwitchPort objects, and Network objects. Modifier actions, that is, actions that modify the network, may also be modeled by software-based objects, such as push virtual local area network (VLAN) objects, pop VLAN objects, sniffer objects, splitter objects, set multiprotocol label switching (MPLS) label objects, push MPLS label, pop MPLS label, or other actions.") Chua '877 at 17:26-35 ("Events sent to SDN controller 112 may trigger one or more of the SDN controller 112 actions on the same or different network, switch, and/or port than the network, switch, and/or port the initial traffic came from. For example, for DDOS, the identification of the DDOS attack may trigger a block action on the network under attack, trigger a block action to the upstream internet service provider (ISP) network (or a spe-cific upstream device), and/or trigger a counter attack (redi-rection) from a different network, ISP, or network device (physical or virtual).")

No.	'111 Patent Claim 22	Chua '877
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.	Chua '877 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. For example, Chua '877 discloses network traffic flowing between client devices and server devices in both directions. *See supra* at Claim 1.

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1254 of 2137

No.	'111 Patent Claim 22	Chua '877
		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 116 may also additionally 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.")

No.	'111 Patent Claim 23	Chua '877
23[a]	The method according to claim 22, wherein the server device comprises a web server, and	Chua '877 discloses the method according to claim 22, wherein the server device comprises a web server. For example, Chua '877 discloses a server device including a web server. *See supra* at Claim 22. 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-

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1255 of 2137

No.	'111 Patent Claim 23	Chua '877
		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 additionally 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.") 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.") Chua '877 at 20:18-26 ("SDN controller 218 is configured to program various net-work devices, such as switch 210, in accordance with one or more of the techniques of this disclosure. Although the example of FIG. 3 illustrates only one switch, i.e., switch 210, it should be understood that system 200 may include many interconnected switches or other programmable network devices. Likewise, service devices, such as web proxy 212, IDS 214, and web server 216 may be coupled to various switches (and not necessarily the same switch).") Chua '877 at 20:37-52 ("A user, such as an administrator, determines service devices, such as web proxy 212, IDS 214, and web server 216, to which particular devices (e.g., users of web clients 202) request access. The user may interface with administrator workstation 220 to configure SDN controller 218, to provide information indicative of users and services required for those users. SDN controller 218, in turn, may develop packet flows that ensure web traffic from particular ones of web clients 202 (and, potentially, certain applications executed by web clients 202 and/or with certain destinations) are directed through appropriate service devices. In other words, based on the connections determined between network

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1256 of 2137

No.	'111 Patent Claim 23	Chua '877
		devices in an SDN of system 200, SDN controller 218 may determine one or more paths for network traffic between the network devices. Likewise, SDN controller 218 may program the network devices to direct network traffic along the paths.") Chua '877 at 21:17-31 ("SDN controller 218 may be configured to program a switch, such as switch 210, to send packets of a particular packet flow to a service device, e.g., one of web proxy 212, IDS 214, and web server 216. For example, switch 210 is communicatively coupled to web proxy 210 by connection 222, which may represent ports 7 and 8. One of web clients 202 may request data from a web server, which may be internal to or external to system 200. Web proxy 212 may be configured to act as a proxy device to the web server. That is, SDN controller 218 may program switch 210 to direct net-work traffic destined for the web server to web proxy 212 via connection 222, instead. Web proxy 212 may then act on behalf of the web server, e.g., by caching resources stored by the web server and responding to requests for the resources on behalf of the web server.") Chua '877 at 21:53-58 ("As yet another example, switch 210 is communicatively coupled to web server 216 via connection 228. Connection 228 may represent ports 4 and 6. Web server 216 may gener-ally store and/or manage network accessible resources, such as files, databases, printers, or the like. Web clients 202 may access web server 216 via the SDN of system 200.")
23[b]	wherein the client device comprises a smartphone, a tablet computer, a personal computer, a laptop computer, or a wearable computing device.	Chua '877 discloses wherein the client device comprises a smartphone, a tablet computer, a personal computer, a laptop computer, or a wearable computing device. For example, Chua '877 discloses a client device including a web client that is capable of connecting to the Internet. 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)

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1257 of 2137

No.	'111 Patent Claim 23	Chua '877
		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 additionally 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.") 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.") Chua '877 at 20:5-17 ("As shown in FIG. 3, web clients 202 connect to Internet 204. Internet 204 provides access to system 200, which implements the techniques of this disclosure. For example, network devices of system 200 form an SDN controlled by SDN controller 218. SDN controller 218 may generally correspond to SDN controller 112 of FIGS. 1 and 2. In particular, SDN controller 218 (in this example) programs one or more switches (e.g., Pica8 3290 switches, IBM G8264), such as switch 210, to direct traffic through an SDN to other switches, service devices (e.g., web proxy 212, IDS 214, and/or web server 216). In general, SDN controller 218 may be configured to perform any or all of the techniques described above, alone or in any combination.") Chua '877 at 20:37-52 ("A user, such as an administrator, determines service devices, such as web proxy 212, IDS 214, and web server 216, to which particular devices (e.g., users of web clients 202) request access. The user may interface with administrator workstation 220 to configure SDN controller 218, to provide information indicative of users and services required for those users. SDN controller 218, in turn, may develop packet flows that ensure

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1258 of 2137

No.	'111 Patent Claim 23	Chua '877
		web traffic from particular ones of web clients 202 (and, potentially, certain applications executed by web clients 202 and/or with certain destinations) are directed through appropriate service devices. In other words, based on the connections determined between network devices in an SDN of system 200, SDN controller 218 may determine one or more paths for network traffic between the network devices. Likewise, SDN controller 218 may program the network devices to direct network traffic along the paths.") Chua '877 at 21:17-31 ("SDN controller 218 may be configured to program a switch, such as switch 210, to send packets of a particular packet flow to a service device, e.g., one of web proxy 212, IDS 214, and web server 216. For example, switch 210 is communicatively coupled to web proxy 210 by connection 222, which may represent ports 7 and 8. One of web clients 202 may request data from a web server, which may be internal to or external to system 200. Web proxy 212 may be configured to act as a proxy device to the web server. That is, SDN controller 218 may program switch 210 to direct net-work traffic destined for the web server to web proxy 212 via connection 222, instead. Web proxy 212 may then act on behalf of the web server, e.g., by caching resources stored by the web server and responding to requests for the resources on behalf of the web server.") Chua '877 at 21:53-58 ("As yet another example, switch 210 is communicatively coupled to web server 216 via connection 228. Connection 228 may represent ports 4 and 6. Web server 216 may gener-ally store and/or manage network accessible resources, such as files, databases, printers, or the like. Web clients 202 may access web server 216 via the SDN of system 200.")

No.	'111 Patent Claim 24	Chua '877
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.	Chua '877 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. For example, Chua '877 discloses communication between the network devices and SDN controller using standard protocols such as OpenFlow, OSPF, BGP, etc. *See supra* at Claim 22.

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1259 of 2137

No.	'111 Patent Claim 24	Chua '877
		Chua '877 at 1:25-41 ("Recently, software defined networks have been developed as a way to deploy and operate networks and develop new networking applications. In general, software defined net-works involve the use of a standalone controller that performs the control functionality for a set of network devices. As an example of software defined networking, in the case of routing, rather than routers performing individual analyses to determine routes through the network, the controller can determine the routes and program other devices in the net-work to behave according to the determinations made by the controller. Different protocols may be used to implement software defined networking, including open protocols like OpenFlow, and proprietary protocols from network vendors. OpenFlow is described in McKeown et al., "OpenFlow: Enabling Innovation in Campus Networks," http://ww-w.openflow.org//documents/openflow-wp-latest.pdf, which is hereby incorporated by reference in its entirety.") Chua '877 at 6:36-52 ("Users of SDN controller 112 may be provided with the ability to create programs that are executable by SDN con-troller 112. For example, a software development kit (SDK) may be provided for creating programs that can be executed by SDN controller 112. In this manner, users or other third parties may utilize an environment to write applications that can be executed by SDN controller 112. For example, SDN controller 112 itself may host one or more L4-L 7 services, rather than (or in addition to) services provided by service devices 116. As another example, a L3 application (such as Quagga, a Layer 3 routing stack) may be executed by SDN controller 112. In this case, the Quagga routing stack would be running a standard routing protocol such as OSPF (Open Shortest Path Forwarding) or BGP (Border Gateway Protocol) and would update its routing tables like before, except that it would use the SDN controller 112 to command the network equipment to update their forwarding tables.") Chua '877 at 9:27-36 ("SDN controller 112 may provide a platform that encom-passes various capabilities for custom software-defined net-working deployments. Such capabilities may include: a flex-ible network controller with the ability to communicate using standards-based or proprietary control protocols to control network flows on networking devices, both physical and vir-tual; a standards-based policy engine tied into the network controller logic; an extensible API for programmatic access from applications; and/or additional networking services that can execute on SDN controller 112.")

~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1260 of 2137

No.	'111 Patent Claim 24	Chua '877
		Chua '877 at 9:63-10:3 ("Thus, this disclosure also describes certain techniques that may be used to overcome the above noted problems relating to access control in a software defined network. In accordance with these techniques, a platform (such as SDN controller 112) may have network flow control capabilities on one end (e.g., via the OpenFlow (or other) protocol, interacting with both physical and virtual network access devices (switches for instance)), and a policy and identity stack on the other end.") Chua '877 at 10:17-30 ("Using the OpenFlow or other SDN protocol, it is possible to set up the recognition of Extensible Authentication Proto-col (EAP) over Local Area Network (EAPO L) packets (transport mechanism for 802. lX protocol) and to create a conversation with the control platform to emulate an 802.lX exchange. In this manner, system 100 can perform authenti-cation via EAP methods (using certificates or passwords) to identify the end-point (e.g., user and client device, e.g., client device 102). Unlike 802.lX port-based security, however, the use of 802 .1 X in an SD N framework can be used to lock down specific MAC source addresses, and can apply per-flow (that is, per-packet-flow) admission controls, as well as per-flow service orchestration and insertion to scan the traffic for mali-cious or sensitive content, as dictated by a policy engine.")

No.	'111 Patent Claim 27	Chua '877
27	The method according to claim 1, wherein the network node comprises a router, a switch, or a bridge.	Chua '877 discloses the method according to claim 1, wherein the network node comprises a router, a switch, or a bridge. For example, Chua '877 discloses network devices which may comprise switches or routers. *See supra* at Claim 1. Chua '877 at 3:33-54 ("In another example, a method includes obtaining, by a controller device of a software defined network (SDN), data representative of physical network devices in the SDN, wherein the physical network devices comprise one or more switches and one or more controllers including the controller device, instantiating, by the controller device, software-based controller objects for each of the one or more controllers, wherein the software-based controller objects each comprise data representative of a respective

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1261 of 2137

No.	'111 Patent Claim 27	Chua '877
		instance name, a respective server hostname, and a respective server host port, instanti-ating, by the controller device, software-based switch description objects for each of the one or more switches, wherein the software-based switch description objects each comprise data representative of a respective primary controller corresponding to a controller of the one or more control-lers, a respective switch identifier, a respective switch media access control (MAC) address, a respective Internet protocol (IP) address, and a respective array of port description objects for each physical port on the respective switch, and manag-ing, by the controller device, at least some of the switches using the switch description objects and the controller objects.") Chua '877 at 3:55-4:10 ("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 obtain data representative of physical network devices in the SDN, wherein the physical network devices comprise one or more switches and one or more controllers including the controller device, instantiate software-based controller objects for each of the one or more controllers, wherein the software-based controller objects each comprise data representative of a respective instance name, a respective server hostname, and a respective server host port, instantiate software-based switch description objects for each of the one or more switches, wherein the software-based switch description objects each comprise data representative of a respective primary control-ler corresponding to a controller of the one or more control-lers, a respective switch identifier, a respective switch media access control (MAC) address, a respective Internet protocol (IP) address, and a respective array of port description objects for each physical port on the respective switch, and manag-ing, via the network interfaces, at least some of the switches using the switch description objects and the controller objects.") Chua '877 at 5:32-49 ("FIG. 1 is a block diagram illustrating an example system 100 in which various techniques of this disclosure may be used. In this example, system 100 includes software defined network (SDN) 106, which includes network devices 108, 110 and service devices 116. Network devices 108, 110 may comprise switches, and other devices (not shown). These network devices can be physical instantiations or virtual instantiations. SDN 106 may also include other types of devices, such as routers, load balancers, various L4-L 7 net-work devices, or even multi-tenant capable devices, among other network devices. Again,

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1262 of 2137

No.	'111 Patent Claim 27	Chua '877
		these network devices can be either of physical or virtual instantiations. In general, network devices ofSDN 106, such as network devices 108,110, can be programmed to forward network traffic. System 100 also includes software defined network (SDN) controller 112. Administrator 114 uses SDN controller 112 to control (that is, program) network devices of SDN 106, such as network devices 108, 110.") Chua '877 at 11:13-27 ("SDN modeling unit 132 receives data representative of physical network devices in SDN 106, such as network device 108, network device 110, and service devices 116 (FIG. 1). SDN modeling unit 132 may also receive data representative of controller devices, including SDN controller 112, that control devices of SDN 106. SDN modeling unit 132 forms SDN model 140, which includes data that represents the physical devices of SDN 106. SDN model 140 may include software-based objects (that is, data for software objects con-forming to the object oriented programming (OOP) para-digm) that represent the physical devices in a physical view, as well as a logical view of SDN 106. As noted above, the physical view includes devices such as controllers, switches, and devices connected to the switches (e.g., service devices 116) managed by controllers such as SDN controller 112.") Chua '877 at 19:60-20:4 ("FIG. 3 is a conceptual diagram illustrating an example 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, 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.") Chua '877 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1263 of 2137

No.	'111 Patent Claim 27	Chua '877
		as SDN controller 112, can program devices of the data plane, such as network switches, to perform the techniques of this disclosure.")

No.	'111 Patent Claim 28	Chua '877
28	The method according to claim 1, wherein the packet network is an Internet Protocol (IP) network, and the packet is an IP packet.	Chua '877 discloses the method according to claim 1, wherein the packet network is an Internet Protocol (IP) network, and the packet is an IP packet. For example, Chua '877 discloses communication of packet traffic using an IP network and IP packet values. *See supra* at Claim 1. Chua '877 at 3:55-4:10 ("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 obtain data representative of physical network devices in the SDN, wherein the physical network devices comprise one or more switches and one or more controllers including the controller device, instantiate software-based controller objects for each of the one or more controllers, wherein the software-based controller objects each comprise data representative of a respective instance name, a respective server hostname, and a respective server host port, instantiate software-based switch description objects for each of the one or more switches, wherein the software-based switch description objects each comprise data representative of a respective primary control-ler corresponding to a controller of the one or more control-lers, a respective switch identifier, a respective switch media access control (MAC) address, a respective Internet protocol (IP) address, and a respective array of port description objects for each physical port on the respective switch, and manag-ing, via the network interfaces, at least some of the switches using the switch description objects and the controller objects.")

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1264 of 2137

No.	'111 Patent Claim 28	Chua '877
		Chua '877 at 13:12-41 ("A Source object may define the parameters needed to specify the network traffic source. Each Source object may include data representative of an identifier (e.g., a text string name) for the source, and a set of one or more zones. In particular, one or more flows may be grouped into zones. The zones may represent administrative or security areas that ensure that network managers or customers carmot interfere with each other from a switch control or data plane perspec-tive. For example, a test or research SDN environment may operate entirely within a single zone, and a simple production deployment may have a secure administrative zone used for management purposes and an open operational zone used for dynamic network programming. Multi-tenant or multi-appli-cation deployments may utilize unique zones for each cus-tomer or function or application. There is no fundamental restriction to how zones can be partitioned. In some examples, zones are partitioned on a per IP address, per port, per VLAN, or possibly per subnet basis, but this concept can be usefully extended on a per protocol basis or to any other packet header field or per application or per user. SDN con-troller 112 may ensure that a flow rule written for one zone incorporates sufficient additional match rules to prevent flows from other zones from being processed inappropriately. Zones may be ordered by numerical priority to avoid confu-sion that may otherwise occur when overlapping partitions are created. Each zone may support OpenFlow, normal dis-tributed, hybrid, or even alternate protocol types indepen-dently. Zones can also be used to partition off areas where no traffic is expected, allowing illegitimate traffic to be black holed, spanned, or redirected elsewhere for further action.") 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1265 of 2137

No.	'111 Patent Claim 28	Chua '877
		blocking corresponding traf-fic, mirroring corresponding traffic, redirecting correspond-ing traffic.") Chua '877 at 22:53-23:4 ("The switch description objects may include data describing a primary controller for the respective switch, a switch identifier, a switch media access control (MAC) address for the respective switch, an Internet protocol (IP) address for the respective switch, and an array of port description objects for each physical port on the respective switch. For example, for switch 210, a switch description object may include data identifying SDN controller 218 as the primary controller for switch 210, an identifier for switch 210, a MAC address for switch 210, an IP address for switch 210, and an array of port description objects that describe ports of switch 210, such as the ports forming connections 222, 224, 226, and 228. Each of the port description objects may include data describing a port number for the respective port, a port identifier for the respective port, a MAC address of the respective port, and description of capabilities for the respective port (e.g., a media type for the respective port, a speed for the respective port, whether the respective port is half duplex or full duplex, whether the port supports auto-negotiation, or the like).") 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. 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.") Chua '877 at 26:23-43 ("SDN controller 112 may then program the network devices to forward traffic along the determined paths (310). In con-ventional routers, control plane

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1266 of 2137

No.	'111 Patent Claim 28	Chua '877
		elements of a router program a forwarding information base (FIB) of a data plane of the router. However, SDN controller 112 represents a control plane element that can program data plane elements (e.g., switches) remotely, that is, without being part of or even directly coupled to the data plane elements. For example, SDN controller 112 may program any or all of network devices 108,110 and service devices 116 to forward network traffic having particular header data (e.g., source IP address, destination IP address, source port, destination port, protocol, MPLS label(s), VLAN tags, or the like) to a particular next hop (that is, next network device). Moreover, in accordance with the techniques of this disclosure, SDN controller 112 may program network devices of an SDN to forward traffic according to network paths that were determined based on one or more of connections determined between the network devices, service devices among the network devices, zones for packet flows through the network devices, and/or trusted or untrusted packet flows.")

No.	'111 Patent Claim 29	Chua '877
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.	Chua '877 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. For example, Chua '877 discloses communication of packet traffic using a TCP network and TCP packet values. *See supra* at Claim 28. 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,

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1267 of 2137

No.	'111 Patent Claim 29	Chua '877
		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.") Chua '877 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. 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: 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 Create either UDP packet flows or TCP sessions that confirm to the traffic type inferred or discovered form the flowchain configuration 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) 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.
IPR2023-00401, Page 1268 of 2137

No.	'111 Patent Claim 29	Chua '877
		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.") Chua '877 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 network 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 servers), 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.") 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. 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.") Chua '877 at 26:23-43 ("SDN controller 112 may then program the network devices to forward traffic along the determined paths (310). In con-ventional routers, control plane

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1269 of 2137

No.	'111 Patent Claim 29	Chua '877
		elements of a router program a forwarding information base (FIB) of a data plane of the router. However, SDN controller 112 represents a control plane element that can program data plane elements (e.g., switches) remotely, that is, without being part of or even directly coupled to the data plane elements. For example, SDN controller 112 may program any or all of network devices 108,110 and service devices 116 to forward network traffic having particular header data (e.g., source IP address, destination IP address, source port, destination port, protocol, MPLS label(s), VLAN tags, or the like) to a particular next hop (that is, next network device). Moreover, in accordance with the techniques of this disclosure, SDN controller 112 may program network devices of an SDN to forward traffic according to network paths that were determined based on one or more of connections determined between the network devices, service devices among the network devices, zones for packet flows through the network devices, and/or trusted or untrusted packet flows.")

No.	'111 Patent Claim 30	Chua '877
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;	Chua '877 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. For example, Chua '877 discloses the transmission of, including receiving by the network devices, a second set of packet flows. *See supra at* Claim 1, 1[c]. 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).")

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1270 of 2137

No.	'111 Patent Claim 30	Chua '877
30[b]	checking, by the network node, if any one of the one or more additional packets satisfies the criterion;	Chua '877 discloses checking, by the network node, if any one of the one or more additional packets satisfies the criterion. *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	Chua '877 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. *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.	Chua '877 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. *See supra at* Claim 1[f], 30[a].

No.	'111 Patent Claim 31	Chua '877
31[a]	The method according to claim 1, wherein the packet network is a Software Defined Network (SDN),	Chua '877 discloses the method according to claim 1, wherein the packet network is a Software Defined Network (SDN). For example, Chua '877 discloses techniques and methods related to software defined networks (SDNs). *See supra* at Claim 1. Chua '877 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1271 of 2137

No.	'111 Patent Claim 31	Chua '877
		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.") Chua '877 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.") Chua '877 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.") Chua '877 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.") 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1272 of 2137

No.	'111 Patent Claim 31	Chua '877
		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	Chua '877 discloses the packet is routed as part of a data plane. For example, Chua '877 discloses forwarding packets over a data plane to various network destinations. Chua '877 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.") Chua '877 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.")

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1273 of 2137

No.	'111 Patent Claim 31	Chua '877
		Chua '877 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.") Chua '877 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.") Chua '877 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.
IPR2023-00401, Page 1274 of 2137

No.	'111 Patent Claim 31	Chua '877
31[c]	the network node communication with the controller serves as a control.	Chua '877 discloses the network node communication with the controller serves as a control. For example, Chua '877 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. Chua '877 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.") Chua '877 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.") Chua '877 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.") Chua '877 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 disclosure. Flow management system 250 (also referred to as

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1275 of 2137

No.	'111 Patent Claim 31	Chua '877
		"system 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.") Chua '877 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.
IPR2023-00401, Page 1276 of 2137

Chart for U.S. Patent 10,652,111 ("the '111 Patent")
U.S. Patent No. 9,038,151 to Chua '151 et al. ("Chua '151")

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 Chua '151 meets each element of those claims, and/or (2) 35 U.S.C. § 103 because Chua '151 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 Chua '151 titled "Authentication For Software Defined Neworks", which was filed on March 15, 2013, and issued on May 19, 2015.

Motivations to combine the disclosures in Chua '151 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 Preliminary Infringement Disclosure and its January 19, 2023 First Amended Preliminary Infringement Disclosures ("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.
IPR2023-00401, Page 1277 of 2137

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	Chua '151
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:	Chua '151 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. For example, Chua '151 discloses 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. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. Chua '151 at 1:10-11 ("This disclosure relates to computer networks and, in particular, software defined networks.") Chua '151 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.") Chua '151 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.") Chua '151 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

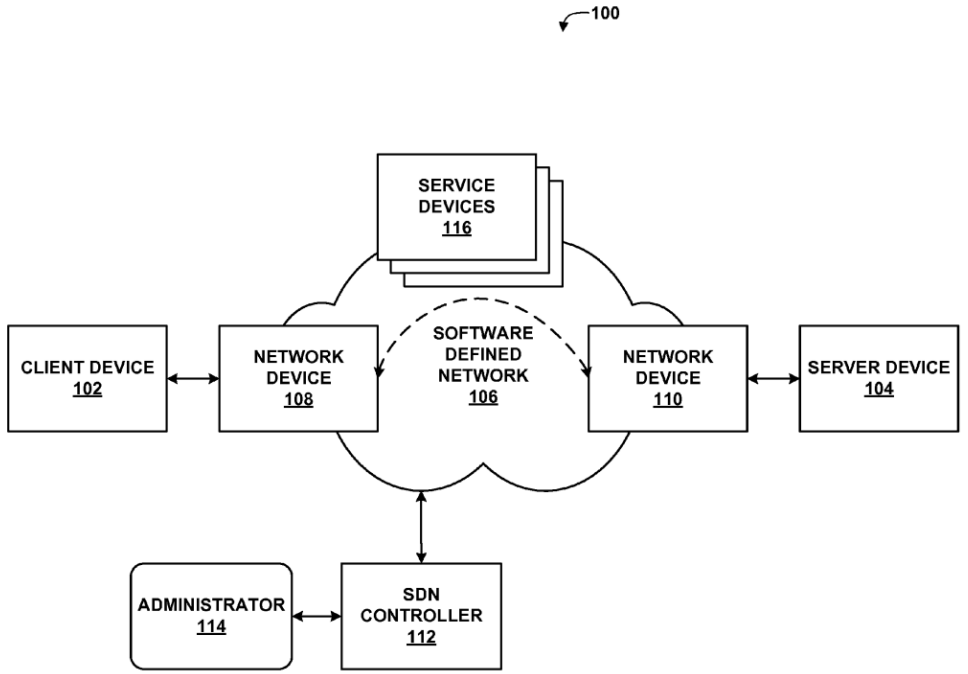
Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1278 of 2137

No.	'111 Patent Claim 1	Chua '151
		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.") Chua '151 at 5:32-49 ("FIG. 1 is a block diagram illustrating an example system 100 in which various techniques of this disclosure may be used. In this example, system 100 includes software defined network (SDN) 106, which includes network devices 108, 110 and service devices 116. Network devices 108, 110 may comprise switches, and other devices (not shown). These network devices can be physical instantiations or virtual instantiations. SDN 106 may also include other types of devices, such as routers, load balancers, various L4-L7 network devices, or even multi-tenant capable devices, among other network devices. Again, these network devices can be either of physical or virtual instantiations. In general, network devices of SDN 106, such as network devices 108,110, can be programmed to forward network traffic. System 100 also includes software defined network (SDN) controller 112. Administrator 114 uses SDN controller 112 to control (that is, program) network devices of SDN 106, such as network devices 108, 110.") Chua '151 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

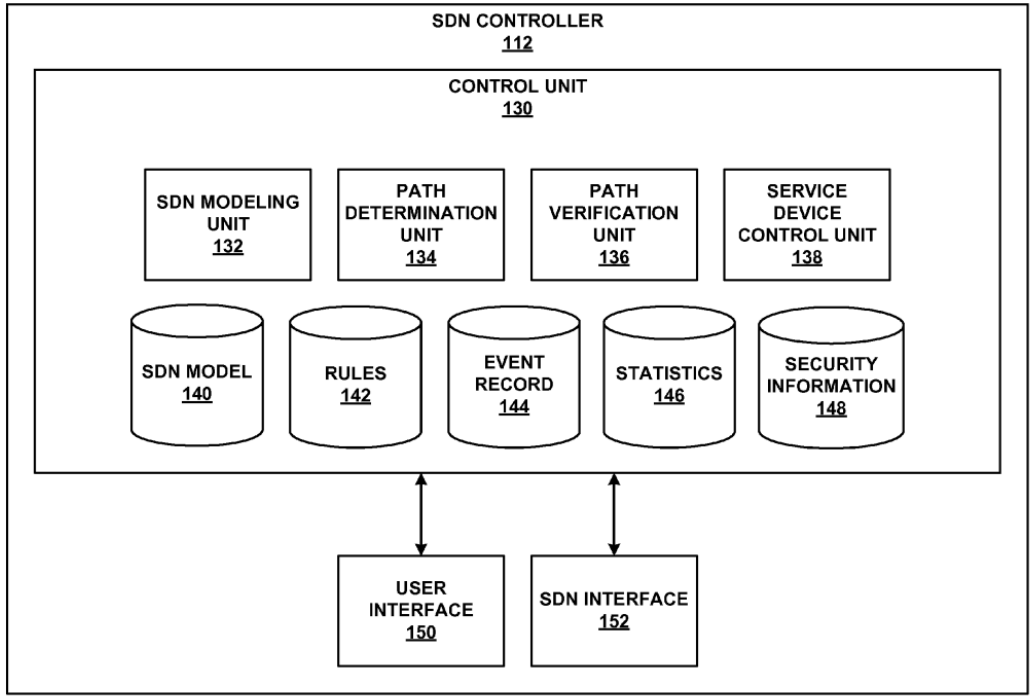
Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1279 of 2137

No.	'111 Patent Claim 1	Chua '151
		net devices, tar pit devices, domain name service (DNS) and global DNS server devices, mail proxies, and anti-spam devices.") Chua '151 at 20:5-18 ("As shown in FIG. 3, web clients 202 connect to Internet 204. Internet 204 provides access to system 200, which implements the techniques of this disclosure. For example, network devices of system 200 form an SDN controlled by SDN controller 218. SDN controller 218 may generally cor-respond to SDN controller 112 of FIGS. 1 and 2. In particular, SDN controller 218 (in this example) programs one or more switches (e.g., Pica8 3290 switches, IBM G8264), such as switch 210, to direct traffic through an SDN to other switches, service devices (e.g., web proxy 212, IDS 214, and/or webs erver 216). In general, SDN controller 218 may be config-ured to perform any or all of the techniques described above, alone or in any combination.") Chua '151 at 26:23-43 ("SDN controller 112 may then program the network devices to forward traffic along the determined paths (310). In con-ventional routers, control plane elements of a router program a forwarding information base (FIB) of a data plane of the router. However, SDN controller 112 represents a control plane element that can program data plane elements (e.g., switches) remotely, that is, without being part of or even directly coupled to the data plane elements. For example, SDN controller 112 may program any or all of network devices 108,110 and service devices 116 to forward network traffic having particular header data (e.g., source IP address, destination IP address, source port, destination port, protocol, MPLS label(s), VLAN tags, or the like) to a particular next hop (that is, next network device). Moreover, in accordance with the techniques of this disclosure, SDN controller 112 may program network devices of an SDN to forward traffic according to network paths that were determined based on one or more of connections determined between the network devices, service devices among the network devices, zones for packet flows through the network devices, and/or trusted or untrusted packet flows.") Chua '151 at Figure 1

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1280 of 2137

No.	'111 Patent Claim 1	Chua '151
		 FIG. 1 Chua '151 at Figure 2

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1281 of 2137

No.	'111 Patent Claim 1	Chua '151
		 FIG. 2
1[a]	sending, by the controller to the network node over the packet network, an instruction and a packet-applicable criterion;	Chua '151 discloses sending, by the controller to the network node over the packet network, an instruction and a packet-applicable criterion. For example, Chua '151 discloses SDN controller 112 make network enforcement decisions for specific traffic flows. The SDN controller 112 program network devices of SDN 106 to perform programmed actions on packets of a packet flow based on this data. Chua '151 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1282 of 2137

No.	'111 Patent Claim 1	Chua '151
		network devices based on the determination of the connections, and program the network devices to direct network traffic along the one or more paths") Chua '151 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.") Chua '151 at 2:49-61 ("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 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.") Chua '151 at 4:44-55 ("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 receive cre-dentials from a client device in accordance with a public key infrastructure (PKI)-based authentication protocol, deter-mine one or more policies that are applicable to the client device based on the received credentials, and program net-work devices of the SDN to enforce the determined policies on a per-packet-flow basis for packet flows including the client device.") Chua '151 at 5:32-49 ("FIG. 1 is a block diagram illustrating an example system 100 in which various techniques of this disclosure may be used. In this example, system 100 includes software defined network (SDN) 106, which includes network devices 108, 110 and service devices 116. Network devices 108, 110 may comprise switches, and other

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1283 of 2137

No.	’111 Patent Claim 1	Chua ’151
		devices (not shown). These network devices can be physical instantiations or virtual instantiations. SDN 106 may also include other types of devices, such as routers, load balancers, various L4-L7 network devices, or even multi-tenant capable devices, among other network devices. Again, these network devices can be either of physical or virtual instantiations. In general, network devices of SDN 106, such as network devices 108,110, can be programmed to forward network traffic. System 100 also includes software defined network (SDN) controller 112. Administrator 114 uses SDN controller 112 to control (that is, program) network devices of SDN 106, such as network devices 108, 110.”) Chua ’151 at 6:25-35 (“SDN controller 112 may implement any or all of the techniques described in this disclosure for controlling various devices of SDN 106, such as network devices 108, 110, and service devices 116. These techniques are described in greater detail below. Examples of the techniques described below include a data model for managing devices of an SDN, implementing high availability among devices of an SDN, performing services for packets flowing along a path through an SDN, trigger-driven policy distribution, and using test traffic to verify whether devices of the SDN are operating correctly.”) Chua ’151 at 7:28-54 (“SDN controller 112 may receive data as input from service devices 116. For example, SDN controller 112 may be configured 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 programmed actions on packets of a packet flow based on this data. Such programmed actions may include: Allow—explicitly allow a certain network flow to proceed to its destination Block—explicitly block a certain flow from traversing SDN 106 Mirror—allow the traffic, but send a copy of the traffic for deeper inspection or recording to, e.g., one of service devices 116 Redirect—redirect the traffic to another network device (such as a honeypot device or traffic

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1284 of 2137

No.	'111 Patent Claim 1	Chua '151
		device of service devices 116) for either inspection or to keep a potential hacker 'busy' to determine if there is a real security threat. Transform—modify or translate values of headers of packets in the network flow Encapsulate—encapsulate packets in the network flow with a particular header") Chua '151 at 21:17-31 ("SDN controller 218 may be configured to program a switch, such as switch 210, to send packets of a particular packet flow to a service device, e.g., one of web proxy 212, IDS 214, and web server 216. For example, switch 210 is communicatively coupled to web proxy 210 by connection 222, which may represent ports 7 and 8. One of web clients 202 may request data from a web server, which may be internal to or external to system 200. Web proxy 212 may be configured to act as a proxy device to the web server. That is, SDN controller 218 may program switch 210 to direct net-work traffic destined for the web server to web proxy 212 via connection 222, instead. Web proxy 212 may then act on behalf of the web server, e.g., by caching resources stored by the web server and responding to requests for the resources on behalf of the web server.") Chua '151 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. 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.")

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1285 of 2137

No.	'111 Patent Claim 1	Chua '151
		Chua '151 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.") Chua '151 at 31:30-47 ("SDN controller 112 may then program network devices of the SDN to enforce the policies (400). For example, SDN controller 112 may program the network devices to direct packet flows including client device 102 along various paths, depending on the applicable policies for the packet flows. That is, certain packet flows originating from client device 102 or destined for client device 102 may be forwarded along different paths than other packet flows originating from or destined for client device 102, based on, e.g., the user of client device 102, an application, corresponding to the packet flow, that is being executed by client device 102, and/or a geographic location of client device 102. Likewise, packet flows for the same user using client device 102 and the same appli-cation may be directed along different paths through SDN 106 based on the geographic location of client device 102. Again, the policies may correspond to any combination of these factors, and may include additional or alternative fac-tors in other examples.") Chua '151 at Figure 5 (annotation added)

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1286 of 2137

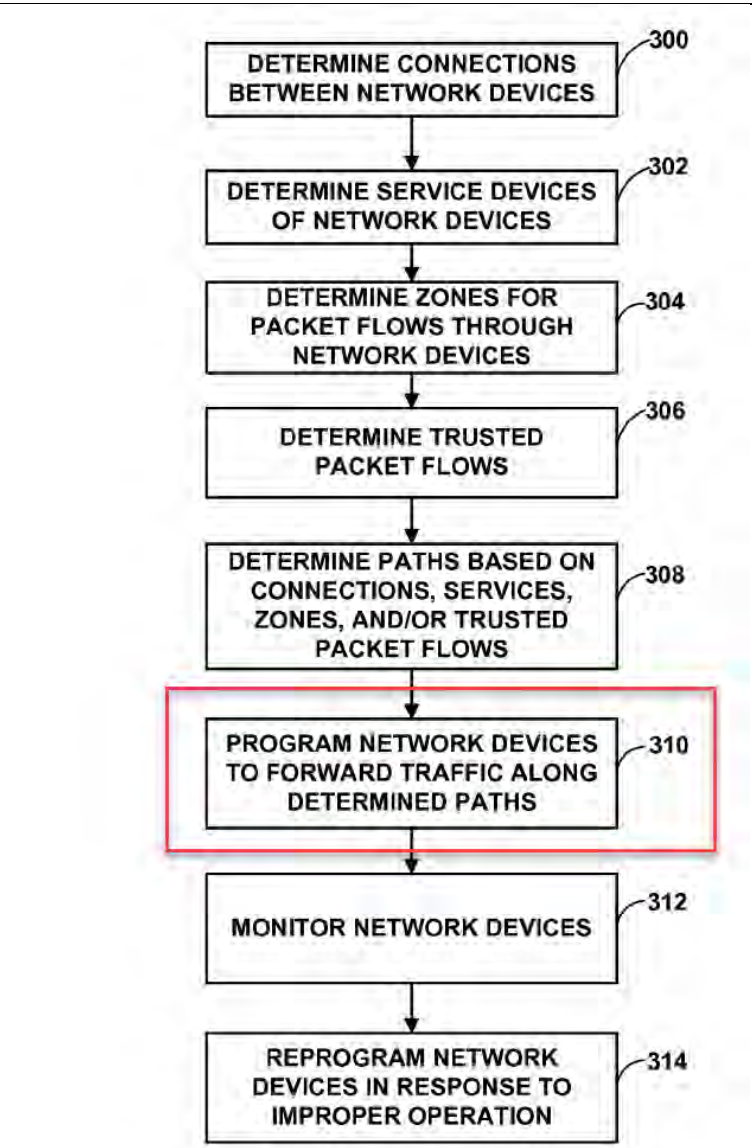

DETERMINE CONNECTIONS BETWEEN NETWORK DEVICES
300
DETERMINE SERVICE DEVICES OF NETWORK DEVICES
302
DETERMINE ZONES FOR PACKET FLOWS THROUGH NETWORK DEVICES
304
DETERMINE TRUSTED PACKET FLOWS
306
DETERMINE PATHS BASED ON CONNECTIONS, SERVICES, ZONES, AND/OR TRUSTED PACKET FLOWS
308
PROGRAM NETWORK DEVICES TO FORWARD TRAFFIC ALONG DETERMINED PATHS
310
MONITOR NETWORK DEVICES
312
REPROGRAM NETWORK DEVICES IN RESPONSE TO IMPROPER OPERATION
314

FIG. 5

No.	'111 Patent Claim 1	Chua '151
1[b]	receiving, by the network node from the controller, the instruction and the criterion;	Chua '151 discloses receiving, by the network node from the first entity over the packet network, a packet addressed to the second entity. *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;	Chua '151 discloses receiving, by the network node from the first entity over the packet network, a packet addressed to the second entity. For example, Chua '151 discloses forwarding network traffic received by network devices having particular header data (e.g., source IP address, destination IP address, source port, destination port, protocol, MPLS label(s), VLAN tags, or the like) to a particular next hop (that is, next network device). Chua '151 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") Chua '151 at 5:32-49 ("FIG. 1 is a block diagram illustrating an example system 100 in which various techniques of this disclosure may be used. In this example, system 100 includes software defined network (SDN) 106, which includes network devices 108, 110 and service devices 116. Network devices 108, 110 may comprise switches, and other devices (not shown). These network devices can be physical instantiations or virtual instantiations. SDN 106 may also include other types of devices, such as routers, load balancers, various L4-L 7 net-work devices, or even multi-tenant capable devices, among other network devices. Again, these network devices can be either of physical or virtual instantiations. In general, network devices of SDN 106, such as network devices 108,110, can be programmed to forward network traffic. System 100 also includes software defined network (SDN) controller 112. Administrator 114 uses SDN controller 112 to control (that is, program) network devices of SDN 106, such as network devices 108, 110.")

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1288 of 2137

No.	'111 Patent Claim 1	Chua '151
		Chua '151 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.") Chua '151 at 26:15-22 ("Determining the paths generally includes determining, from a first network device of the SDN, subsequent "next hop" network devices through the SDN. Thus, SDN control-ler 112 may analyze connections between the network devices (e.g., switches and service devices) and direct traffic from one device to another, assuming that the devices are communicatively coupled and cause the traffic to travel substantially in the direction of the destination of the traffic.") Chua '151 at 26:23-43 ("SDN controller 112 may then program the network devices to forward traffic along the determined paths (310). In conventional routers, control plane elements of a router program a forwarding information base (FIB) of a data plane of the router. However, SDN controller 112 represents a control plane element that can program data plane elements (e.g., switches) remotely, that is, without being part of or even directly coupled to the data plane elements. For example, SDN controller 112 may program any or all of network devices 108, 110 and service devices 116 to forward network traffic having particular header data (e.g., source IP address, destination IP address, source port, destination port, protocol, MPLS label(s), VLAN tags, or the like) to a particular next hop (that is, next network device). Moreover, in accordance with the techniques of this

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1289 of 2137

No.	'111 Patent Claim 1	Chua '151
		disclosure, SDN controller 112 may program network devices of an SDN to forward traffic according to network paths that were determined based on one or more of connections determined between the network devices, service devices among the network devices, zones for packet flows through the network devices, and/or trusted or untrusted packet flows.") Chua '151 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 packets with a particular header, or other such actions.") Chua '151 at 31:30-47 ("SDN controller 112 may then program network devices of the SDN to enforce the policies (400). For example, SDN controller 112 may program the network devices to direct packet flows including client device 102 along various paths, depending on the applicable policies for the packet flows. That is, certain packet flows originating from client device 102 or destined for client device 102 may be forwarded along different paths than other packet flows originating from or destined for client device 102, based on, e.g., the user of client device 102, an application, corresponding to the packet flow, that is being executed by client device 102, and/or a geographic location of client device 102. Likewise, packet flows for the same user using client device 102 and the same appli-cation may be directed along different paths through SDN 106 based on the

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1290 of 2137

No.	'111 Patent Claim 1	Chua '151
		geographic location of client device 102. Again, the policies may correspond to any combination of these factors, and may include additional or alternative factors in other examples.")
1[d]	checking, by the network node, if the packet satisfies the criterion;	Chua '151 discloses checking, by the network node, if the packet satisfies the criterion. For example, Chua '151 discloses a SDN controller 112 programming network devices of SDN 106 to perform programmed actions on packets of a packet flow to enforce policies by the controller based on a per-packet flow basis including analyzing packets having particular header data. Chua '151 at 4:44-55 ("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 receive credentials from a client device in accordance with a public key infrastructure (PKI)-based authentication protocol, determine one or more policies that are applicable to the client device based on the received credentials, and program network devices of the SDN to enforce the determined policies on a per-packet-flow basis for packet flows including the client device.") Chua '151 at 7:28-54 ("SDN controller 112 may receive data as input from service devices 116. For example, SDN controller 112 may be configured 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 programmed actions on packets of a packet flow based on this data. Such programmed actions may include: Allow—explicitly allow a certain network flow to proceed to its destination Block—explicitly block a certain flow from traversing SDN 106 Mirror—allow the traffic, but send a copy of the traffic for deeper inspection or recording

Cisco Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1291 of 2137

No.	'111 Patent Claim 1	Chua '151
		to, e.g., one of service devices 116 Redirect—redirect the traffic to another network device (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. Transform—modify or translate values of headers of packets in the network flow Encapsulate—encapsulate packets in the network flow with a particular header") Chua '151 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.") Chua '151 at 26:15-22 ("Determining the paths generally includes determining, from a first network device of the SDN, Subsequent "next hop' network devices through the SDN. Thus, SDN controller 112 may analyze connections between the network devices (e.g., Switches and service devices) and direct traffic from one device to another, assuming that the devices are communicatively coupled and cause the traffic to travel substantially in the direction of the destination of the traffic.") Chua '151 at 26:23-43 ("SDN controller 112 may then program the network devices to forward traffic along the determined paths (310). In con-ventional routers, control plane elements of a router program a forwarding information base (FIB) of a data plane of the

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1292 of 2137

No.	'111 Patent Claim 1	Chua '151
		router. However, SDN controller 112 represents a control plane element that can program data plane elements (e.g., switches) remotely, that is, without being part of or even directly coupled to the data plane elements. For example, SDN controller 112 may program any or all of network devices 108,110 and service devices 116 to forward network traffic having particular header data (e.g., source IP address, destination IP address, source port, destination port, protocol, MPLS label(s), VLAN tags, or the like) to a particular next hop (that is, next network device). Moreover, in accordance with the techniques of this disclosure, SDN controller 112 may program network devices of an SDN to forward traffic according to network paths that were determined based on one or more of connections determined between the network devices, service devices among the network devices, zones for packet flows through the network devices, and/or trusted or untrusted packet flows.") Chua '151 at 26:23-43 ("SDN controller 112 may then program the network devices to forward traffic along the determined paths (310). In con-ventional routers, control plane elements of a router program a forwarding information base (FIB) of a data plane of the router. However, SDN controller 112 represents a control plane element that can program data plane elements (e.g., switches) remotely, that is, without being part of or even directly coupled to the data plane elements. For example, SDN controller 112 may program any or all of network devices 108,110 and service devices 116 to forward network traffic having particular header data (e.g., source IP address, destination IP address, source port, destination port, protocol, MPLS label(s), VLAN tags, or the like) to a particular next hop (that is, next network device). Moreover, in accordance with the techniques of this disclosure, SDN controller 112 may program network devices of an SDN to forward traffic according to network paths that were determined based on one or more of connections determined between the network devices, service devices among the network devices, zones for packet flows through the network devices, and/or trusted or untrusted packet flows.") Chua '151 at 27:56-28:6 ("In addition, SDN controller 112 may program network devices to receive packets from the service device (332). More particularly, programming the network devices may include programming network devices to receive packets from a service device and forward the packets along the path that was determined for the packet flow corresponding to the packets. In some instances, the network device that forwards a packet to a service device may be the same network device that receives the packet from the

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1293 of 2137

No.	’111 Patent Claim 1	Chua ’151
		service device. In such instances, SDN controller 112 may program a forwarding information base (FIB) of the network device such that the network device analyzes not only a packet header of the packet to determine a next hop for the packet, but also an input interface (e.g., an input port) by which the packet is received. That is, the FIB may include a forwarding table that includes data defining a correspondence from packet header information and an input interface to an output interface, for a packet flow.”) Chua ’151 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.”) Chua ’151 at 31:30-47 (“SDN controller 112 may then program network devices of the SDN to enforce the policies (400). For example, SDN controller 112 may program the network devices to direct packet flows including client device 102 along various paths, depending on the applicable policies for the packet flows. That is, certain packet flows originating from client device 102 or destined for client device 102 may be forwarded along different paths than other packet flows originating from or destined for client device 102, based on, e.g., the user of client device 102, an application, corresponding to the packet flow, that is being executed by client device 102, and/or a geographic location of client

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1294 of 2137

No.	'111 Patent Claim 1	Chua '151
		device 102. Likewise, packet flows for the same user using client device 102 and the same appli-cation may be directed along different paths through SDN 106 based on the geographic location of client device 102. Again, the policies may correspond to any combination of these factors, and may include additional or alternative fac-tors in other examples.")
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	Chua '151 discloses responsive to the packet not satisfying the criterion, sending, by the network node over the packet network, the packet to the second entity. For example, Chua '151 discloses network devices sending packets to a next hop depending on the particular header data, including header data that does not match the programmed policy. Chua '151 at 7:28-54 ("SDN controller 112 may receive data as input from service devices 116. For example, SDN controller 112 may be configured 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 programmed actions on packets of a packet flow based on this data. Such programmed actions may include: Allow—explicitly allow a certain network flow to proceed to its destination Block—explicitly block a certain flow from traversing SDN 106 Mirror—allow the traffic, but send a copy of the traffic for deeper inspection or recording to, e.g., one of service devices 116 Redirect—redirect the traffic to another network device (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.
IPR2023-00401, Page 1295 of 2137

No.	’111 Patent Claim 1	Chua ’151
		Transform—modify or translate values of headers of packets in the network flow Encapsulate—encapsulate packets in the network flow with a particular header”) Chua ’151 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.”) Chua ‘151 at 26:15-22 (“Determining the paths generally includes determining, from a first network device of the SDN, Subsequent “next hop' network devices through the SDN. Thus, SDN controller 112 may analyze connections between the network devices (e.g., Switches and service devices) and direct traffic from one device to another, assuming that the devices are communicatively coupled and cause the traffic to travel substantially in the direction of the destination of the traffic.”) Chua ’151 at 26:23-43 (“SDN controller 112 may then program the network devices to forward traffic along the determined paths (310). In con-ventional routers, control plane elements of a router program a forwarding information base (FIB) of a data plane of the router. However, SDN controller 112 represents a control plane element that can program data plane elements (e.g., switches) remotely, that is, without being part of or even directly coupled to the data plane elements. For example, SDN controller 112 may program any or all of network devices 108,110 and service devices 116 to forward network traffic having

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1296 of 2137

No.	'111 Patent Claim 1	Chua '151
		particular header data (e.g., source IP address, destination IP address, source port, destination port, protocol, MPLS label(s), VLAN tags, or the like) to a particular next hop (that is, next network device). Moreover, in accordance with the techniques of this disclosure, SDN controller 112 may program network devices of an SDN to forward traffic according to network paths that were determined based on one or more of connections determined between the network devices, service devices among the network devices, zones for packet flows through the network devices, and/or trusted or untrusted packet flows.") Chua '151 at 27:56-28:6 ("In addition, SDN controller 112 may program network devices to receive packets from the service device (332). More particularly, programming the network devices may include programming network devices to receive packets from a service device and forward the packets along the path that was determined for the packet flow corresponding to the packets. In some instances, the network device that forwards a packet to a service device may be the same network device that receives the packet from the service device. In such instances, SDN controller 112 may program a forwarding information base (FIB) of the network device such that the network device analyzes not only a packet header of the packet to determine a next hop for the packet, but also an input interface (e.g., an input port) by which the packet is received. That is, the FIB may include a forwarding table that includes data defining a correspondence from packet header information and an input interface to an output interface, for a packet flow.") Chua '151 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,

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1297 of 2137

No.	'111 Patent Claim 1	Chua '151
		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.") Chua '151 at 31:30-47 ("SDN controller 112 may then program network devices of the SDN to enforce the policies (400). For example, SDN controller 112 may program the network devices to direct packet flows including client device 102 along various paths, depending on the applicable policies for the packet flows. That is, certain packet flows originating from client device 102 or destined for client device 102 may be forwarded along different paths than other packet flows originating from or destined for client device 102, based on, e.g., the user of client device 102, an application, corresponding to the packet flow, that is being executed by client device 102, and/or a geographic location of client device 102. Likewise, packet flows for the same user using client device 102 and the same appli-cation may be directed along different paths through SDN 106 based on the geographic location of client device 102. Again, the policies may correspond to any combination of these factors, and may include additional or alternative fac-tors in other examples.")
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.	Chua '151 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. For example, Chua '151 discloses network devices sending packets to the controller or another device based on network devices performing the action programmed by the controller, by analyzing the packet header. Chua '151 at 2:49-61 ("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 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1298 of 2137

No.	'111 Patent Claim 1	Chua '151
		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.") Chua '151 at 7:28-54 ("SDN controller 112 may receive data as input from service devices 116. For example, SDN controller 112 may be configured 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 programmed actions on packets of a packet flow based on this data. Such programmed actions may include: Allow—explicitly allow a certain network flow to proceed to its destination Block—explicitly block a certain flow from traversing SDN 106 Mirror—allow the traffic, but send a copy of the traffic for deeper inspection or recording to, e.g., one of service devices 116 Redirect—redirect the traffic to another network device (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. Transform—modify or translate values of headers of packets in the network flow Encapsulate—encapsulate packets in the network flow with a particular header") Chua '151 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1299 of 2137

No.	'111 Patent Claim 1	Chua '151
		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.") Chua '151 at 26:23-43 ("SDN controller 112 may then program the network devices to forward traffic along the determined paths (310). In con-ventional routers, control plane elements of a router program a forwarding information base (FIB) of a data plane of the router. However, SDN controller 112 represents a control plane element that can program data plane elements (e.g., switches) remotely, that is, without being part of or even directly coupled to the data plane elements. For example, SDN controller 112 may program any or all of network devices 108,110 and service devices 116 to forward network traffic having particular header data (e.g., source IP address, destination IP address, source port, destination port, protocol, MPLS label(s), VLAN tags, or the like) to a particular next hop (that is, next network device). Moreover, in accordance with the techniques of this disclosure, SDN controller 112 may program network devices of an SDN to forward traffic according to network paths that were determined based on one or more of connections determined between the network devices, service devices among the network devices, zones for packet flows through the network devices, and/or trusted or untrusted packet flows.") Chua '151 at 27:56-28:6 ("In addition, SDN controller 112 may program network devices to receive packets from the service device (332). More particularly, programming the network devices may include programming network devices to receive packets from a service device and forward the packets along the path that was determined for the packet flow corresponding to the packets. In some instances, the network device that forwards a packet to a service device may be the same network device that receives the packet from the service device. In such instances, SDN controller 112 may program a forwarding information base (FIB) of the network device such that the network device analyzes not only a packet header of the packet to determine a next hop for the packet, but also an input interface (e.g., an input port) by which the packet is received. That is, the FIB may include

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1300 of 2137

No.	'111 Patent Claim 1	Chua '151
		a forwarding table that includes data defining a correspondence from packet header information and an input interface to an output interface, for a packet flow.") Chua '151 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.")

No.	'111 Patent Claim 2	Chua '151
2[a]	The method according to claim 1, wherein the instruction is 'probe', 'mirror', or 'terminate' instruction, and	Chua '151 discloses the method according to claim 1, wherein the instruction is 'probe', 'mirror', or 'terminate' instruction. For example, Chua '151 discloses the programmed actions include the following actions such as Allow to explicitly allow a certain network flow to proceed to its destination, Block to explicitly block a certain flow from traversing SDN 106, and Mirror to allow the traffic, but send a copy of the traffic for deeper inspection or recording to, e.g., one of service devices 116.

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1301 of 2137

No.	'111 Patent Claim 2	Chua '151
		Chua '151 at 7:28-54 ("SDN controller 112 may receive data as input from service devices 116. For example, SDN controller 112 may be configured 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 programmed actions on packets of a packet flow based on this data. Such programmed actions may include: Allow—explicitly allow a certain network flow to proceed to its destination Block—explicitly block a certain flow from traversing SDN 106 Mirror—allow the traffic, but send a copy of the traffic for deeper inspection or recording to, e.g., one of service devices 116 Redirect—redirect the traffic to another network device (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. Transform—modify or translate values of headers of packets in the network flow Encapsulate—encapsulate packets in the network flow with a particular header") Chua '151 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1302 of 2137

No.	'111 Patent Claim 2	Chua '151
		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.")
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.	Chua '151 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. For example, Chua '151 discloses a Block programmed action to explicitly block a certain flow from traversing SDN 106. Chua '151 at 7:28-54 ("SDN controller 112 may receive data as input from service devices 116. For example, SDN controller 112 may be configured 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 programmed actions on packets of a packet flow based on this data. Such programmed actions may include: Allow—explicitly allow a certain network flow to proceed to its destination Block—explicitly block a certain flow from traversing SDN 106 Mirror—allow the traffic, but send a copy of the traffic for deeper inspection or recording to, e.g., one of service devices 116 Redirect—redirect the traffic to another network device (such as a honeypot device or other device of service devices 116) for either inspection or to keep a potential hacker 'busy' to

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1303 of 2137

No.	'111 Patent Claim 2	Chua '151
		determine if there is a real security threat. Transform—modify or translate values of headers of packets in the network flow Encapsulate—encapsulate packets in the network flow with a particular header") Chua '151 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.") Chua '151 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1304 of 2137

No.	'111 Patent Claim 2	Chua '151
		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.") Chua '151 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.")

No.	'111 Patent Claim 3	Chua '151
3[a]	The method according to claim 1, wherein the instruction is a 'probe', a 'mirror', or a 'terminate' instruction, and	Chua '151 discloses the method according to claim 1, wherein the instruction is a 'probe', a 'mirror', or a 'terminate' instruction. *See supra* at 2(a).
3[b]	upon receiving by the network node the 'mirror' instruction	Chua '151 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.

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1305 of 2137

No.	'111 Patent Claim 3	Chua '151
	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.	For example, Chua '151 discloses a Mirror programmed action to allow the traffic, but send a copy of the traffic for deeper inspection or recording to, e.g., one of service devices 116. Chua '151 at 7:28-54 ("SDN controller 112 may receive data as input from service devices 116. For example, SDN controller 112 may be configured 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 programmed actions on packets of a packet flow based on this data. Such programmed actions may include: Allow—explicitly allow a certain network flow to proceed to its destination Block—explicitly block a certain flow from traversing SDN 106 Mirror—allow the traffic, but send a copy of the traffic for deeper inspection or recording to, e.g., one of service devices 116 Redirect—redirect the traffic to another network device (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. Transform—modify or translate values of headers of packets in the network flow Encapsulate—encapsulate packets in the network flow with a particular header") Chua '151 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1306 of 2137

No.	'111 Patent Claim 3	Chua '151
		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.") Chua '151 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.") Chua '151 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1307 of 2137

No.	'111 Patent Claim 3	Chua '151
		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.")

No.	'111 Patent Claim 4	Chua '151
4[a]	The method according to claim 1, wherein the instruction is 'probe', 'mirror', or 'terminate' instruction, and	Chua '151 discloses the method according to claim 1, wherein the instruction is 'probe', 'mirror', or 'terminate' instruction. *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;	Chua '151 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. For example, Chua '151 discloses the Redirect programmed action to redirect the traffic to another network device for either inspection or to keep a potential hacker 'busy' to determine if there is a real security threat. Chua '151 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:

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1308 of 2137

No.	'111 Patent Claim 4	Chua '151
		Allow-explicitly allow a certain network flow to proceed to its destination Block-explicitly block a certain flow from traversing SDN 106 Mirror-allow the traffic, but send a copy of the traffic for deeper inspection or recording to, e.g., one of service devices 116 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. Transform-modify or translate values of headers of packets in the network flow Encapsulate-encapsulate packets in the network flow with a particular header") Chua '151 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.") Chua '151 at 24:48-63 ("Initially, SDN controller 112 determines connections between network devices of an SDN (300). For example, network devices of the SDN (such as network devices 108, 110, which may correspond to network switches) may be configured to send acknowledgements to SDN controller 112 in response to receiving packets from SDN controller 112, as well as to forward the packets to other network devices to which the network devices are communicatively coupled. By analyzing the responses from the network devices, SDN con-troller 112 may determine connections between network devices of an SDN. For example, all ports on one network device 110 might be programmed to send

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1309 of 2137

No.	'111 Patent Claim 4	Chua '151
		all incoming pack-ets of a certain type to the SDN Controller 112. By sending an appropriately crafted packet (of the same type) on all ports of 108, the SDN Controller can determine how Network Device 108 is connected to Network Device 110.") Chua '151 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.") Chua '151 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.")

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1310 of 2137

No.	'111 Patent Claim 4	Chua '151
4[c]	responsive to receiving the packet, analyzing the packet, by the controller;	Chua '151 discloses responsive to receiving the packet, analyzing the packet, by the controller. For example, Chua '151 discloses the SDN controller 112 further performs deep packet inspection (DPI) on packets from client device 102 (402). Chua '151 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.") Chua '151 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.") Chua '151 at 24:48-63 ("Initially, SDN controller 112 determines connections between network devices of an SDN (300). For example, network devices of the SDN (such as

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1311 of 2137

No.	'111 Patent Claim 4	Chua '151
		network devices 108, 110, which may correspond to network switches) may be configured to send acknowledgements to SDN controller 112 in response to receiving packets from SDN controller 112, as well as to forward the packets to other network devices to which the network devices are communicatively coupled. By analyzing the responses from the network devices, SDN con-troller 112 may determine connections between network devices of an SDN. For example, all ports on one network device 110 might be programmed to send all incoming pack-ets of a certain type to the SDN Controller 112. By sending an appropriately crafted packet (of the same type) on all ports of 108, the SDN Controller can determine how Network Device 108 is connected to Network Device 110.") Chua '151 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.") Chua '151 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1312 of 2137

No.	'111 Patent Claim 4	Chua '151
		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.")
4[d]	sending the packet, by the controller, to the network node; and	Chua '151 discloses sending the packet, by the controller, to the network node. For example, Chua '151 discloses the SDN controller 112 program network devices to receive packets from the service device (332). A person of ordinary skill in the art would understand that a packet sent to the controller for analysis is then sent back to the network node. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Chua '151 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. Chua '151 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.")

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1313 of 2137

No.	'111 Patent Claim 4	Chua '151
		Chua '151 at 24:48-63 ("Initially, SDN controller 112 determines connections between network devices of an SDN (300). For example, network devices of the SDN (such as network devices 108, 110, which may correspond to network switches) may be configured to send acknowledgements to SDN controller 112 in response to receiving packets from SDN controller 112, as well as to forward the packets to other network devices to which the network devices are communicatively coupled. By analyzing the responses from the network devices, SDN con-troller 112 may determine connections between network devices of an SDN. For example, all ports on one network device 110 might be programmed to send all incoming pack-ets of a certain type to the SDN Controller 112. By sending an appropriately crafted packet (of the same type) on all ports of 108, the SDN Controller can determine how Network Device 108 is connected to Network Device 110.") Chua '151 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.") Chua '151 at 26:23-43 ("In addition, SDN controller 112 may program network devices to receive packets from the service device (332). More particularly, programming the network devices may include programming network devices to receive packets from a service device and forward the packets along the path that was determined for the packet flow corresponding to the packets. In some instances, the network device that forwards a packet to a service device may be the same network device that receives the packet from the service device. In such instances, SDN controller 112 may program a forwarding information base (FIB) of the network device such that the network device analyzes not only a packet header of the packet to determine a next hop for the packet, but also an input interface (e.g., an input port) by which the packet is received. That is, the FIB may include a forwarding table that includes data defining a correspondence from packet header

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1314 of 2137

No.	'111 Patent Claim 4	Chua '151
		information and an input interface to an output interface, for a packet flow.") Chua '151 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.") Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Chua '151 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 are examples of two such references. 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 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.
IPR2023-00401, Page 1315 of 2137

No.	’111 Patent Claim 4	Chua ’151
		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.”) 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.”) 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

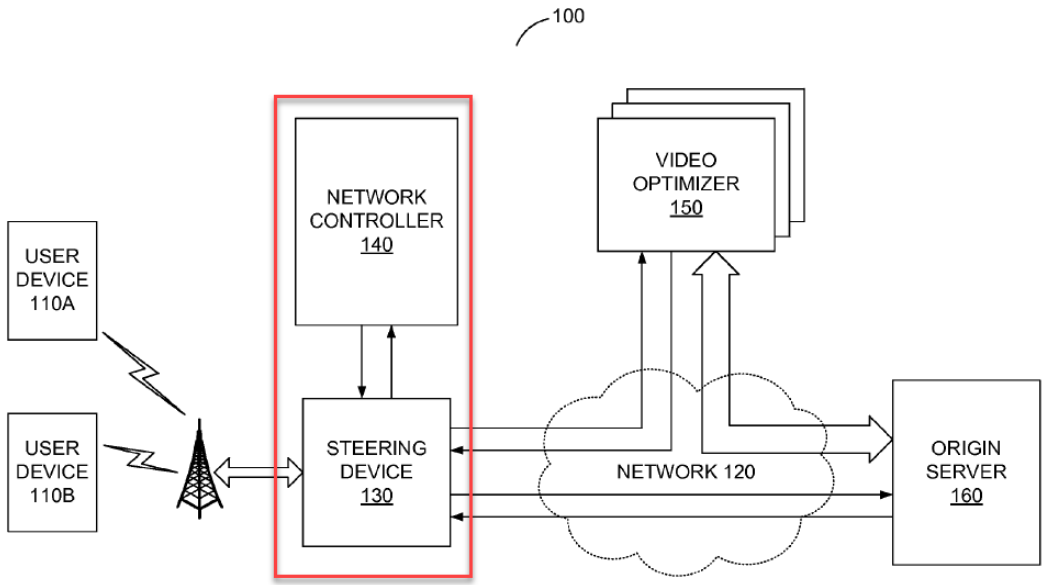
Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1316 of 2137

No.	'111 Patent Claim 4	Chua '151
		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.") 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.") 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1317 of 2137

No.	’111 Patent Claim 4	Chua ’151
		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.”) 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.”) Swenson at Figure 1 (annotation added)

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1318 of 2137

No.	'111 Patent Claim 4	Chua '151
		 FIG. 1
4[e]	responsive to receiving the packet, sending the packet, by the network node, to the second entity.	Chua '151 discloses responsive to receiving the packet, sending the packet, by the network node, to the second entity. For example, Chua '151 discloses the SDN controller 112 programming 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. A person having ordinary skill in the art would understand that the packet would then be sent to its originally intended destination, 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 Chua '151 is found to not meet this limitation, responsive to receiving the

No.	'111 Patent Claim 4	Chua '151
		packet, sending the packet, by the network node, to the second entity would have been obvious to a person having ordinary skill in the art, as explained below. Chua '151 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.") Chua '151 at 24:48-63 ("Initially, SDN controller 112 determines connections between network devices of an SDN (300). For example, network devices of the SDN (such as network devices 108, 110, which may correspond to network switches) may be configured to send acknowledgements to SDN controller 112 in response to receiving packets from SDN controller 112, as well as to forward the packets to other network devices to which the network devices are communicatively coupled. By analyzing the responses from the network devices, SDN con-troller 112 may determine connections between network devices of an SDN. For example, all ports on one network device 110 might be programmed to send all incoming pack-ets of a certain type to the SDN Controller 112. By sending an appropriately crafted packet (of the same type) on all ports of 108, the SDN Controller can determine how Network Device 108 is connected to Network Device 110.") Chua '151 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1320 of 2137

No.	'111 Patent Claim 4	Chua '151
		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.") Chua '151 at 26:23-43 ("In addition, SDN controller 112 may program network devices to receive packets from the service device (332). More particularly, programming the network devices may include programming network devices to receive packets from a service device and forward the packets along the path that was determined for the packet flow corresponding to the packets. In some instances, the network device that forwards a packet to a service device may be the same network device that receives the packet from the service device. In such instances, SDN controller 112 may program a forwarding information base (FIB) of the network device such that the network device analyzes not only a packet header of the packet to determine a next hop for the packet, but also an input interface (e.g., an input port) by which the packet is received. That is, the FIB may include a forwarding table that includes data defining a correspondence from packet header information and an input interface to an output interface, for a packet flow.") Chua '151 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.") Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, the Chua '151 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)

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1321 of 2137

No.	'111 Patent Claim 4	Chua '151
		of the references identified in element 4(e) of Exhibit E-4 renders the claim, including the present limitation, obvious. Below is an example. 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, for example, the user device or origin server. 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.") 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

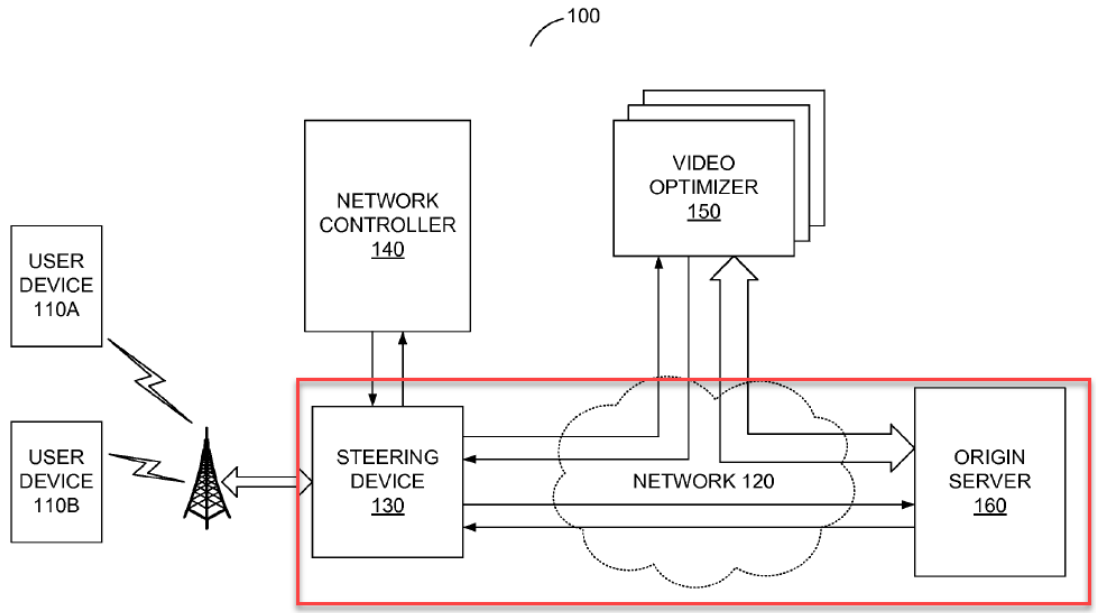
Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1322 of 2137

No.	'111 Patent Claim 4	Chua '151
		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.") 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.") 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.") 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.
IPR2023-00401, Page 1323 of 2137

No.	’111 Patent Claim 4	Chua ’151
		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.”) 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.”) 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1324 of 2137

No.	'111 Patent Claim 4	Chua '151
		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.") Swenson at Figure 1 (annotation added)  FIG. 1

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1325 of 2137

No.	'111 Patent Claim 5	Chua '151
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.	Chua '151 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. For example, Chua '151 discloses in response to a program enforced by the network device on particular packet header data, sending the packet or data representative of the packet to the controller. *See supra* at Claim 1. Chua '151 at 2:38-47 ("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.") Chua '151 at 2:49-61 ("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 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.") Chua '151 at 2:62-3:6 ("In another example, a computer-readable storage medium has stored thereon instructions that, when executed, cause a processor of a controller device for a software defined network (SDN) to program a set of network devices of the SDN to form a path through the SDN and to send data representa-tive of packets sent along the path to the controller device, send packets of a packet flow corresponding to the path to one of the set

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1326 of 2137

No.	'111 Patent Claim 5	Chua '151
		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.") Chua '151 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: Allow-explicitly allow a certain network flow to proceed to its destination Block-explicitly block a certain flow from traversing SDN 106 Mirror-allow the traffic, but send a copy of the traffic for deeper inspection or recording to, e.g., one of service devices 116 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. Transform-modify or translate values of headers of packets in the network flow Encapsulate-encapsulate packets in the network flow with a particular header") Chua '151 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1327 of 2137

No.	'111 Patent Claim 5	Chua '151
		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.") Chua '151 at 29:28-39 ("In this manner, the method of FIG. 7 represents an example of a method including 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 presenting a report representative of the determination.")

No.	'111 Patent Claim 6	Chua '151
6	The method according to claim 5, further comprising storing the received packet or a portion thereof, by the controller, in a memory.	Chua '151 discloses the method according to claim 5, further comprising storing the received packet or a portion thereof, by the controller, in a memory. For example, Chua '151 discloses logging and storing the packets, representative data of the packets, paths, and programs. *See supra* at Claim 5. Chua '151 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:

~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1328 of 2137

No.	'111 Patent Claim 6	Chua '151
		Allow-explicitly allow a certain network flow to proceed to its destination Block-explicitly block a certain flow from traversing SDN 106 Mirror-allow the traffic, but send a copy of the traffic for deeper inspection or recording to, e.g., one of service devices 116 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. Transform-modify or translate values of headers of packets in the network flow Encapsulate-encapsulate packets in the network flow with a particular header") Chua '151 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.") Chua '151 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.")

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1329 of 2137

No.	'111 Patent Claim 6	Chua '151
		Chua '151 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. 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.")

No.	'111 Patent Claim 7	Chua '151
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.	Chua '151 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. For example, Chua '151 discloses in response to a program enforced by the network device on particular packet header data, sending data representative of the packet to the controller. *See supra* at Claim 5. Chua '151 at 2:38-47 ("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,

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1330 of 2137

No.	'111 Patent Claim 7	Chua '151
		along the path based on data received from the set of network devices, and present-ing a report representative of the determination.") Chua '151 at 2:49-61 ("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 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.") Chua '151 at 2:62-3:6 ("In another example, a computer-readable storage medium has stored thereon instructions that, when executed, cause a processor of a controller device for a software defined network (SDN) to program a set of network devices of the SDN to form a path through the SDN and to send data representa-tive of packets sent along the path to the controller device, send 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.") Chua '151 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: Allow-explicitly allow a certain network flow to proceed to its destination Block-explicitly block a certain flow from traversing SDN 106

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1331 of 2137

No.	'111 Patent Claim 7	Chua '151
		Mirror-allow the traffic, but send a copy of the traffic for deeper inspection or recording to, e.g., one of service devices 116 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. Transform-modify or translate values of headers of packets in the network flow Encapsulate-encapsulate packets in the network flow with a particular header") Chua '151 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.") Chua '151 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1332 of 2137

No.	'111 Patent Claim 7	Chua '151
		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.") Chua '151 at 29:28-39 ("In this manner, the method of FIG. 7 represents an example of a method including 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 presenting a report representative of the determination.")

No.	'111 Patent Claim 8	Chua '151
8[a]	The method according to claim 7, wherein the portion of the packet consists of multiple consecutive bytes, and	Chua '151 discloses the method according to claim 7, wherein the portion of the packet consists of multiple consecutive bytes. For example, Chua '151 discloses representative data of packets of various byte sizes and counts. A person of ordinary skill in the art would understand that packets consist of multiple consecutive bytes comprising a portion of a packet. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Chua at '151 is found to not meet this limitation, wherein the portion of the packet consists of multiple consecutive byte would have been obvious to a person having ordinary skill in the art, as explained below. *See supra* at Claim 7. Chua '151 at 13:42-53 ("Control unit 130 may also maintain data representative of a history of statistics for elements of SDN 106, such as switches, switch ports, flows, and flow groups. Statistics 146 of FIG. 2 represents this data. The statistics entry for each instance in

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1333 of 2137

No.	'111 Patent Claim 8	Chua '151
		time may be dynamically extensible to allow additional counters and other values to be added as the system is extended. Statistics 146 may include a set of data entries, where each entry includes date and time of the last update, packet count, byte count, received packets count, received byte count, transmitted packet count, transmitted byte count, receive errors, transmit errors, received frame errors, transmit frame errors, and collisions.") Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Chua '151 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. For example, Kempf discloses consecutive bytes of a packet header field. 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.") 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.
IPR2023-00401, Page 1334 of 2137

No.	'111 Patent Claim 8	Chua '151
		cloud computing system or to the billing entity handling GTP' and not the S-GW-D and P-GW-D data plane switches.") 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.") 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.
IPR2023-00401, Page 1335 of 2137

No.	'111 Patent Claim 8	Chua '151
		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.") 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.") 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: Write-Metadata (GTP-TEID, 0x FFFFFFFF) Apply-Actions (Set-Output-Port LOCAL_GTP _U_ENCAP)")

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1336 of 2137

No.	'111 Patent Claim 8	Chua '151
		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. 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.	Chua '151 discloses wherein the instruction comprises identification of the consecutive bytes in the packet. For example, Chua '151 discloses programming network devices by the controller based on information gathered from the byte size and counts of representative data of packets. A person of ordinary skill in the art would understand that the consecutive bytes of the packet are identified according to an instruction. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Chua '151 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. Chua '151 at 13:12-41 ("A Source object may define the parameters needed to specify the network traffic source. Each Source object may include data representative of an identifier (e.g., a text string name) for the source, and a set of one or more zones. In particular, one or more flows may be grouped into zones. The zones may represent administrative or security areas that ensure that network managers or customers cannot interfere with each other from a switch control or data plane perspec-tive. For example, a test or research SDN environment may operate entirely within a single zone, and a simple production deployment may have a secure administrative zone used for management purposes and an open operational zone used for dynamic network programming. Multi-tenant or multi-appli-cation deployments may utilize unique zones for each cus-tomer or function or application. There is no fundamental restriction to how zones can be partitioned. In some

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1337 of 2137

No.	'111 Patent Claim 8	Chua '151
		examples, zones are partitioned on a per IP address, per port, per VLAN, or possibly per subnet basis, but this concept can be usefully extended on a per protocol basis or to any other packet header field or per application or per user. SDN con-troller 112 may ensure that a flow rule written for one zone incorporates sufficient additional match rules to prevent flows from other zones from being processed inappropriately. Zones may be ordered by numerical priority to avoid confu-sion that may otherwise occur when overlapping partitions are created. Each zone may support OpenFlow, normal dis-tributed, hybrid, or even alternate protocol types indepen-dently. Zones can also be used to partition off areas where no traffic is expected, allowing illegitimate traffic to be black holed, spanned, or redirected elsewhere for further action.") Chua '151 at 13:42-53 ("Control unit 130 may also maintain data representative of a history of statistics for elements of SDN 106, such as switches, switch ports, flows, and flow groups. Statistics 146 of FIG. 2 represents this data. The statistics entry for each instance in time may be dynamically extensible to allow additional counters and other values to be added as the system is extended. Statistics 146 may include a set of data entries, where each entry includes date and time of the last update, packet count, byte count, received packets count, received byte count, transmitted packet count, transmitted byte count, receive errors, transmit errors, received frame errors, transmit frame errors, and collisions.") Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Chua '151 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. For example, Kempf discloses rules in which the flow table includes matching to the consecutive bytes of a packet header. 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.
IPR2023-00401, Page 1338 of 2137

No.	’111 Patent Claim 8	Chua ’151
		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.”) 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.”) 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1339 of 2137

No.	'111 Patent Claim 8	Chua '151
		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.") 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.") 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.") 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.
IPR2023-00401, Page 1340 of 2137

No.	'111 Patent Claim 8	Chua '151
		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.") 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: Write-Metadata (GTP-TEID, 0x FFFFFFFF) Apply-Actions (Set-Output-Port LOCAL_GTP _U_ENCAP)") For example, Copeland discloses identifying the sampled packet headers comprised of fragmented packets of smaller byte sizes. 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.") 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.") 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.
IPR2023-00401, Page 1341 of 2137

No.	'111 Patent Claim 8	Chua '151
		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	Chua '151
9	The method according to claim 5, further comprising responsive to receiving the packet, analyzing the packet, by the controller.	Chua '151 discloses the method according to claim 5, further comprising responsive to receiving the packet, analyzing the packet, by the controller. For example, Chua '151 discloses the SDN controller 112 further performs deep packet inspection (DPI) on packets from client device 102 (402). *See supra* at Claim 5. Chua '151 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.") Chua '151 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 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1342 of 2137

No.	'111 Patent Claim 9	Chua '151
		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.") Chua '151 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).") Chua '151 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.") Chua '151 at 15:4-17 ("Another example use case for SDN controller 112 includes establishing a perimeter DMZ interconnection. Traffic entering a data center must often be distributed to a variety of appliances or server-based application-aware devices based on potentially deep inspection of incoming (and for certain flow types, outgoing) packets. Rather than route all traffic through L4-L 7 firewalls or application load balancers, or utilize distributed protocols such as policy based routing with limited capability and complex network interactions, SDN controller 112 may allow customers (e.g., administrator 114) to directly sort traffic on a per-packet or per-flow basis for individualized processing, saving cost, improving performance, limiting risk, improving availability, and increasing control.") Chua '151 at 15:49-63 ("In addition or in the alternative to the above, control unit 130 may be configured to utilize external triggers and information to change rules within programmable flow devices, e.g., switches of SDN 106. In particular, path determination

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1343 of 2137

No.	'111 Patent Claim 9	Chua '151
		unit 134 may be configured to utilize security information 148 to develop paths through SDN 106 to provide protection to SDN 106. SDN controller 112 may develop rules, stored in rules 142, that define triggers associated with events that cause control unit 130 to program network devices of SDN 106 via SDN interface 152 in response to the triggers. SDN controller 112 may program the network devices of SDN 106 such that, in response to one of the triggers "firing," that is, becoming activated, the network devices modify forwarding information for a packet flow related to the external trigger that fired.") Chua '151 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. 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.") Chua '151 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.")

~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1344 of 2137

No.	'111 Patent Claim 12	Chua '151
12	The method according to claim 9, wherein the analyzing comprises applying security or data analytic application.	Chua '151 discloses the method according to claim 9, wherein the analyzing comprises applying security or data analytic application. For example, Chua '151 discloses the service devices 116 include, for example, intrusion detection service (IDS) devices, intrusion prevention system (IPS) devices, web proxies, and web-application firewalls. *See supra* at Claim 9. Chua '151 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.") Chua '151 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. 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1345 of 2137

No.	'111 Patent Claim 12	Chua '151
		intended destination. As noted above, zones may be ordered based on priority values when overlap occurs.") Chua '151 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.") Chua '151 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.") Chua '151 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1346 of 2137

No.	’111 Patent Claim 12	Chua ’151
		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.”) Chua ’151 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.”) Chua ’151 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 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.”)

No.	'111 Patent Claim 12	Chua '151
		Chua '151 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).") Chua '151 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.") Chua '151 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.
IPR2023-00401, Page 1348 of 2137

No.	'111 Patent Claim 13	Chua '151
13	The method according to claim 9, wherein the analyzing comprises applying security application that comprises firewall or intrusion detection functionality.	Chua '151 discloses the method according to claim 9, wherein the analyzing comprises applying security application that comprises firewall or intrusion detection functionality. For example, Chua '151 discloses security determinations and analysis that involve the use of firewalls or intrusion detection services. *See supra* at Claim 9. Chua '151 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.") Chua '151 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. 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.")

~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1349 of 2137

No.	'111 Patent Claim 13	Chua '151
		Chua '151 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.") Chua '151 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.") Chua '151 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1350 of 2137

No.	’111 Patent Claim 13	Chua ’151
		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.”) Chua ’151 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.”) Chua ’151 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 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.”) Chua ’151 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1351 of 2137

No.	'111 Patent Claim 13	Chua '151
		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).") Chua '151 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.") Chua '151 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.") Chua '151 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1352 of 2137

No.	'111 Patent Claim 13	Chua '151
		explicitly shown in FIG. 3, such as routers, switches, hubs, gateways, security devices, or the like.")

No.	'111 Patent Claim 14	Chua '151
14	The method according to claim 9, wherein the analyzing comprises performing Deep Packet Inspection (DPI) or using a DPI engine on the packet.	Chua '151 discloses the method according to claim 9, wherein the analyzing comprises performing Deep Packet Inspection (DPI) or using a DPI engine on the packet. For example, Chua '151 discloses the SDN controller 112 further performs deep packet inspection (DPI) on packets. *See supra* at Claim 9. Chua '151 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.") Chua '151 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.") Chua '151 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1353 of 2137

No.	'111 Patent Claim 14	Chua '151
		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.") Chua '151 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.")

No.	'111 Patent Claim 15	Chua '151
15[a]	The method according to claim 9, wherein the packet comprises distinct header and payload fields, and	Chua '151 discloses the method according to claim 9, wherein the packet comprises distinct header and payload fields. For example, Chua '151 discloses the network traffic having particular header data (e.g., source IP address, destination IP address, source port, destination port, protocol, MPLS label(s), VLAN tags, or the like). A person of ordinary skill in the art would understand that packets also have distinct payload fields to be forwarded to the intended destination. *See supra* at Claim 9.

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1354 of 2137

No.	'111 Patent Claim 15	Chua '151
		Chua '151 at 13:12-41 ("A Source object may define the parameters needed to specify the network traffic source. Each Source object may include data representative of an identifier (e.g., a text string name) for the source, and a set of one or more zones. In particular, one or more flows may be grouped into zones. The zones may represent administrative or security areas that ensure that network managers or customers cannot interfere with each other from a switch control or data plane perspec-tive. For example, a test or research SDN environment may operate entirely within a single zone, and a simple production deployment may have a secure administrative zone used for management purposes and an open operational zone used for dynamic network programming. Multi-tenant or multi-appli-cation deployments may utilize unique zones for each cus-tomer or function or application. There is no fundamental restriction to how zones can be partitioned. In some examples, zones are partitioned on a per IP address, per port, per VLAN, or possibly per subnet basis, but this concept can be usefully extended on a per protocol basis or to any other packet header field or per application or per user. SDN con-troller 112 may ensure that a flow rule written for one zone incorporates sufficient additional match rules to prevent flows from other zones from being processed inappropriately. Zones may be ordered by numerical priority to avoid confu-sion that may otherwise occur when overlapping partitions are created. Each zone may support OpenFlow, normal dis-tributed, hybrid, or even alternate protocol types indepen-dently. Zones can also be used to partition off areas where no traffic is expected, allowing illegitimate traffic to be black holed, spanned, or redirected elsewhere for further action.") Chua '151 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 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1355 of 2137

No.	'111 Patent Claim 15	Chua '151
		enterprise network cannot be trusted and, therefore, should be monitored by a security device.") Chua '151 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).") Chua '151 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.") Chua '151 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. 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.")

~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1356 of 2137

No.	'111 Patent Claim 15	Chua '151
		Chua '151 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.") Chua '151 at 26:23-43 ("SDN controller 112 may then program the network devices to forward traffic along the determined paths (310). In conventional routers, control plane elements of a router program a forwarding information base (FIB) of a data plane of the router. However, SDN controller 112 represents a control plane element that can program data plane elements (e.g., switches) remotely, that is, without being part of or even directly coupled to the data plane elements. For example, SDN controller 112 may program any or all of network devices 108, 110 and service devices 116 to forward network traffic having particular header data (e.g., source IP address, destination IP address, source port, destination port, protocol, MPLS label(s), VLAN tags, or the like) to a particular next hop (that is, next network device). Moreover, in accordance with the techniques of this disclosure, SDN controller 112 may program network devices of an SDN to forward traffic according to network paths that were determined based on one or more of connections determined between the network devices, service devices among the network devices, zones for packet flows through the network devices, and/or trusted or untrusted packet flows.")
15[b]	wherein the analyzing comprises checking part of, or whole of, the payload field.	Chua '151 discloses wherein the analyzing comprises checking part of, or whole of, the payload field. For example, Chua '151 discloses analysis of packets in which the controller inspects packets of packet flows. A person of ordinary skill in the art would understand that analysis of a packet, including by deep packet inspect, checks the payload field of a packet to determine whether the content of a packet is malicious.

~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1357 of 2137

No.	'111 Patent Claim 15	Chua '151
		Chua '151 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.")

No.	'111 Patent Claim 16	Chua '151
16[a]	The method according to claim 1, wherein the packet comprises distinct header and payload fields,	Chua '151 discloses the method according to claim 1, wherein the packet comprises distinct header and payload fields. *See supra* at Claim 1, 15[a].
16[b]	the header comprises one or more flag bits, and	Chua '151 discloses the header comprises one or more flag bits. For example, Chua '151 discloses packets with packet header field values represented in various ways. A person of ordinary skill in the art would understand that a header can 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 Chua '151 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. Chua '151 at 12:24-39 ("A FlowChain object may define the configuration data needed to configure and manage a set of flows (that is, packet flows) through the network infrastructure (e.g., through SDN 106). Each FlowChain object may include an identifier (e.g., a name), a Source object, one or more FlowLink objects, a flag to indicate whether the FlowChain is enabled or disabled, a flag to indicate whether the FlowChain is bi-directional, and a set of statistics for the FlowChain. The FlowChain may be defined using a set of configuration commands, e.g., extensible markup language (XML) commands, JavaScript Object Notation (JSON) commands, or other commands. In some examples, the commands

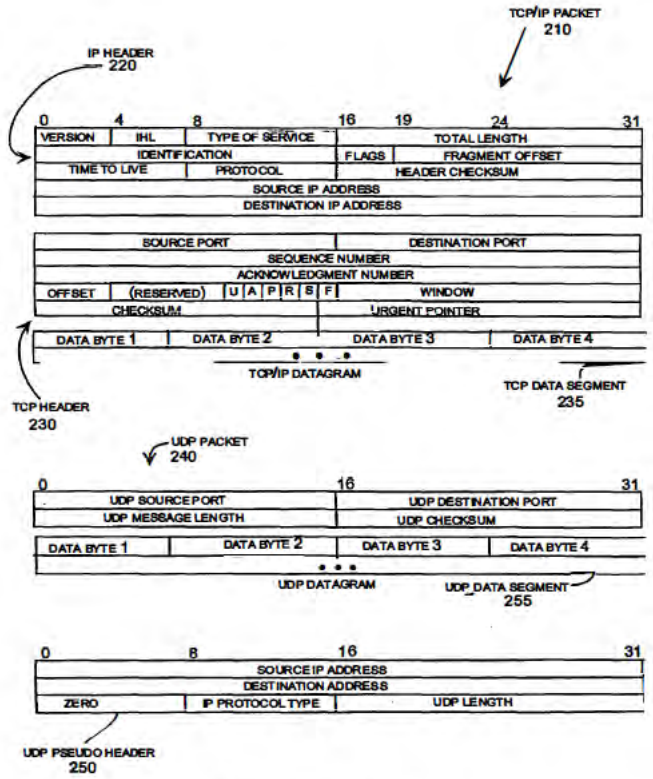
Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1358 of 2137

No.	'111 Patent Claim 16	Chua '151
		may be natural language friendly. That is, the commands may be easily read and interpreted by a human operator. Such commands may benefit users in interacting with a device for creating a FlowChain, as well as other users, e.g., for audits and validation.") Chua '151 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. 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.") Chua '151 at 26:23-43 ("SDN controller 112 may then program the network devices to forward traffic along the determined paths (310). In con-ventional routers, control plane elements of a router program a forwarding information base (FIB) of a data plane of the router. However, SDN controller 112 represents a control plane element that can program data plane elements (e.g., switches) remotely, that is, without being part of or even directly coupled to the data plane elements. For example, SDN controller 112 may program any or all of network devices 108,110 and service devices 116 to forward network traffic having particular header data (e.g., source IP address, destination IP address, source port, destination port, protocol, MPLS label(s), VLAN tags, or the like) to a particular next hop (that is, next network device). Moreover, in accordance with the techniques of this disclosure, SDN controller 112 may program network devices of an SDN to forward traffic according to network paths that were determined based on one or more of connections determined between the network devices, service devices among the network devices, zones for packet flows through the network devices, and/or trusted or untrusted packet flows.")

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1359 of 2137

No.	'111 Patent Claim 16	Chua '151
		Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Chua '151 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. For example, Copeland discloses packet headers with flag bits. Copeland at Figure 2

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1360 of 2137

No.	'111 Patent Claim 16	Chua '151
		 FIG. 2 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.
IPR2023-00401, Page 1361 of 2137

No.	'111 Patent Claim 16	Chua '151
		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.") 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.") 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 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.
IPR2023-00401, Page 1362 of 2137

No.	'111 Patent Claim 16	Chua '151
		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.") 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.") As another example, Kempf discloses packet headers with flag bits. 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.") 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.
IPR2023-00401, Page 1363 of 2137

No.	'111 Patent Claim 16	Chua '151
		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.") 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.") 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.") 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.
IPR2023-00401, Page 1364 of 2137

No.	'111 Patent Claim 16	Chua '151
		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.") 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).") 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.") 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.
IPR2023-00401, Page 1365 of 2137

No.	'111 Patent Claim 16	Chua '151

		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.")
		
		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:
		
		`struct ermst_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.")

16[c]	wherein the packet-applicable criterion is that one or more of the flag bits is set.	Chua '151 discloses wherein the packet-applicable criterion is that one or more of the flag bits is set. For example, Chua '151 discloses specific packet header data used by the network devices to enforce programs, included flag bits of the packet header. A person of ordinary skill in the art would understand that whether flag bits are set of a packet header may constitute the packet header data used by network devices to determine enforcement of a program or

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1366 of 2137

No.	'111 Patent Claim 16	Chua '151
		perform an action. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Chua '151 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. Chua '151 at 26:23-43 ("SDN controller 112 may then program the network devices to forward traffic along the determined paths (310). In con-ventional routers, control plane elements of a router program a forwarding information base (FIB) of a data plane of the router. However, SDN controller 112 represents a control plane element that can program data plane elements (e.g., switches) remotely, that is, without being part of or even directly coupled to the data plane elements. For example, SDN controller 112 may program any or all of network devices 108,110 and service devices 116 to forward network traffic having particular header data (e.g., source IP address, destination IP address, source port, destination port, protocol, MPLS label(s), VLAN tags, or the like) to a particular next hop (that is, next network device). Moreover, in accordance with the techniques of this disclosure, SDN controller 112 may program network devices of an SDN to forward traffic according to network paths that were determined based on one or more of connections determined between the network devices, service devices among the network devices, zones for packet flows through the network devices, and/or trusted or untrusted packet flows.") Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Chua '151 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. For example, Copeland discloses packet specific characteristics including flag bits that are set. 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.
IPR2023-00401, Page 1367 of 2137

No.	'111 Patent Claim 16	Chua '151
		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 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.") 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.") 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.") 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.
IPR2023-00401, Page 1368 of 2137

No.	'111 Patent Claim 16	Chua '151
		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.") As another example, Kempf flow table matches in which the flag bits is set. 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.") 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.
IPR2023-00401, Page 1369 of 2137

No.	'111 Patent Claim 16	Chua '151
		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.") 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.") 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.") 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.
IPR2023-00401, Page 1370 of 2137

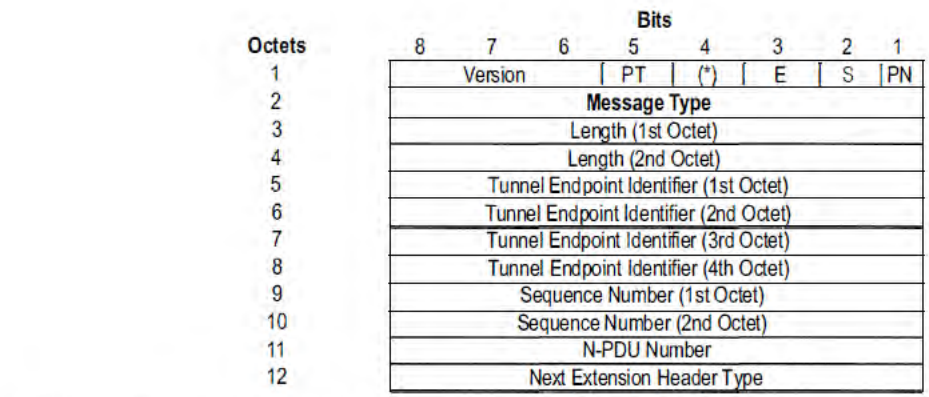
No.	'111 Patent Claim 16	Chua '151
		while the lower two bytes match 0x30, i.e. the packet is a GTP packet not a GTP' packet and the version number is 1.") Kempf at [0094] ("In one embodiment, the system implements han-dling of GTP-U control packets. The OpenFlow controller programs the gateway switch 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).") 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.") 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.
IPR2023-00401, Page 1371 of 2137

No.	'111 Patent Claim 16	Chua '151

		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:
		
		struct ermst_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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1372 of 2137

No.	'111 Patent Claim 16	Chua '151
		

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	Chua '151
17[a]	The method according to claim 16, wherein the packet is an Transmission Control Protocol (TCP) packet, and	Chua '151 discloses the method according to claim 16, wherein the packet is an Transmission Control Protocol (TCP) packet. For example, Chua '151 discloses TCP traffic sessions, i,e., the packet is an Transmission Control Protocol (TCP) packet. *See supra* at Claim 16. Chua '151 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.

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1373 of 2137

No.	'111 Patent Claim 17	Chua '151
		troubleshooting that includes the appropriate IP ranges to watch on, and the TCP/ UDP services that the flowchain is meant to service. 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: 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 Create either UDP packet flows or TCP sessions that confirm to the traffic type inferred or discovered form the flowchain configuration 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) 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 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.") Chua '151 at 24:20-37 ("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 network switches 282 to define multiple paths, e.g., a primary path and one or more backup paths, as discussed above,

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1374 of 2137

No.	'111 Patent Claim 17	Chua '151
		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 servers), 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.")
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.	Chua '151 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. For example, Chua '151 discloses specific packet header data used by the network devices, which may include flag bits. A person of ordinary skill in the art would understand that flag bits may include SYN, ACK, FIN, RST, or any combination of those. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Chua '151 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. Chua '151 at 26:23-43 ("SDN controller 112 may then program the network devices to forward traffic along the determined paths (310). In con-ventional routers, control plane elements of a router program a forwarding information base (FIB) of a data plane of the router. However, SDN controller 112 represents a control plane element that can program data plane elements (e.g., switches) remotely, that is, without being part of or even directly coupled to the data plane elements. For example, SDN controller 112 may program any or all of network devices 108,110 and service devices 116 to forward network traffic having particular header data (e.g., source IP address, destination IP address, source port, destination port, protocol, MPLS label(s), VLAN tags, or the like) to a particular next hop (that is, next network device). Moreover, in accordance with the techniques of this disclosure, SDN controller 112 may program network devices of an SDN to forward traffic according to network paths that were determined based on one or more of connections determined between the network devices, service devices among the network devices, zones for packet flows through the network devices, and/or trusted or untrusted packet flows.")

No.	'111 Patent Claim 17	Chua '151
		Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Chua '151 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 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.") 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

No.	'111 Patent Claim 17	Chua '151
		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.") 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.") 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.") 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.") 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.") 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.") 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.
IPR2023-00401, Page 1377 of 2137

No.	'111 Patent Claim 17	Chua '151
		ACK packet providing to Host2 an acknowledgment number one greater than the sequence number provided in the FIN-ACK packet of Host2.") Uchida 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. As another example, Uchida discloses the TCP (Transmission Control Protocol) FIN flag, RST flag, and SYN flag. 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.") 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.") 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.") 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.")

~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1378 of 2137

No.	'111 Patent Claim 17	Chua '151
		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.") 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.") 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,

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1379 of 2137

No.	'111 Patent Claim 17	Chua '151
		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.") 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).") 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.") 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).") 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1380 of 2137

No.	'111 Patent Claim 17	Chua '151
		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).") 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.") 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.") 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).") 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

No.	'111 Patent Claim 17	Chua '151
		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).") 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.") 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.") 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.") 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.") 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.") 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1382 of 2137

No.	'111 Patent Claim 17	Chua '151
		Cl 7 refers to the packet storage, unit Cll, monitors a TCP SYN flag, and finds a SYN packet (step S301 in FIG. 14).") 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).") 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.") 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.") 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.")

No.	'111 Patent Claim 17	Chua '151
		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.") 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).") 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).") 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	Chua '151
18[a]	The method according to claim 1, wherein the packet comprises distinct header and payload fields,	Chua '151 discloses the method according to claim 1, wherein the packet comprises distinct header and payload fields. *See supra* at Claim 1, 15[a].

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1384 of 2137

No.	'111 Patent Claim 18	Chua '151
18[b]	the header comprises at least the first and second entities addresses in the packet network, and	Chua '151 discloses the header comprises at least the first and second entities addresses in the packet network. For example, Chua '151 discloses the header data include source IP address, destination IP address, source port, and destination port. Chua '151 at 22:53-23:4 ("The switch description objects may include data describing a primary controller for the respective switch, a switch identifier, a switch media access control (MAC) address for the respective switch, an Internet protocol (IP) address for the respective switch, and an array of port description objects for each physical port on the respective switch. For example, for switch 210, a switch description object may include data identifying SDN controller 218 as the primary controller for switch 210, an identifier for switch 210, a MAC address for switch 210, an IP address for switch 210, and an array of port description objects that describe ports of switch 210, such as the ports forming connections 222, 224, 226, and 228. Each of the port description objects may include data describing a port number for the respective port, a port identifier for the respective port, a MAC address of the respective port, and a description of capabilities for the respective port (e.g., a media type for the respective port, a speed for the respective port, whether the respective port is half duplex or full duplex, whether the port supports auto-negotiation, or the like).") Chua '151 at 26:23-43 ("SDN controller 112 may then program the network devices to forward traffic along the determined paths (310). In con-ventional routers, control plane elements of a router program a forwarding information base (FIB) of a data plane of the router. However, SDN controller 112 represents a control plane element that can program data plane elements (e.g., switches) remotely, that is, without being part of or even directly coupled to the data plane elements. For example, SDN controller 112 may program any or all of network devices 108,110 and service devices 116 to forward network traffic having particular header data (e.g., source IP address, destination IP address, source port, destination port, protocol, MPLS label(s), VLAN tags, or the like) to a particular next hop (that is, next network device). Moreover, in accordance with the techniques of this disclosure, SDN controller 112 may program network devices of an SDN to forward traffic according to network paths that were determined based on one or more of connections

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1385 of 2137

No.	'111 Patent Claim 18	Chua '151
		determined between the network devices, service devices among the network devices, zones for packet flows through the network devices, and/or trusted or untrusted packet flows.")
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.	Chua '151 discloses wherein the packet-applicable criterion is that the first entity address, the second entity address, or both match a predetermined address or addresses. For example, Chua '151 discloses the SDN controller 112 determine whether a packet flow can be trusted based on values of packet headers for the packet flows, e.g., values of headers. Chua '151 at 22:53-23:4 ("The switch description objects may include data describing a primary controller for the respective switch, a switch identifier, a switch media access control (MAC) address for the respective switch, an Internet protocol (IP) address for the respective switch, and an array of port description objects for each physical port on the respective switch. For example, for switch 210, a switch description object may include data identifying SDN controller 218 as the primary controller for switch 210, an identifier for switch 210, a MAC address for switch 210, an IP address for switch 210, and an array of port description objects that describe ports of switch 210, such as the ports forming connections 222, 224, 226, and 228. Each of the port description objects may include data describing a port number for the respective port, a port identifier for the respective port, a MAC address of the respective port, and a description of capabilities for the respective port (e.g., a media type for the respective port, a speed for the respective port, whether the respective port is half duplex or full duplex, whether the port supports auto-negotiation, or the like).") Chua '151 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1386 of 2137

No.	'111 Patent Claim 18	Chua '151
		any or all of determination of service devices, determination of zones, and/or determination of trusted packet flows.") Chua '151 at 26:23-43 ("SDN controller 112 may then program the network devices to forward traffic along the determined paths (310). In con-ventional routers, control plane elements of a router program a forwarding information base (FIB) of a data plane of the router. However, SDN controller 112 represents a control plane element that can program data plane elements (e.g., switches) remotely, that is, without being part of or even directly coupled to the data plane elements. For example, SDN controller 112 may program any or all of network devices 108,110 and service devices 116 to forward network traffic having particular header data (e.g., source IP address, destination IP address, source port, destination port, protocol, MPLS label(s), VLAN tags, or the like) to a particular next hop (that is, next network device). Moreover, in accordance with the techniques of this disclosure, SDN controller 112 may program network devices of an SDN to forward traffic according to network paths that were determined based on one or more of connections determined between the network devices, service devices among the network devices, zones for packet flows through the network devices, and/or trusted or untrusted packet flows.")

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1387 of 2137

No.	'111 Patent Claim 19	Chua '151
19	The method according to claim 18, wherein the addresses are Internet Protocol (IP) addresses.	Chua '151 discloses the method according to claim 18, wherein the addresses are Internet Protocol (IP) addresses. For example, Chua '151 discloses source IP and destination IP addresses. *See supra* at Claim 18. Chua '151 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. 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: 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 Create either UDP packet flows or TCP sessions that confirm to the traffic type inferred or discovered form the flowchain configuration 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) 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.
IPR2023-00401, Page 1388 of 2137

No.	'111 Patent Claim 19	Chua '151
		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.") Chua '151 at 22:53-23:4 ("The switch description objects may include data describing a primary controller for the respective switch, a switch identifier, a switch media access control (MAC) address for the respective switch, an Internet protocol (IP) address for the respective switch, and an array of port description objects for each physical port on the respective switch. For example, for switch 210, a switch description object may include data identifying SDN controller 218 as the primary controller for switch 210, an identifier for switch 210, a MAC address for switch 210, an IP address for switch 210, and an array of port description objects that describe ports of switch 210, such as the ports forming connections 222, 224, 226, and 228. Each of the port description objects may include data describing a port number for the respective port, a port identifier for the respective port, a MAC address of the respective port, and a description of capabilities for the respective port (e.g., a media type for the respective port, a speed for the respective port, whether the respective port is half duplex or full duplex, whether the port supports auto-negotiation, or the like).") Chua '151 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. 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.")

~~Orckit Exhibit~~ 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1389 of 2137

No.	'111 Patent Claim 19	Chua '151
		Chua '151 at 26:23-43 ("SDN controller 112 may then program the network devices to forward traffic along the determined paths (310). In con-ventional routers, control plane elements of a router program a forwarding information base (FIB) of a data plane of the router. However, SDN controller 112 represents a control plane element that can program data plane elements (e.g., switches) remotely, that is, without being part of or even directly coupled to the data plane elements. For example, SDN controller 112 may program any or all of network devices 108,110 and service devices 116 to forward network traffic having particular header data (e.g., source IP address, destination IP address, source port, destination port, protocol, MPLS label(s), VLAN tags, or the like) to a particular next hop (that is, next network device). Moreover, in accordance with the techniques of this disclosure, SDN controller 112 may program network devices of an SDN to forward traffic according to network paths that were determined based on one or more of connections determined between the network devices, service devices among the network devices, zones for packet flows through the network devices, and/or trusted or untrusted packet flows.")

No.	'111 Patent Claim 20	Chua '151
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	Chua '151 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. *See supra* at Claim 1, 17[a].

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1390 of 2137

No.	'111 Patent Claim 20	Chua '151
	sequence mask fields, and	
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.	Chua '151 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. For example, Chua '151 discloses packet header data used by the network devices, including TCP values, to enforce policies programmed on the network devices for forwarding the packet, i,e., 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. A person of ordinary skill in the art would understand that a TCP packet would include header information that may be a source TCP port, the destination TCP port, the TCP sequence number, the TCP sequence mask, or any combination thereof. Chua '151 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.
IPR2023-00401, Page 1391 of 2137

No.	'111 Patent Claim 20	Chua '151
		Chua '151 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. 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: 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 Create either UDP packet flows or TCP sessions that confirm to the traffic type inferred or discovered form the flowchain configuration 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) 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 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.") Chua '151 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1392 of 2137

No.	'111 Patent Claim 20	Chua '151
		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 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.") Chua '151 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.") Chua '151 at 26:23-43 ("SDN controller 112 may then program the network devices to forward traffic along the determined paths (310). In con-ventional routers, control plane elements of a router program a forwarding information base (FIB) of a data plane of the router. However, SDN controller 112 represents a control plane element that can program data plane elements (e.g., switches) remotely, that is, without being part of or even directly coupled to the data plane elements. For example, SDN controller 112 may program any or all of network devices 108,110 and service devices 116 to forward network traffic having particular header data (e.g., source IP address, destination IP address, source port, destination port, protocol, MPLS label(s), VLAN tags, or the like) to a particular next hop (that is, next network device). Moreover, in accordance with the techniques of this

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1393 of 2137

No.	'111 Patent Claim 20	Chua '151
		disclosure, SDN controller 112 may program network devices of an SDN to forward traffic according to network paths that were determined based on one or more of connections determined between the network devices, service devices among the network devices, zones for packet flows through the network devices, and/or trusted or untrusted packet flows.")

No.	'111 Patent Claim 21	Chua '151
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.	Chua '151 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. For example, Chua '151 discloses transmitting packets in a network including a Local Area Network (LAN) and Internet service provider (ISP) network. *See supra* at Claim 1. Chua '151 at 10:17-30 ("Using the OpenFlow or other SDN protocol, it is possible to set up the recognition of Extensible Authentication Proto-col (EAP) over Local Area Network (EAPO L) packets (transport mechanism for 802. lX protocol) and to create a conversation with the control platform to emulate an 802.lX exchange. In this manner, system 100 can perform authenti-cation via EAP methods (using certificates or passwords) to identify the end-point (e.g., user and client device, e.g., client device 102). Unlike 802.lX port-based security, however, the use of 802 .1 X in an SD N framework can be used to lock down specific MAC source addresses, and can apply per-flow (that is, per-packet-flow) admission controls, as well as per-flow service orchestration and insertion to scan the traffic for mali-cious or sensitive content, as dictated by a policy engine.") Chua '151 at 12:10-23 ("SDN model 140 may further include data representing a network topology for SDN 106. Rules 142 corresponds to data representing rules for generating flow rules. The network topology and rules of rules 142 may be defined by software-based objects, such as FlowChain objects, Flow Link objects, SubDevice objects, Device objects, DevicePort objects, Por-tRange objects, Source objects, Zone objects, SwitchPort objects,

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1394 of 2137

No.	'111 Patent Claim 21	Chua '151
		and Network objects. Modifier actions, that is, actions that modify the network, may also be modeled by software-based objects, such as push virtual local area network (VLAN) objects, pop VLAN objects, sniffer objects, splitter objects, set multiprotocol label switching (MPLS) label objects, push MPLS label, pop MPLS label, or other actions.") Chua '151 at 17:26-35 ("Events sent to SDN controller 112 may trigger one or more of the SDN controller 112 actions on the same or different network, switch, and/or port than the network, switch, and/or port the initial traffic came from. For example, for DDOS, the identification of the DDOS attack may trigger a block action on the network under attack, trigger a block action to the upstream internet service provider (ISP) network (or a spe-cific upstream device), and/or trigger a counter attack (redi-rection) from a different network, ISP, or network device (physical or virtual).")

No.	'111 Patent Claim 22	Chua '151
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.	Chua '151 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. For example, Chua '151 discloses network traffic flowing between client devices and server devices in both directions. *See supra* at Claim 1. Chua '151 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1395 of 2137

No.	'111 Patent Claim 22	Chua '151
		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 additionally 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.")

No.	'111 Patent Claim 23	Chua '151
23[a]	The method according to claim 22, wherein the server device comprises a web server, and	Chua '151 discloses the method according to claim 22, wherein the server device comprises a web server. For example, Chua '151 discloses a server device including a web server. *See supra* at Claim 22. Chua '151 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 additionally 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.
IPR2023-00401, Page 1396 of 2137

No.	'111 Patent Claim 23	Chua '151
		Chua '151 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.") Chua '151 at 20:18-26 ("SDN controller 218 is configured to program various net-work devices, such as switch 210, in accordance with one or more of the techniques of this disclosure. Although the example of FIG. 3 illustrates only one switch, i.e., switch 210, it should be understood that system 200 may include many interconnected switches or other programmable network devices. Likewise, service devices, such as web proxy 212, IDS 214, and web server 216 may be coupled to various switches (and not necessarily the same switch).") Chua '151 at 20:37-52 ("A user, such as an administrator, determines service devices, such as web proxy 212, IDS 214, and web server 216, to which particular devices (e.g., users of web clients 202) request access. The user may interface with administrator workstation 220 to configure SDN controller 218, to provide information indicative of users and services required for those users. SDN controller 218, in turn, may develop packet flows that ensure web traffic from particular ones of web clients 202 (and, potentially, certain applications executed by web clients 202 and/or with certain destinations) are directed through appropriate service devices. In other words, based on the connections determined between network devices in an SDN of system 200, SDN controller 218 may determine one or more paths for network traffic between the network devices. Likewise, SDN controller 218 may program the network devices to direct network traffic along the paths.") Chua '151 at 21:17-31 ("SDN controller 218 may be configured to program a switch, such as switch 210, to send packets of a particular packet flow to a service device, e.g., one of web proxy 212, IDS 214, and web server 216. For example, switch 210 is communicatively coupled to web proxy 210 by connection 222, which may represent ports 7 and 8. One of

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1397 of 2137

No.	'111 Patent Claim 23	Chua '151
		web clients 202 may request data from a web server, which may be internal to or external to system 200. Web proxy 212 may be configured to act as a proxy device to the web server. That is, SDN controller 218 may program switch 210 to direct net-work traffic destined for the web server to web proxy 212 via connection 222, instead. Web proxy 212 may then act on behalf of the web server, e.g., by caching resources stored by the web server and responding to requests for the resources on behalf of the web server.") Chua '151 at 21:53-58 ("As yet another example, switch 210 is communicatively coupled to web server 216 via connection 228. Connection 228 may represent ports 4 and 6. Web server 216 may gener-ally store and/or manage network accessible resources, such as files, databases, printers, or the like. Web clients 202 may access web server 216 via the SDN of system 200.")
23[b]	wherein the client device comprises a smartphone, a tablet computer, a personal computer, a laptop computer, or a wearable computing device.	Chua '151 discloses wherein the client device comprises a smartphone, a tablet computer, a personal computer, a laptop computer, or a wearable computing device. For example, Chua '151 discloses a client device including a web client that is capable of connecting to the Internet. Chua '151 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 additionally or alternatively include malware detection devices, net-work anti-virus devices, network packet capture and analysis devices, honeypot devices, reflector net

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1398 of 2137

No.	'111 Patent Claim 23	Chua '151
		devices, tar pit devices, domain name service (DNS) and global DNS server devices, mail proxies, and anti-spam devices.") Chua '151 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.") Chua '151 at 20:5-17 ("As shown in FIG. 3, web clients 202 connect to Internet 204. Internet 204 provides access to system 200, which implements the techniques of this disclosure. For example, network devices of system 200 form an SDN controlled by SDN controller 218. SDN controller 218 may generally correspond to SDN controller 112 of FIGS. 1 and 2. In particular, SDN controller 218 (in this example) programs one or more switches (e.g., Pica8 3290 switches, IBM G8264), such as switch 210, to direct traffic through an SDN to other switches, service devices (e.g., web proxy 212, IDS 214, and/or web server 216). In general, SDN controller 218 may be configured to perform any or all of the techniques described above, alone or in any combination.") Chua '151 at 20:37-52 ("A user, such as an administrator, determines service devices, such as web proxy 212, IDS 214, and web server 216, to which particular devices (e.g., users of web clients 202) request access. The user may interface with administrator workstation 220 to configure SDN controller 218, to provide information indicative of users and services required for those users. SDN controller 218, in turn, may develop packet flows that ensure web traffic from particular ones of web clients 202 (and, potentially, certain applications executed by web clients 202 and/or with certain destinations) are directed through appropriate service devices. In other words, based on the connections determined between network devices in an SDN of system 200, SDN controller 218 may determine one or more paths for network traffic between the network devices. Likewise, SDN controller 218 may program the network devices to direct network traffic along the paths.")

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1399 of 2137

No.	'111 Patent Claim 23	Chua '151
		Chua '151 at 21:17-31 ("SDN controller 218 may be configured to program a switch, such as switch 210, to send packets of a particular packet flow to a service device, e.g., one of web proxy 212, IDS 214, and web server 216. For example, switch 210 is communicatively coupled to web proxy 210 by connection 222, which may represent ports 7 and 8. One of web clients 202 may request data from a web server, which may be internal to or external to system 200. Web proxy 212 may be configured to act as a proxy device to the web server. That is, SDN controller 218 may program switch 210 to direct net-work traffic destined for the web server to web proxy 212 via connection 222, instead. Web proxy 212 may then act on behalf of the web server, e.g., by caching resources stored by the web server and responding to requests for the resources on behalf of the web server.") Chua '151 at 21:53-58 ("As yet another example, switch 210 is communicatively coupled to web server 216 via connection 228. Connection 228 may represent ports 4 and 6. Web server 216 may gener-ally store and/or manage network accessible resources, such as files, databases, printers, or the like. Web clients 202 may access web server 216 via the SDN of system 200.")

No.	'111 Patent Claim 24	Chua '151
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.	Chua '151 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. For example, Chua '151 discloses communication between the network devices and SDN controller using standard protocols such as OpenFlow, OSPF, BGP, etc. *See supra* at Claim 22. Chua '151 at 1:25-41 ("Recently, software defined networks have been developed as a way to deploy and operate networks and develop new networking applications. In general, software defined net-works involve the use of a standalone controller that performs the control functionality for a set of network devices. As an example of software defined

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1400 of 2137

No.	'111 Patent Claim 24	Chua '151
		networking, in the case of routing, rather than routers performing individual analyses to determine routes through the network, the controller can determine the routes and program other devices in the net-work to behave according to the determinations made by the controller. Different protocols may be used to implement software defined networking, including open protocols like OpenFlow, and proprietary protocols from network vendors. OpenFlow is described in McKeown et al., "OpenFlow: Enabling Innovation in Campus Networks," http://ww-w.openflow.org//documents/openflow-wp-latest.pdf, which is hereby incorporated by reference in its entirety.") Chua '151 at 6:36-52 ("Users of SDN controller 112 may be provided with the ability to create programs that are executable by SDN con-troller 112. For example, a software development kit (SDK) may be provided for creating programs that can be executed by SDN controller 112. In this manner, users or other third parties may utilize an environment to write applications that can be executed by SDN controller 112. For example, SDN controller 112 itself may host one or more L4-L 7 services, rather than (or in addition to) services provided by service devices 116. As another example, a L3 application (such as Quagga, a Layer 3 routing stack) may be executed by SDN controller 112. In this case, the Quagga routing stack would be running a standard routing protocol such as OSPF (Open Shortest Path Forwarding) or BGP (Border Gateway Protocol) and would update its routing tables like before, except that it would use the SDN controller 112 to command the network equipment to update their forwarding tables.") Chua '151 at 9:27-36 ("SDN controller 112 may provide a platform that encom-passes various capabilities for custom software-defined net-working deployments. Such capabilities may include: a flex-ible network controller with the ability to communicate using standards-based or proprietary control protocols to control network flows on networking devices, both physical and vir-tual; a standards-based policy engine tied into the network controller logic; an extensible API for programmatic access from applications; and/or additional networking services that can execute on SDN controller 112.") Chua '151 at 9:63-10:3 ("Thus, this disclosure also describes certain techniques that may be used to overcome the above noted problems relating to access control in a software defined network. In accordance with these techniques, a platform (such as SDN controller 112) may have network flow control capabilities on one end (e.g., via the OpenFlow (or other)

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1401 of 2137

No.	'111 Patent Claim 24	Chua '151
		protocol, interacting with both physical and virtual network access devices (switches for instance)), and a policy and identity stack on the other end.") Chua '151 at 10:17-30 ("Using the OpenFlow or other SDN protocol, it is possible to set up the recognition of Extensible Authentication Proto-col (EAP) over Local Area Network (EAPO L) packets (transport mechanism for 802. lX protocol) and to create a conversation with the control platform to emulate an 802.lX exchange. In this manner, system 100 can perform authenti-cation via EAP methods (using certificates or passwords) to identify the end-point (e.g., user and client device, e.g., client device 102). Unlike 802.lX port-based security, however, the use of 802 .1 X in an SD N framework can be used to lock down specific MAC source addresses, and can apply per-flow (that is, per-packet-flow) admission controls, as well as per-flow service orchestration and insertion to scan the traffic for mali-cious or sensitive content, as dictated by a policy engine.")

No.	'111 Patent Claim 27	Chua '151
27	The method according to claim 1, wherein the network node comprises a router, a switch, or a bridge.	Chua '151 discloses the method according to claim 1, wherein the network node comprises a router, a switch, or a bridge. For example, Chua '151 discloses network devices which may comprise switches or routers. *See supra* at Claim 1. Chua '151 at 3:33-54 ("In another example, a method includes obtaining, by a controller device of a software defined network (SDN), data representative of physical network devices in the SDN, wherein the physical network devices comprise one or more switches and one or more controllers including the controller device, instantiating, by the controller device, software-based controller objects for each of the one or more controllers, wherein the software-based controller objects each comprise data representative of a respective instance name, a respective server hostname, and a respective server host port, instanti-ating, by the controller device, software-based switch description objects for each of the one or more switches, wherein the software-based switch description objects each comprise data representative of a respective primary controller corresponding to a controller of the one or more control-lers, a respective switch identifier, a respective switch media

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1402 of 2137

No.	'111 Patent Claim 27	Chua '151
		access control (MAC) address, a respective Internet protocol (IP) address, and a respective array of port description objects for each physical port on the respective switch, and manag-ing, by the controller device, at least some of the switches using the switch description objects and the controller objects.") Chua '151 at 3:55-4:10 ("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 obtain data representative of physical network devices in the SDN, wherein the physical network devices comprise one or more switches and one or more controllers including the controller device, instantiate software-based controller objects for each of the one or more controllers, wherein the software-based controller objects each comprise data representative of a respective instance name, a respective server hostname, and a respective server host port, instantiate software-based switch description objects for each of the one or more switches, wherein the software-based switch description objects each comprise data representative of a respective primary control-ler corresponding to a controller of the one or more control-lers, a respective switch identifier, a respective switch media access control (MAC) address, a respective Internet protocol (IP) address, and a respective array of port description objects for each physical port on the respective switch, and manag-ing, via the network interfaces, at least some of the switches using the switch description objects and the controller objects.") Chua '151 at 5:32-49 ("FIG. 1 is a block diagram illustrating an example system 100 in which various techniques of this disclosure may be used. In this example, system 100 includes software defined network (SDN) 106, which includes network devices 108, 110 and service devices 116. Network devices 108, 110 may comprise switches, and other devices (not shown). These network devices can be physical instantiations or virtual instantiations. SDN 106 may also include other types of devices, such as routers, load balancers, various L4-L 7 net-work devices, or even multi-tenant capable devices, among other network devices. Again, these network devices can be either of physical or virtual instantiations. In general, network devices of SDN 106, such as network devices 108,110, can be programmed to forward network traffic. System 100 also includes software defined network (SDN) controller 112. Administrator 114 uses SDN controller 112 to control (that is, program) network devices of SDN 106, such as network devices 108, 110.")

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1403 of 2137

No.	'111 Patent Claim 27	Chua '151
		Chua '151 at 11:13-27 ("SDN modeling unit 132 receives data representative of physical network devices in SDN 106, such as network device 108, network device 110, and service devices 116 (FIG. 1). SDN modeling unit 132 may also receive data representative of controller devices, including SDN controller 112, that control devices of SDN 106. SDN modeling unit 132 forms SDN model 140, which includes data that represents the physical devices of SDN 106. SDN model 140 may include software-based objects (that is, data for software objects con-forming to the object oriented programming (OOP) para-digm) that represent the physical devices in a physical view, as well as a logical view of SDN 106. As noted above, the physical view includes devices such as controllers, switches, and devices connected to the switches (e.g., service devices 116) managed by controllers such as SDN controller 112.") Chua '151 at 19:60-20:4 ("FIG. 3 is a conceptual diagram illustrating an example 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, 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.") Chua '151 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.")

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1404 of 2137

No.	'111 Patent Claim 28	Chua '151
28	The method according to claim 1, wherein the packet network is an Internet Protocol (IP) network, and the packet is an IP packet.	Chua '151 discloses the method according to claim 1, wherein the packet network is an Internet Protocol (IP) network, and the packet is an IP packet. For example, Chua '151 discloses communication of packet traffic using an IP network and IP packet values. *See supra* at Claim 1. Chua '151 at 3:55-4:10 ("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 obtain data representative of physical network devices in the SDN, wherein the physical network devices comprise one or more switches and one or more controllers including the controller device, instantiate software-based controller objects for each of the one or more controllers, wherein the software-based controller objects each comprise data representative of a respective instance name, a respective server hostname, and a respective server host port, instantiate software-based switch description objects for each of the one or more switches, wherein the software-based switch description objects each comprise data representative of a respective primary control-ler corresponding to a controller of the one or more control-lers, a respective switch identifier, a respective switch media access control (MAC) address, a respective Internet protocol (IP) address, and a respective array of port description objects for each physical port on the respective switch, and manag-ing, via the network interfaces, at least some of the switches using the switch description objects and the controller objects.") Chua '151 at 13:12-41 ("A Source object may define the parameters needed to specify the network traffic source. Each Source object may include data representative of an identifier (e.g., a text string name) for the source, and a set of one or more zones. In particular, one or more flows may be grouped into zones. The zones may represent administrative or security areas that ensure that network managers or customers cannot interfere with each other from a switch control or data plane perspec-tive. For example, a test or research SDN environment may operate entirely within a single zone, and a simple production deployment may have a secure administrative zone used for management purposes and an open operational zone used for dynamic network programming. Multi-tenant or multi-appli-cation deployments may

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1405 of 2137

No.	'111 Patent Claim 28	Chua '151
		utilize unique zones for each cus-tomer or function or application. There is no fundamental restriction to how zones can be partitioned. In some examples, zones are partitioned on a per IP address, per port, per VLAN, or possibly per subnet basis, but this concept can be usefully extended on a per protocol basis or to any other packet header field or per application or per user. SDN con-troller 112 may ensure that a flow rule written for one zone incorporates sufficient additional match rules to prevent flows from other zones from being processed inappropriately. Zones may be ordered by numerical priority to avoid confu-sion that may otherwise occur when overlapping partitions are created. Each zone may support OpenFlow, normal dis-tributed, hybrid, or even alternate protocol types indepen-dently. Zones can also be used to partition off areas where no traffic is expected, allowing illegitimate traffic to be black holed, spanned, or redirected elsewhere for further action.") Chua '151 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.") Chua '151 at 22:53-23:4 ("The switch description objects may include data describing a primary controller for the respective switch, a switch identifier, a switch media access control (MAC) address for the respective switch, an Internet protocol (IP) address for the respective switch, and an array of port description objects for each physical port on the respective switch. For example, for switch 210, a switch description object may include data identifying SDN controller 218 as the primary controller for switch 210, an identifier for switch 210, a MAC

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1406 of 2137

No.	'111 Patent Claim 28	Chua '151
		address for switch 210, an IP address for switch 210, and an array of port description objects that describe ports of switch 210, such as the ports forming connections 222, 224, 226, and 228. Each of the port description objects may include data describing a port number for the respective port, a port identifier for the respective port, a MAC address of the respective port, and description of capabilities for the respective port (e.g., a media type for the respective port, a speed for the respective port, whether the respective port is half duplex or full duplex, whether the port supports auto-negotiation, or the like).") Chua '151 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. 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.") Chua '151 at 26:23-43 ("SDN controller 112 may then program the network devices to forward traffic along the determined paths (310). In con-ventional routers, control plane elements of a router program a forwarding information base (FIB) of a data plane of the router. However, SDN controller 112 represents a control plane element that can program data plane elements (e.g., switches) remotely, that is, without being part of or even directly coupled to the data plane elements. For example, SDN controller 112 may program any or all of network devices 108,110 and service devices 116 to forward network traffic having particular header data (e.g., source IP address, destination IP address, source port, destination port, protocol, MPLS label(s), VLAN tags, or the like) to a particular next hop (that is, next network device). Moreover, in accordance with the techniques of this

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1407 of 2137

No.	'111 Patent Claim 28	Chua '151
		disclosure, SDN controller 112 may program network devices of an SDN to forward traffic according to network paths that were determined based on one or more of connections determined between the network devices, service devices among the network devices, zones for packet flows through the network devices, and/or trusted or untrusted packet flows.")

No.	'111 Patent Claim 29	Chua '151
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.	Chua '151 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. For example, Chua '151 discloses communication of packet traffic using a TCP network and TCP packet values. *See supra* at Claim 28. Chua '151 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.") Chua '151 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1408 of 2137

No.	'111 Patent Claim 29	Chua '151
		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. 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: 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 Create either UDP packet flows or TCP sessions that confirm to the traffic type inferred or discovered form the flowchain configuration 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) 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 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.") Chua '151 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1409 of 2137

No.	'111 Patent Claim 29	Chua '151
		examples, flow management server 256 may program network 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 servers), 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.") Chua '151 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. 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.") Chua '151 at 26:23-43 ("SDN controller 112 may then program the network devices to forward traffic along the determined paths (310). In con-ventional routers, control plane elements of a router program a forwarding information base (FIB) of a data plane of the router. However, SDN controller 112 represents a control plane element that can program data plane elements (e.g., switches) remotely, that is, without being part of or even directly coupled to the data plane elements. For example, SDN controller 112 may program any or all of network devices 108,110 and service devices 116 to forward network traffic having particular header data (e.g., source IP address, destination IP address, source port, destination port, protocol, MPLS label(s), VLAN tags, or the like) to a particular next hop (that is, next network device). Moreover, in accordance with the techniques of this

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1410 of 2137

No.	'111 Patent Claim 29	Chua '151
		disclosure, SDN controller 112 may program network devices of an SDN to forward traffic according to network paths that were determined based on one or more of connections determined between the network devices, service devices among the network devices, zones for packet flows through the network devices, and/or trusted or untrusted packet flows.")

No.	'111 Patent Claim 30	Chua '151
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;	Chua '151 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. For example, Chua '151 discloses the transmission of, including receiving by the network devices, a second set of packet flows. *See supra at* Claim 1, 1[c]. Chua '151 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).")
30[b]	checking, by the network node, if any one of the one or more additional packets satisfies the criterion;	Chua '151 discloses checking, by the network node, if any one of the one or more additional packets satisfies the criterion. *See supra at* Claim 1[d], 30[a].
30[c]	responsive to an additional packet not satisfying the criterion, sending, by the	Chua '151 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. *See supra at* Claim 1[e], 30[a].

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1411 of 2137

No.	'111 Patent Claim 30	Chua '151
	network node over the packet network, the additional packet to the second entity; and	
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.	Chua '151 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. *See supra at* Claim 1[f], 30[a].

No.	'111 Patent Claim 31	Chua '151
31[a]	The method according to claim 1, wherein the packet network is a Software Defined Network (SDN),	Chua '151 discloses the method according to claim 1, wherein the packet network is a Software Defined Network (SDN). For example, Chua '151 discloses techniques and methods related to software defined networks (SDNs). *See supra* at Claim 1. Chua '151 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.") Chua '151 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1412 of 2137

No.	'111 Patent Claim 31	Chua '151
		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.") Chua '151 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.") Chua '151 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.") Chua '151 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1413 of 2137

No.	'111 Patent Claim 31	Chua '151
		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	Chua '151 discloses the packet is routed as part of a data plane. For example, Chua '151 discloses forwarding packets over a data plane to various network destinations. Chua '151 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.") Chua '151 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.") Chua '151 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1414 of 2137

No.	'111 Patent Claim 31	Chua '151
		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.") Chua '151 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.") Chua '151 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.")

No.	'111 Patent Claim 31	Chua '151
31[c]	the network node communication with the controller serves as a control plane.	Chua '151 discloses the network node communication with the controller serves as a control. For example, Chua '151 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. Chua '151 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.") Chua '151 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.") Chua '151 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.") Chua '151 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

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1416 of 2137

No.	'111 Patent Claim 31	Chua '151
		the techniques of this disclosure. Flow management system 250 (also referred to as "system 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.") Chua '151 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).") Chua '151 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 disclosure. Flow management system 250 (also referred to as "system 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.")

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1417 of 2137

EXHIBIT D-10
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. 2005/0210533 to Copeland et al. ("Copeland")

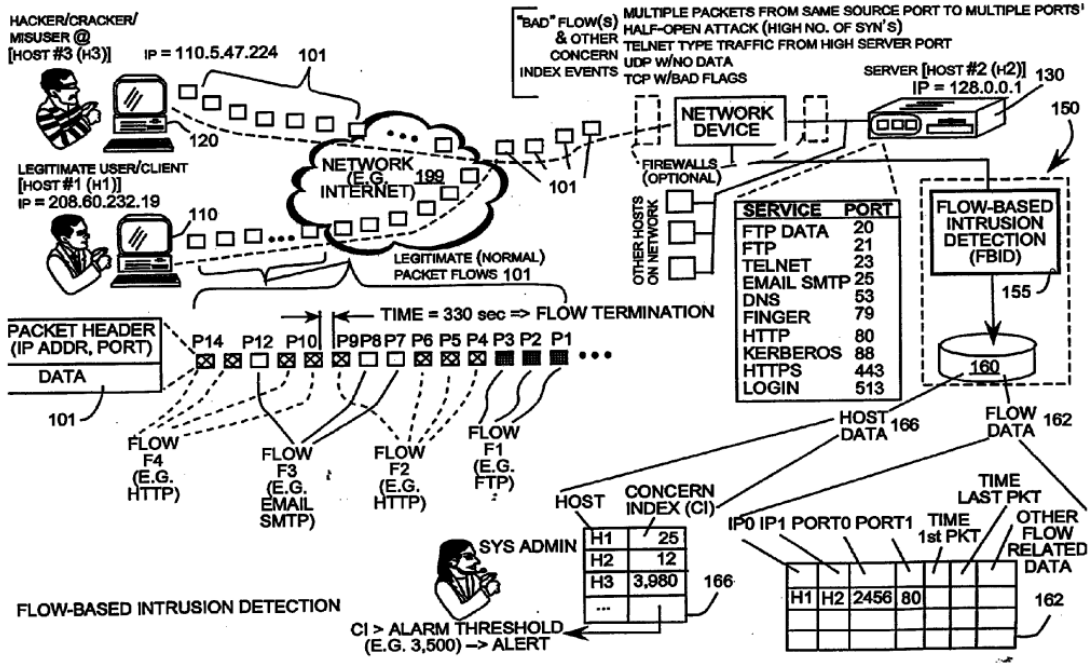
As shown in the chart below, all Asserted Claims of the '111 Patent are invalid under (1) AIA-35 U.S.C. § 102 (a) because Copeland meets each element of those claims, and/or (2) 35 U.S.C. § 103 because Copeland renders those claims obvious either alone, or in combination with the knowledge of a person 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 Copeland titled "Packet Sampling Flow-Based Detection Of Network Intrusions", which was filed on May 26, 2005, and published on September 22, 2005.

Motivations to combine the disclosures in Copeland 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 Preliminary Infringement Disclosure, 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	Copeland
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:	Copeland 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. For example, Copeland discloses steps taken in an operating environment in which packets are transmitted between first and second hosts connected to the Internet, under the monitoring of a flow based intrusion detection system of a monitoring device. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. Copeland at [0042] ("Turning to the figures, in which like numerals indicate like elements throughout the several figures, FIG. 1 provides an overview of a flow-based intrusion detection system or engine 155 in accordance with an exemplary embodiment of the present invention. The flow-based intrusion detection system 155 monitors network computer communications. The network computer communications are routed via a known global computer network commonly known as the Internet 199. In accordance with an aspect of the invention, the intrusion detection engine 155 is incorporated into a monitoring appliance 150, together with a database 160 that stores information utilized in the intrusion detection methodology.") Copeland at [0043] ("The operating environment of the intrusion detection system 155 is contemplated to have numerous hosts connected by the Internet 199, e.g. Host #1, Host #2, Host #3 (also referred to as Hl-H3 respectively). Hosts are any computers that have full two-way access to other computers on the Internet 199 and have their own unique IP address. For example Host #1 has an exemplary IP address of 208.60.239.19. The Internet 199 connects clients 110 with a host server 130 in known client/server relationship.") Copeland at [0044] ("In a typical configuration, some computers are referred to as "servers", while others are referred to as "clients." A server computer such as Host #2130 typically provides responses to requests from client computers and provides services, data, resources, and the like. While a client computer such as Host #1110 typically requests and utilizes the services, data, resources, and the like provided by the server.") Copeland at [0045] ("It is known in the art to send communications between hosts via the Internet 199. The Internet Protocol (IP) is the method by which data is sent from one host

No.	'111 Patent Claim 1	Copeland
		computer to another on the Internet 199. Each host on the Internet 199 has an IP address that uniquely identifies it from all other computers. When data is transmitted, the message gets divided into packets 101. Packets 101 are discussed in more detail in reference to FIG. 2.") Copeland at Figure 1  FIG. 1
1[a]	sending, by the controller to the network node over the packet network, an instruction and a	Copeland discloses sending, by the controller to the network node over the packet network, an instruction and a packet-applicable criterion. For example, Copeland discloses use of an intrusion detection engine that operates on Netflow in a distributed nature to apply rules and algorithms to network devices to send packets and sample packet headers to the monitoring appliance.

No.	'111 Patent Claim 1	Copeland
	packet-applicable criterion;	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.") Copeland at [0025] ("The present invention provides an accurate and reliable method for detecting network attacks through the use of sampled packet headers provided by a source such as defined in sFlow and is based in large part on "flows" as opposed to signatures or anomalies. By utilizing the host and flow information structures inherent with flow-based analy-sis and applying rules of statistical significance and analysis, the intrusion detection system can operate with sampled data such as provided by sFlow to provide hybrid solution that combines many of the benefits of a packet capture imple-mentation with the distributed nature of an IDS that operates on Netflow to provide an enhanced wide-area IDS solution.") Copeland at [0062] ("In accordance with an aspect of the invention, a variable denominated as "concern index" (CI) is provided in association with each host identified by the intrusion detec-tion engine 155. This concern index CI variable is used to accumulate values associated with various abnormal events occurring on the network, e.g. when a particular flow is deemed unusual or when particular types of anomalous events occur. At such time as the cumulated value of the CI variable for a particular host exceeds a predetermined threshold value, that host may be considered a sufficient threat to warrant generating an alert or alarm and action taken.") 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

No.	'111 Patent Claim 1	Copeland
		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.") 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.") Copeland at [0071] ("A modified implementation of the sFlow sampling algorithm comprises sending the first packet sent by each respective host in a flow in addition to the normal packet sampling that is accomplished by the sampling algorithm. This particular aspect augments the sampled data by providing the IDS with the exact starting time of each flow on a network and thereby provides greater detail about a particular host and their communication tendencies.") Copeland at [0072] ("The conventional format of an sFlow datagram record 259 comprises a record header 262 that is followed by one or more packet samples. Each packet sample comprises an sFlow sample header 270 and packet data 290, wherein the packet data 290 comprises a sampled packet header 292, as shown in FIG. 2A. Also illustrated in FIG. 2A, the sFlow sample header can comprise data relating to the sFlow sample data type 272, the sample sequence number 274, the data sampling rate 276, the total number of packets observed 278, the number of packets dropped 280, the destination input port 282, the output port 284 and the data sampling type 286. On network devices 135 the sFlow agent will collect and transmit the sFlow record 259 on a regular basis, normally forwarding all packet samples within one second of collection.") Copeland at [0113] ("The computer programs or threads are executed on a computer system 800 constructed as described in reference to FIG. 8, which illustrates a suitable exemplary computer system that may be utilized to construct a monitoring appliance 150 including an intrusion detection engine 155, or a separately implemented intrusion detection engine

No.	'111 Patent Claim 1	Copeland
		Although the described embodiments are generally described in reference to an Internet-accessible computer system that is dedicated to implementing the engine 155, those of ordinary skill in the art will recognize that the present invention can be implemented in computer program code that can execute in conjunction with other program modules in various types of general purpose, special purpose, or dedicated computers. Accordingly, it will be under-stood that the terms "computer," "operating system," and "application program" include all types of computers and the program modules designed to be implemented by the computers.") 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.")
1[b]	receiving, by the network node from the controller, the instruction and the criterion;	Copeland discloses receiving, by the network node from the controller, the 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;	Copeland discloses receiving, by the network node from the first entity over the packet network, a packet addressed to the second entity. For example, Copeland discloses receiving packets at a network device that is transmitting packets between hosts such as clients to servers. Copeland further discloses that the packet may be transmitted to a second host. Copeland at [0044] ("In a typical configuration, some computers are referred to as "servers", while others are referred to as "clients." A server computer such as Host #2130 typically provides responses to requests from client computers and provides services, data, resources, and the like. While a client computer such as Host #1110 typically requests and utilizes the services, data, resources, and the like provided by the server.")

No.	'111 Patent Claim 1	Copeland
		Copeland at [0045] ("It is known in the art to send communications between hosts via the Internet 199. The Internet Protocol (IP) is the method by which data is sent from one host computer to another on the Internet 199. Each host on the Internet 199 has an IP address that uniquely identifies it from all other computers. When data is transmitted, the message gets divided into packets 101. Packets 101 are discussed in more detail in reference to FIG. 2.") Copeland at [0047] ("As shown, a client 110 communicates with a server 130 by sending packets 101 of data. A packet 101 is a unit of data that is routed between an origin and destination. As illustrated, messages are segmented into numerous packets 101 and routed via the Internet 199 to the receiving host. The receiving host reassembles the stream of packets 101 to recreate the original message, which is then handled by application programs running on the receiving computer system.")
1[d]	checking, by the network node, if the packet satisfies the criterion;	Copeland discloses checking, by the network node, if the packet satisfies the criterion. For example, Copeland receiving packets at the network device, which are checked to determine if the packet is suspicious, and Copeland further discloses that such analysis includes determining if the headers of the packets meet predetermined criteria. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Cisco EtherChannel System 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. 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.")

No.	'111 Patent Claim 1	Copeland
		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.") 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.") 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: (1) both port numbers match and no port is marked as the "server" port, or (2) the port number previously marked as the "server" port matches, or (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).") 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

No.	'111 Patent Claim 1	Copeland
		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."") 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.") 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: (a). both port numbers match and no port is marked as the "server" port, or (b). the port number previously marked as the "server" port matches, or (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).") Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Copeland 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. For example, Swenson discloses determining by the steering device packet flows that match one or more signatures, conditions, or criteria of the packet.

No.	'111 Patent Claim 1	Copeland
		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.") 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.") 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

No.	'111 Patent Claim 1	Copeland
		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.") 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.") 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.")

No.	'111 Patent Claim 1	Copeland
		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.") 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

No.	'111 Patent Claim 1	Copeland
		or image objects that need to be redirected, the steering device 130 would forward the response to the user device 110.") 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.") For example, Kempf discloses determining by the network element if the packet header field match an associated 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 be provided by any

No.	'111 Patent Claim 1	Copeland
		type of network including a local area network (LAN) or a wide area network (WAN), such as the Internet.") 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.") 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.") 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.") 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

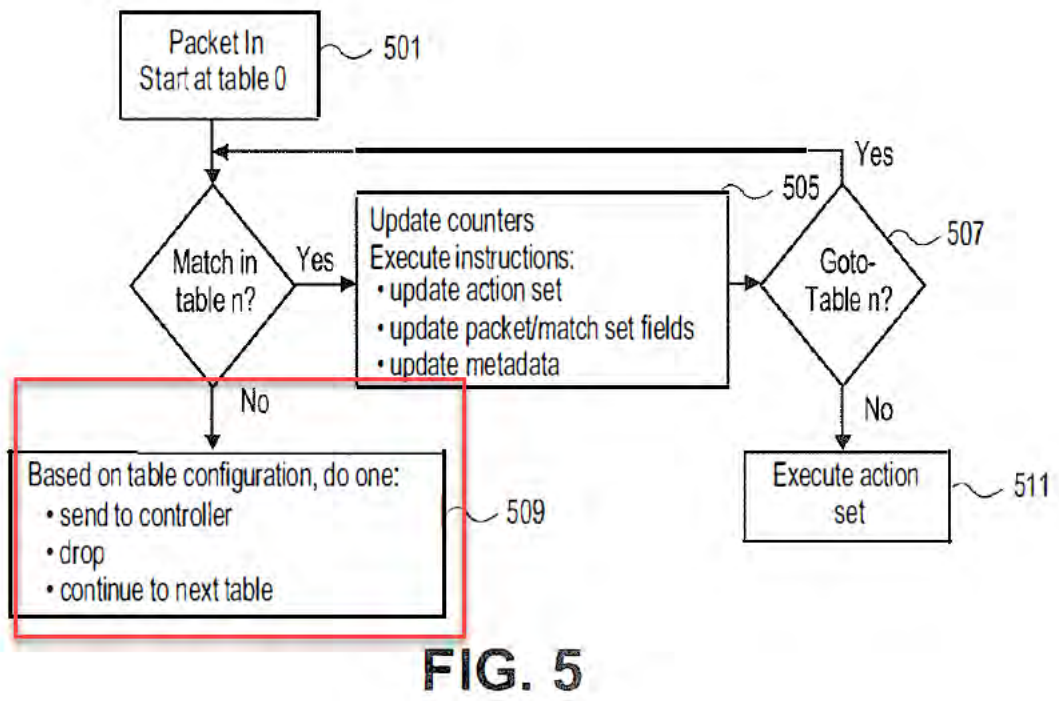
No.	'111 Patent Claim 1	Copeland
		are executed at the end 403 of the packet processing pipeline. An action allows the writing of data to a metadata register, which is carried along in the packet processing pipe-line like the packet header.") Kempf at [0051] ("FIG. 5 is a flowchart of one embodiment 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.")
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	Copeland discloses responsive to the packet not satisfying the criterion, sending, by the network node over the packet network, the packet to the second entity. For example, Copeland discloses where packets do not have prior data records or predetermined concern index forwarding the packet to its originally intended destination host. A person of ordinary skill in the art would have understood that a packet that does not have prior data records or predetermined concern index would 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 Cisco EtherChannel 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.

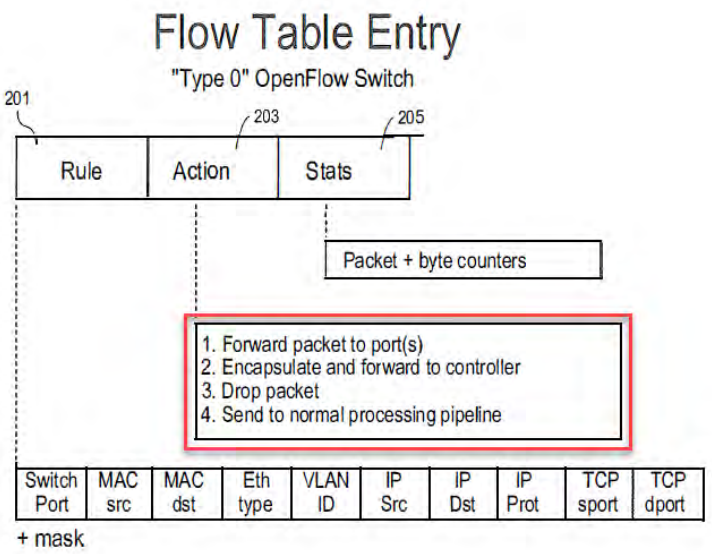
No.	'111 Patent Claim 1	Copeland
		Copeland at [0065] ("The intrusion detection engine 155 analyzes the flow data 160 to determine if the flow appears to be legitimate traffic or possible suspicious activity. Flows with suspicious activity are assigned a predetermined concern index (CI) value based upon 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.") 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.") Copeland at [0130] ("If no prior data record exists in the flow data structure 162 that matches the current packet, a new flow data record is created in the flow data structure 162 using the IP addresses and port numbers from the current packet, and is linked to the end of the appropriate linked list of flow records. The time that the flow started, i.e. the first packets capture time, is written into the record as the "start" time, in a predetermined field of the data record. The time of each packet is written into the record "last", overwriting the previous value.") Copeland at [0165] ("Step 916 is followed by step 918, in which the flow records in the flow data structure 162 are updated. The time that the flow started, the packet capture time, is written into the record "start." The flow data structures updated by the packet classifier thread is discussed in detail in reference to FIG. 5. Step 918 is followed by step 920, where the flow is tested to determine if the packet received belongs to a primary flow. If the packet does not belong to a primary flow, then the "No" branch is followed to step 912. If the packet does belong to a primary flow, then the "Yes" branch is followed to step 922. Step 922 performs data scaling on the traffic statistics relating to the packet by multiplying the number of bytes

No.	'111 Patent Claim 1	Copeland
		and packets by the sample rate. Step 924 follows step 922 and updates the appropriate hosts statistics with the scaled byte and packet counts. Step 924 is returned to step 912, in which the thread 510 determines if a new packet is available.") Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Copeland 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 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.") 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.") 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

No.	'111 Patent Claim 1	Copeland
		the matching rule to a subset of the available fields by using wildcards for the unwanted fields.") 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.") 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.") 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

No.	'111 Patent Claim 1	Copeland
		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.") 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.") 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.") 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.")

No.	'111 Patent Claim 1	Copeland
		Kempf at Figure 5 (annotation added)  FIG. 5 Kempf at Figure 2 (annotation added)

No.	'111 Patent Claim 1	Copeland
		 FIG. 2 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. 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

No.	’111 Patent Claim 1	Copeland
		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.”) 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.”) 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

No.	'111 Patent Claim 1	Copeland
		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.") 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.") 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.") 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

No.	'111 Patent Claim 1	Copeland
		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.") 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.") 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.

No.	'111 Patent Claim 1	Copeland
		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.") 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	Copeland 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. For example, Copeland discloses sending packets to the monitoring appliance operating a flow based intrusion detection engine that matches predetermined value entries associated with a concern index.

No.	'111 Patent Claim 1	Copeland
	instruction and is other than the second entity.	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.") 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.") 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: (1) both port numbers match and no port is marked as the "server" port, or (2) the port number previously marked as the "server" port matches, or (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).") 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

No.	'111 Patent Claim 1	Copeland
		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."") 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.") 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: (a). both port numbers match and no port is marked as the "server" port, or (b). the port number previously marked as the "server" port matches, or (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 2	Copeland
2[a]	The method according to claim 1, wherein the instruction is 'probe', 'mirror', or 'terminate' instruction, and	Copeland discloses the method according to claim 1, wherein the instruction is 'probe', 'mirror', or 'terminate' instruction. For example, Copeland discloses probing, copying, and terminating rules configured on the network device. 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.

No.	'111 Patent Claim 2	Copeland
		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.") 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.") 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.") 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.") 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

No.	'111 Patent Claim 2	Copeland
		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.") 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.") 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.") 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.") 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 2	Copeland
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.	Copeland 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. For example, Copeland discloses terminating a communication session by stopping packet exchange between host devices and to the monitoring appliance. Copeland at [0072] ("The conventional format of an sFlow datagram record 259 comprises a record header 262 that is followed by one or more packet samples. Each packet sample comprises an sFlow sample header 270 and packet data 290, wherein the packet data 290 comprises a sampled packet header 292, as shown in FIG. 2A. Also illustrated in FIG. 2A, the sFlow sample header can comprise data relating to the sFlow sample data type 272, the sample sequence number 274, the data sampling rate 276, the total number of packets observed 278, the number of packets dropped 280, the destination input port 282, the output port 284 and the data sampling type 286. On network devices 135 the sFlow agent will collect and transmit the sFlow record 259 on a regular basis, normally forwarding all packet samples within one second of collection.") 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.") 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.") 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.")

No.	'111 Patent Claim 2	Copeland
		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.") 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 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.") 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.")

No.	'111 Patent Claim 3	Copeland
3[a]	The method according to claim 1, wherein the	Copeland discloses the method according to claim 1, wherein the instruction is a 'probe', a 'mirror', or a 'terminate' instruction.

No.	'111 Patent Claim 3	Copeland
	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.	Copeland 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. For example, Copeland discloses making a copy of a packet. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Copeland 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. 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.") 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.") Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Copeland 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.

No.	'111 Patent Claim 3	Copeland
		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. 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: Allow-explicitly allow a certain network flow to proceed to its destination Block-explicitly block a certain flow from traversing SDN 106 Mirror-allow the traffic, but send a copy of the traffic for deeper inspection or recording to, e.g., one of service devices 116 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. Transform-modify or translate values of headers of packets in the network flow Encapsulate-encapsulate packets in the network flow with a particular header") 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.

No.	'111 Patent Claim 3	Copeland
		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.") 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.") 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. 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

No.	'111 Patent Claim 3	Copeland
		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.") 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.") 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

No.	'111 Patent Claim 3	Copeland
		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.") 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.") 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.")

No.	'111 Patent Claim 3	Copeland
		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	Copeland
4[a]	The method according to claim 1, wherein the instruction is 'probe', 'mirror', or 'terminate' instruction, and	Copeland discloses the method according to claim 1, wherein the instruction is 'probe', 'mirror', or 'terminate' instruction. *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;	Copeland 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. For example, Copeland discloses probe sessions that result in the network device sending packets to the monitoring device when they match 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 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.")

No.	'111 Patent Claim 4	Copeland
		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.") 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: (1) both port numbers match and no port is marked as the "server" port, or (2) the port number previously marked as the "server" port matches, or (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).") 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."")

No.	'111 Patent Claim 4	Copeland
		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.") 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.") 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: (a). both port numbers match and no port is marked as the "server" port, or (b). the port number previously marked as the "server" port matches, or (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).")
4[c]	responsive to receiving the packet, analyzing the packet, by the controller;	Copeland discloses responsive to receiving the packet, analyzing the packet, by the controller. For example, Copeland discloses analyzing packets received by the intrusion detection engine on the monitoring appliance. 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

No.	'111 Patent Claim 4	Copeland
		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.") 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.") 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.") 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 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

No.	'111 Patent Claim 4	Copeland
		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.") 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.") 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.") 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.) 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,

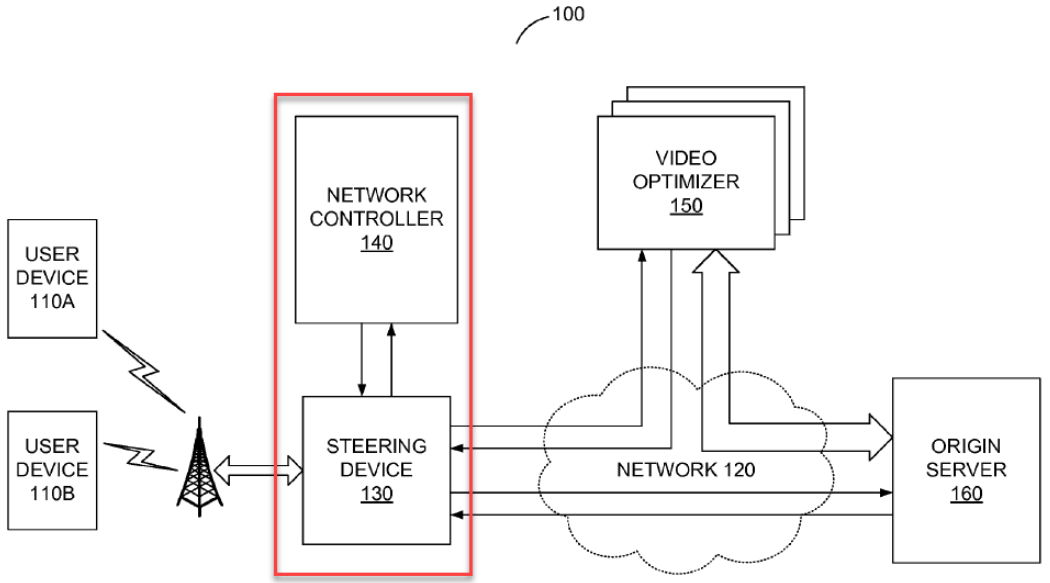
No.	'111 Patent Claim 4	Copeland
		communications for abnormal flow characteristics, attacks can be determined without the need for resource-intensive packet data analysis. A flow can be determined from the packets 101 that are transmitted between two hosts utilizing a single service. The addresses and port numbers of communications are easily discerned by analysis of the header information in a datagram.") 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.") 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.") 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.") 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

No.	'111 Patent Claim 4	Copeland
		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	Copeland discloses sending the packet, by the controller, to the network node. For example, Copeland discloses a network device as the location for connecting the monitoring appliance to the network for monitoring network traffic. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Copeland 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. 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.") Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Copeland 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 such an example. 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 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

No.	'111 Patent Claim 4	Copeland
		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.") 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.") 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

No.	'111 Patent Claim 4	Copeland
		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.") 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.") 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.")

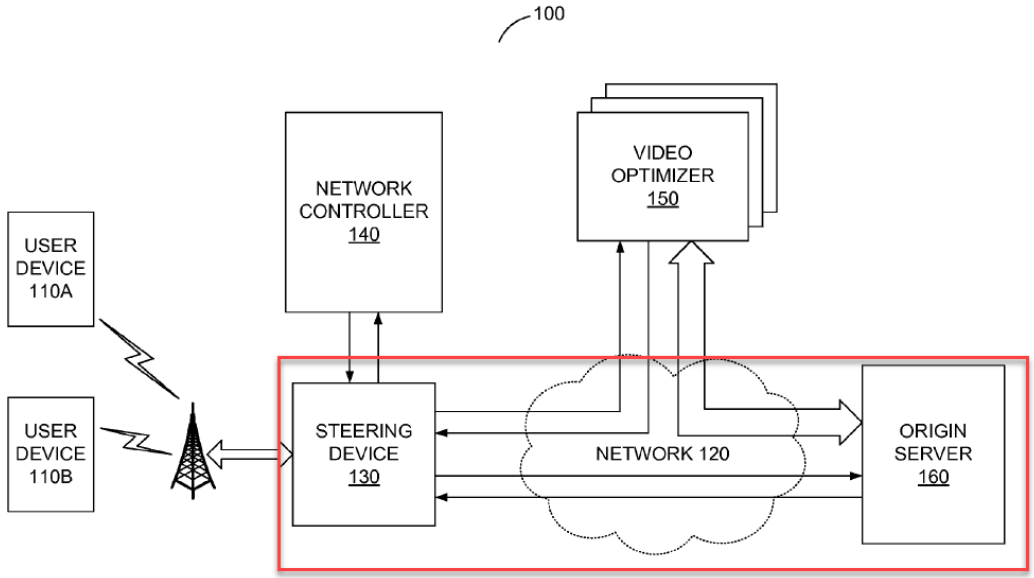
No.	'111 Patent Claim 4	Copeland
		Swenson at [0060] ("If the flow is deemed of interest, the steering device 130 is notified to steer the flow through the network controller 140. This is known as the "continue" working mode for bandwidth monitoring. In the "continue" mode, the network controller 140 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.") 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.") Swenson at Figure 1 (annotation added)

No.	'111 Patent Claim 4	Copeland
		 FIG. 1
4[e]	responsive to receiving the packet, sending the packet, by the network node, to the second entity.	Copeland discloses responsive to receiving the packet, sending the packet, by the network node, to the second entity. For example, Copeland discloses forwarding network traffic to its originally intended location once it has been passed through the monitoring appliance filter and then back to the network device which serves as the location for connection the monitoring appliance to the network. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Copeland is found to not meet this limitation, responsive to receiving the packet, sending the packet, by the network node, to the second entity would have been obvious to a person having ordinary skill in the art, as explained below.

No.	'111 Patent Claim 4	Copeland
		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.") Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Copeland 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 one such refernece. 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. 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.") 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

No.	'111 Patent Claim 4	Copeland
		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.") 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.") 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

No.	'111 Patent Claim 4	Copeland
		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.") 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.") 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.")

No.	'111 Patent Claim 4	Copeland
		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.") Swenson at Figure 1 (annotation added)  FIG. 1

No.	'111 Patent Claim 4	Copeland

No.	'111 Patent Claim 5	Copeland
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.	Copeland 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. 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. *See supra* at Claim 1. 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.") 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.") 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

No.	'111 Patent Claim 5	Copeland
		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.") 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.") 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: (1) both port numbers match and no port is marked as the "server" port, or (2) the port number previously marked as the "server" port matches, or (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).") 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

No.	'111 Patent Claim 5	Copeland
		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."") 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.") 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: (a). both port numbers match and no port is marked as the "server" port, or (b). the port number previously marked as the "server" port matches, or (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	Copeland
6	The method according to claim 5, further comprising storing the received packet or a portion thereof, by the controller, in a memory.	Copeland discloses the method according to claim 5, further comprising storing the received packet or a portion thereof, by the controller, in a memory. For example, Copeland discloses storing packet and sampled packet headers. *See supra* at Claim 5.

No.	'111 Patent Claim 6	Copeland
		Copeland at [0042] ("Turning to the figures, in which like numerals indicate like elements throughout the several figures, FIG. 1 provides an overview of a flow-based intrusion detection system or engine 155 in accordance with an exemplary embodiment of the present invention. The flow-based intru-sion detection system 155 monitors network computer com-munications. The network computer communications are routed via a known global computer network commonly known as the Internet 199. In accordance with an aspect of the invention, the intrusion detection engine 155 is incor-porated into a monitoring appliance 150, together with a database 160 that stores information utilized in the intrusion detection methodology.") Copeland at [0119] ("FIG. 5 illustrates the operation of the preferred flow-based engine 155. The engine stores data from its operations in a database 160, which in the disclosed embodi-ment comprises two data structures, one used to collect statistics on data flows (flow data structure 162) in progress, and another to accumulate data on the host computers (host data structure 166) involved in those flows. It has four main threads or processes that read and write these data structures to identify possible intruders, which are identified as high concern index hosts or high CI hosts. These threads are a sampled packet collector 505, packet classifier thread 510, a flow collector thread 520, and an alert manager thread 530. The threads also identify the client and server network applications that are being operating by the hosts that are observed participating in the flows observed (port profilmg).") Copeland at [0138] ("After the selection of a "Primary Thread," data reconciliation must be done. During the start of a flow, before any flow is designated as a "Primary Flow" no host statistics can be gathered. Therefore, important information, such as the number of packets and bytes sent, is stored in temporary storage that is associated with the flow data. When a flow is designated as a Primary Flow the data is scaled by a sample rate that is associated with the selected flow and added to the host data. Thereafter, the host data is updated as the packets are processed in order to provide a real-time view of the hosts for data associated with the Primary Flow. A yet further aspect of the present invention adds the host data at the termination of the flow. This aspect is less desirable, as a long duration flow would have no data representation in the host data structure until the flow has ended.") Copeland at [0153] ("As discussed in reference to FIG. 5, the processor 850 executes the packet classifier thread 510, the flow collector thread 520, and the alert manager thread 530,

No.	'111 Patent Claim 6	Copeland
		These threads interact with flow data structure 162 and the host data structure 166, as described. The data structures provide temporary storage of information. As discussed in reference to FIG. 5, a log file is maintained on the hard drive 840 for forensic analysis, if desired.")

No.	'111 Patent Claim 7	Copeland
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.	Copeland 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. For example, Copeland discloses sampled packet headers, to the intrusion detection engine on the monitoring appliance based on matching predetermined values associated with a concern index. Sending samples packet headers to the controller in response to satisfying the criterion and instruction is sending a portion of a packet because packets can be fragmented into smaller bit sizes, such as headers. *See supra* at Claim 5. Copeland at [0072] ("The conventional format of an sFlow datagram record 259 comprises a record header 262 that is followed by one or more packet samples. Each packet sample comprises an sFlow sample header 270 and packet data 290, wherein the packet data 290 comprises a sampled packet header 292, as shown in FIG. 2A. Also illustrated in FIG. 2A, the sFlow sample header can comprise data relating to the sFlow sample data type 272, the sample sequence number 274, the data sampling rate 276, the total number of packets observed 278, the number of packets dropped 280, the destination input port 282, the output port 284 and the data sampling type 286. On network devices 135 the sFlow agent will collect and transmit the sFlow record 259 on a regular basis, normally forwarding all packet samples within one second of collection.") 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.")

No.	'111 Patent Claim 7	Copeland
		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.") 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 8	Copeland
8[a]	The method according to claim 7, wherein the portion of the packet consists of multiple consecutive bytes, and	Copeland discloses the method according to claim 7, wherein the portion of the packet consists of multiple consecutive bytes. 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. *See supra* at Claim 7. 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.") 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

No.	'111 Patent Claim 8	Copeland
		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.") 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.")
8[b]	wherein the instruction comprises identification of the consecutive bytes in the packet.	Copeland discloses wherein the instruction comprises identification of the consecutive bytes in the packet. For example, Copeland discloses identifying the sampled packet headers comprised of fragmented packets of smaller byte sizes. 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.") 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.") 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

No.	'111 Patent Claim 8	Copeland
		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	Copeland
9	The method according to claim 5, further comprising responsive to receiving the packet, analyzing the packet, by the controller.	Copeland discloses the method according to claim 5, further comprising responsive to receiving the packet, analyzing the packet, by the controller. For example, Copeland discloses analyzing the packet by the monitoring appliance for potential detrimental activity, via the intrusion detection engine. *See supra* at Claim 5. 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.") 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

No.	'111 Patent Claim 9	Copeland
		value of a host exceeds a preset alarm value, an alert is issued and appropriate action can be taken.") Copeland at [0024] ("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.") 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 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.") 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.")

No.	'111 Patent Claim 9	Copeland
		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.") 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.) 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.") 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.")

No.	'111 Patent Claim 9	Copeland
		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.") 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.")

No.	'111 Patent Claim 12	Copeland
12	The method according to claim 9, wherein the analyzing comprises applying security or data analytic application.	Copeland discloses the method according to claim 9, wherein the analyzing comprises applying security or data analytic application. For example, Copeland discloses analysis by the intrusion detection engine of the monitoring appliance to include security analytics and determining suspicious activity. *See supra* at Claim 9. Copeland at [0022] ("This novel detection system does not require an updated database of signatures. Instead, the intrusion detec-tion system inspects all inbound and outbound activity and identifies suspicious patterns that denote non-normal flows and may indicate an attack. The computational simplicity of the technique allows for operation at much higher speeds than is possible with a signature-based system on compa-rable hardware. The ability of the system to operate with sampled data provides the ability to operate at much higher network speeds and cover a wide network area with a high degree of accuracy and with more detailed information than would otherwise be possible.")

No.	'111 Patent Claim 12	Copeland
		Copeland at [0026] ("This novel detection system does not require an updated database of signatures. Instead, the intrusion detec-tion system inspects all inbound and outbound activity and identifies suspicious patterns that denote non-normal flows and may indicate an attack. The computational simplicity of the technique allows for operation at much higher speeds than is possible with a signature-based system on compa-rable hardware. The ability of the system to operate with sampled data provides the ability to operate at much higher network speeds and cover a wide network area with a high degree of accuracy and with more detailed information than would otherwise be possible.") 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.") Copeland at [0143] ("For security monitoring, a single event multiplied by a sample rate should not be allowed to dominate a calculation. In the above example, a host sending a single SYN packet that is sampled at 1/10,000 would be inaccurately identified as generating a SYN flood if we sampled a single packet containing the SYN flag and multiplied it by 10,000. As such, multiple samples must be obtained for threshold based event alarms such as this to add statistical significance to the data. Common approaches for this comprise requiring an absolute minimum number of incidents via sampling (e.g., 5). A second approach would be based on an algorithm that determines a minimum number of samples that would be required based on the sample rate and the threshold value in order to maintain a good balance of sensitivity and accuracy.") 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

No.	'111 Patent Claim 12	Copeland
		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 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.")

No.	'111 Patent Claim 13	Copeland
13	The method according to claim 9, wherein the analyzing comprises applying security application that comprises firewall or intrusion detection functionality.	Copeland discloses the method according to claim 9, wherein the analyzing comprises applying security application that comprises firewall or intrusion detection functionality. 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. *See supra* at Claim 9. 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.") 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

No.	'111 Patent Claim 13	Copeland
		appliance 150 moni-tors the communications between the host server 130 and other hosts 120, 110 in the attempt to detect intrusion activity.") 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.") 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.") 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.") Copeland at Figure 5

No.	'111 Patent Claim 13	Copeland

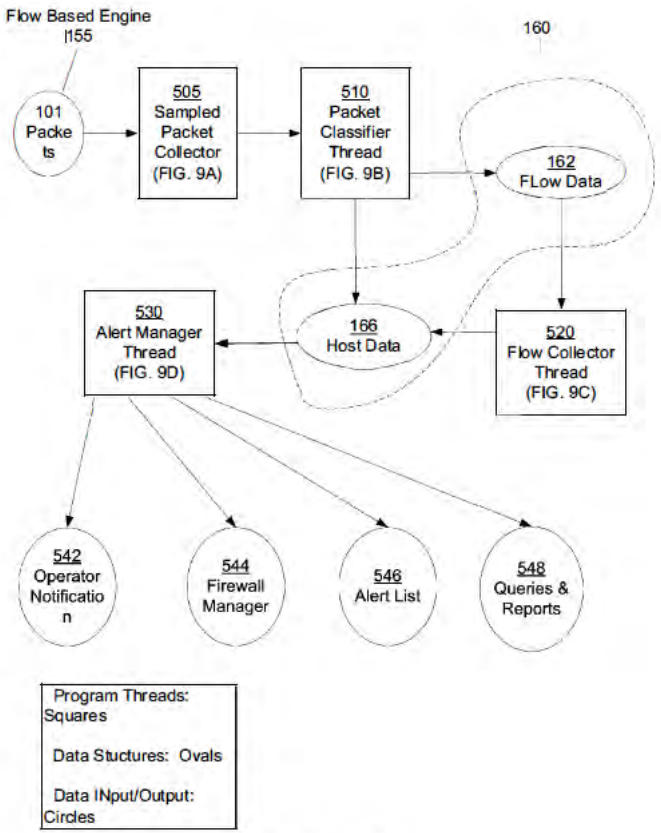


FIG. 5

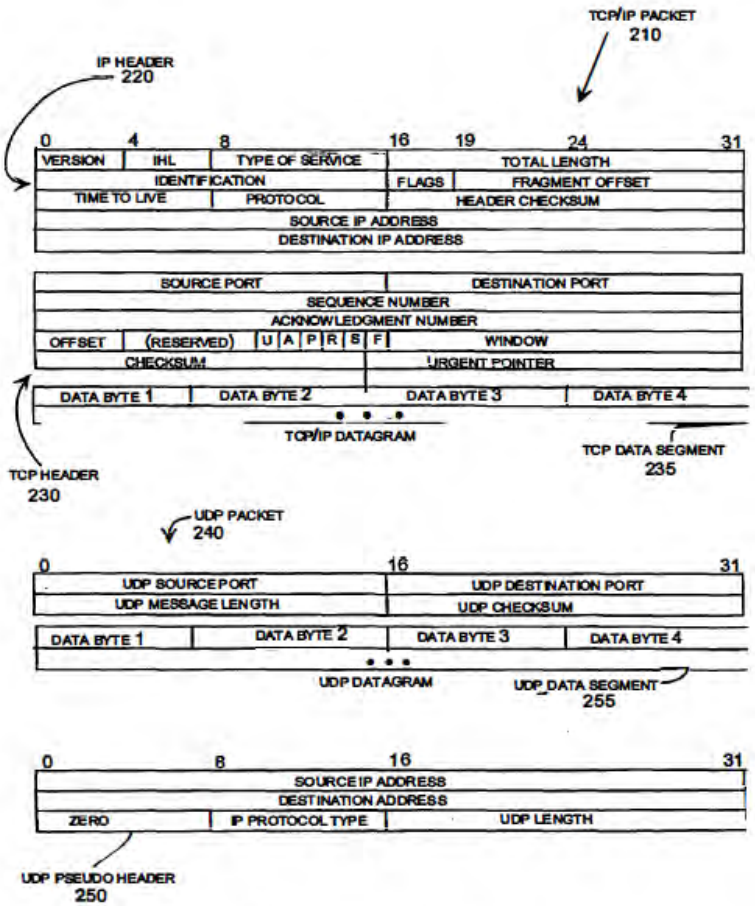
Copeland at [0149] ("The alert manager 530 also looks for hosts whose CI or traffic (byte rate) exceeds preset alarm thresholds and which have not been handled on previous runs. The new alarm conditions can cause immediate operator notification by an operator notification process 542. These conditions 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 appreciated that the present

No.	'111 Patent Claim 13	Copeland
		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.") 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.")

No.	'111 Patent Claim 14	Copeland
14	The method according to claim 9, wherein the analyzing comprises performing Deep Packet Inspection (DPI) or using a DPI engine on the packet.	Copeland discloses the method according to claim 9, wherein the analyzing comprises performing Deep Packet Inspection (DPI) or using a DPI engine on the packet. For example, Copeland discloses analysis by the intrusion detection engine on the monitoring device to include inspection of all inbound and outbound activity. A person of ordinary skill in the art would understand that such inspection of network traffic would include deep packet inspection. *See supra* at Claim 9. Copeland at [0022] ("This novel detection system does not require an updated database of signatures. Instead, the intrusion detec-tion system inspects all inbound and outbound activity and identifies suspicious patterns that denote non-normal flows and may indicate an attack. The computational simplicity of the technique allows for operation at much higher speeds than is possible with a signature-based system on compa-rable hardware. The ability of the system to operate with sampled data provides the ability to operate at much higher network

No.	'111 Patent Claim 14	Copeland
		speeds and cover a wide network area with a high degree of accuracy and with more detailed information than would otherwise be possible.") Copeland at [0026] ("This novel detection system does not require an updated database of signatures. Instead, the intrusion detec-tion system inspects all inbound and outbound activity and identifies suspicious patterns that denote non-normal flows and may indicate an attack. The computational simplicity of the technique allows for operation at much higher speeds than is possible with a signature-based system on compa-rable hardware. The ability of the system to operate with sampled data provides the ability to operate at much higher network speeds and cover a wide network area with a high degree of accuracy and with more detailed information than would otherwise be possible.")

No.	'111 Patent Claim 15	Copeland
15[a]	The method according to claim 9, wherein the packet comprises distinct header and payload fields, and	Copeland discloses the method according to claim 9, wherein the packet comprises distinct header and payload fields. For example, Copeland discloses packets with header fields and data segment fields. *See supra* at Claim 9. Copeland at Figure 2

No.	'111 Patent Claim 15	Copeland
		 FIG. 2 Copeland at [0024] ("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

No.	'111 Patent Claim 15	Copeland
		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.") Copeland at [0025] ("The present invention provides an accurate and reliable method for detecting network attacks through the use of sampled packet headers provided by a source such as defined in sFlow and is based in large part on "flows" as opposed to signatures or anomalies. By utilizing the host and flow information structures inherent with flow-based analy-sis and applying rules of statistical significance and analysis, the intrusion detection system can operate with sampled data such as provided by sFlow to provide hybrid solution that combines many of the benefits of a packet capture imple-mentation with the distributed nature of an IDS that operates on Netflow to provide an enhanced wide-area IDS solution.") 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.") Copeland at [0046] ("Each IP packet 101 includes a header that contains both the sender's Internet address and receiver's Internet address. The packets 101 are forwarded to the computer whose address is specified. Illustrated is a legitimate user/ client 110, host #1 (Hl), with an IP address of 208.60.239.19 and a server, host #2 (H2), with an IP address of 128.0.0.1.") Copeland at [0049] ("110.5.47.224. The intruder 120 also communicates by sending packets 101 via the Internet 199. As previously stated, the packets 101 contain the IP address of the origi-nator and destination to ensure proper routing. As shown, the stream of packets 101 sent

No.	'111 Patent Claim 15	Copeland
		by the intruder 120 can be interleaved with the packets 101 sent by other hosts. The packets 101 contain header information that enables the receiving host to reassemble the interleaved stream of pack-ets into the original messages as sent.") 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.") Copeland at [0074] ("Referring now to FIG. 2, and inasmuch as an understanding of Internet data packets is helpful for con-structing embodiments of the present invention, a descrip-tion of such packets, also called "datagrams", will next be provided as an aid to understanding. A packet or datagram 101 is a self-contained, independent entity or unit of data carrying sufficient information to be routed from a source to a destination computer without reliance on earlier exchanges between the source and destination computer. Packets 101 have a header and a data segment as illustrated by FIG. 2. The term "packet" in present-day parlance has generally replaced the term "datagram".") 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.") 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.")

No.	'111 Patent Claim 15	Copeland
		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.") 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 [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.")
15[b]	wherein the analyzing comprises checking part of, or whole of, the payload field.	Copeland discloses wherein the analyzing comprises checking part of, or whole of, the payload field. For example, Copeland discloses processing packet data sent to the monitoring appliance. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Copeland 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. 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

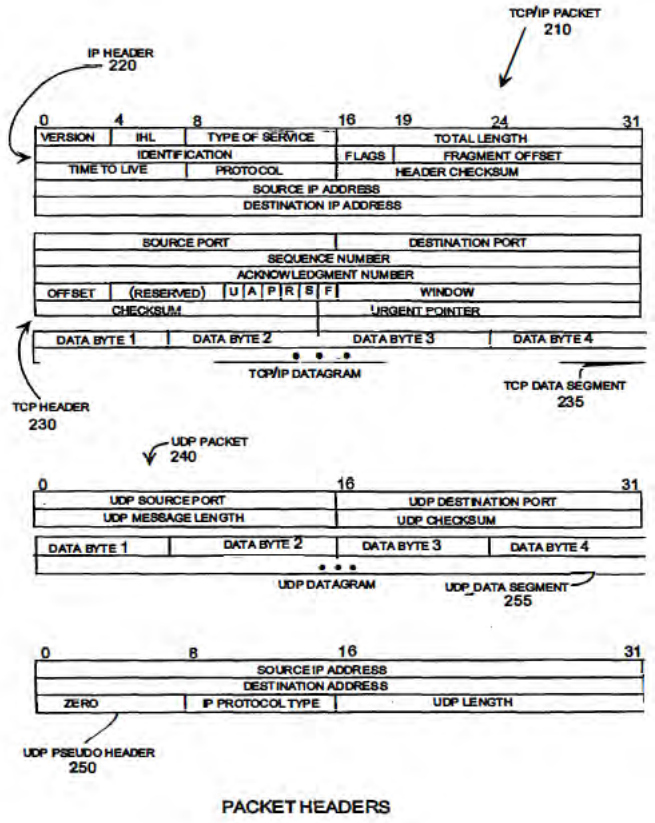
No.	'111 Patent Claim 15	Copeland
		IP addresses and assures that the packet is delivered to the right host. The transport layer protocol (TCP) header follows the Internet protocol header and specifies the port numbers for the associated service.") Copeland at [0120] ("The sampled packet headers sent from the sFlow agent are captured and processed by the sample packet collector 505 in order to create a "Packet Data" data structure 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 provides 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.") Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Copeland 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 of such a reference. 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. 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

No.	'111 Patent Claim 15	Copeland
		130 receives instructions from the network controller 140 based on the desired criteria for categorizing flows of interest for further inspection.") 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.") 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.") 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

No.	'111 Patent Claim 15	Copeland
		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.") 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.") 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

No.	'111 Patent Claim 15	Copeland
		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 16	Copeland
16[a]	The method according to claim 1, wherein the packet comprises distinct header and payload fields,	Copeland discloses the method according to claim 1, wherein the packet comprises distinct header and payload fields. *See supra* at Claim 1, 15[a].
16[b]	the header comprises one or more flag bits, and	Copeland discloses the header comprises one or more flag bits. For example, Copeland discloses packet headers with flag bits. Copeland at Figure 2

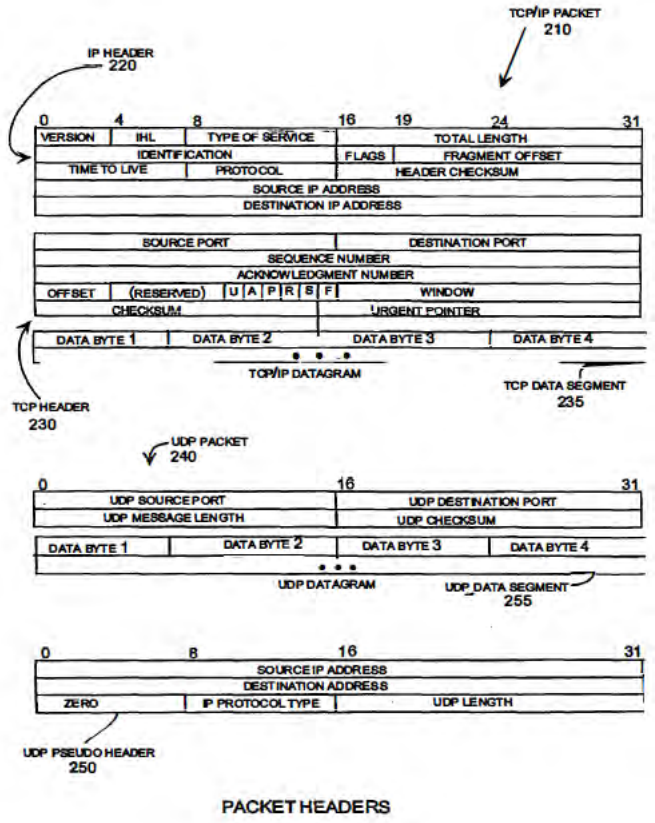
No.	'111 Patent Claim 16	Copeland
		 FIG. 2 Copeland at [0076] ("FIG. 2 illustrates an exemplary TCP/IP packet or datagram 210 and an exemplary UDP datagram 240. In a typical TCP/IP packet like 210, each packet typically includes a header portion comprising an IP header 220 and a TCP header 230, followed 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.")

No.	'111 Patent Claim 16	Copeland
		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.") 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.") 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 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.") 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

No.	'111 Patent Claim 16	Copeland
		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.") 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.")
16[c]	wherein the packet-applicable criterion is that one or more of the flag bits is set.	Copeland discloses wherein the packet-applicable criterion is that one or more of the flag bits is set. For example, Copeland discloses packet specific characteristics including flag bits that are set. 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 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

No.	’111 Patent Claim 16	Copeland
		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.”) 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.”) 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.”) 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).”) 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.”)

No.	'111 Patent Claim 17	Copeland
17[a]	The method according to claim 16, wherein the packet is an Transmission Control Protocol (TCP) packet, and	Copeland discloses the method according to claim 16, wherein the packet is an Transmission Control Protocol (TCP) packet. For example, Copeland discloses TCP/IP packets and packets sent in a TCP session. *See supra* at Claim 16. Copeland at Figure 2

No.	'111 Patent Claim 17	Copeland
		 FIG. 2 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.")

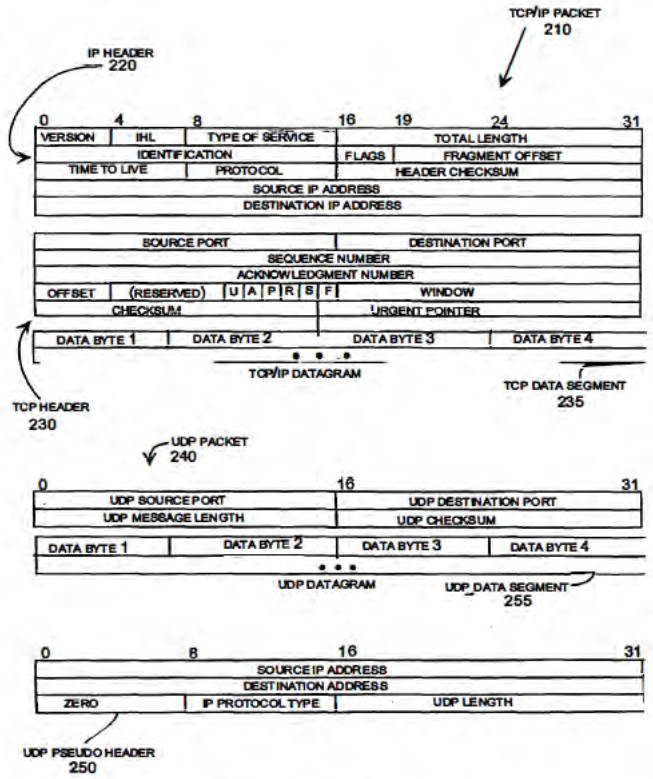
No.	'111 Patent Claim 17	Copeland
		Copeland at [0077] ("The header portion in the typical TCP/IP datagram 210 is 40 bytes including 20 bytes of IP header 220 information and 20 bytes of TCP header 230 information. The data portion or segment associated with the packet 210 follows the header information.") 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 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.") 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.") 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,

No.	'111 Patent Claim 17	Copeland
		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.")
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.	Copeland 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. 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 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 17	Copeland
		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.") 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.") 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.") 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.") 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.") 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.")

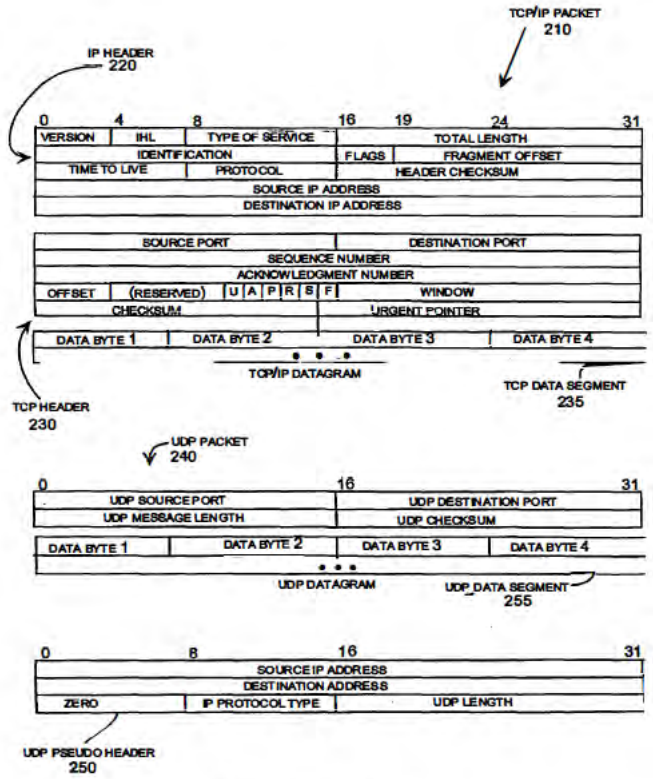
No.	'111 Patent Claim 17	Copeland
		Copeland at [0095] ("When Host 2 is ready to terminate the session, it sends its own packet with the FIN and ACK flags set. Hostl responds that it has received the final packet with an ACK packet providing to Host2 an acknowledgment number one greater than the sequence number provided in the FIN-ACK packet of Host2.")

No.	'111 Patent Claim 18	Copeland
18[a]	The method according to claim 1, wherein the packet comprises distinct header and payload fields,	Copeland discloses the method according to claim 1, wherein the packet comprises distinct header and payload fields. *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	Copeland discloses the header comprises at least the first and second entities addresses in the packet network. For example, Copeland discloses packet headers with source and destination IP addresses of the hosts. Copeland at Figure 2

No.	'111 Patent Claim 18	Copeland
		 PACKET HEADERS FIG. 2 Copeland at [0024] ("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

No.	'111 Patent Claim 18	Copeland
		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.") 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.") Copeland at [0043] ("The operating environment of the intrusion detec-tion system 155 is contemplated to have numerous hosts connected by the Internet 199, e.g. Host #1, Host #2, Host #3 (also referred to as Hl-H3 respectively). Hosts are any computers that have full two-way access to other computers on the Internet 199 and have their own unique IP address. For example Host #1 has an exemplary IP address of 208.60.239.19. The Internet 199 connects clients 110 with a host server 130 in known client/server relationship.") Copeland at [0045] ("It is known in the art to send communications between hosts via the Internet 199. The Internet Protocol (IP) is the method by which data is sent from one host computer to another on the Internet 199. Each host on the Internet 199 has an IP address that uniquely identifies it from all other computers. When data is transmitted, the message gets divided into packets 101. Packets 101 are discussed in more detail in reference to FIG. 2.") Copeland at [0046] ("Each IP packet 101 includes a header that contains both the sender's Internet address and receiver's Internet address. The packets 101 are forwarded to the computer whose address is specified. Illustrated is a legitimate user/ client 110, host #1 (Hl), with an IP address of 208.60.239.19 and a server, host #2 (H2), with an IP address of 128.0.0.1.") Copeland at [0049] ("110.5.47.224. The intruder 120 also communicates by sending packets 101 via the Internet 199. As previously stated, the packets 101 contain the IP address of the

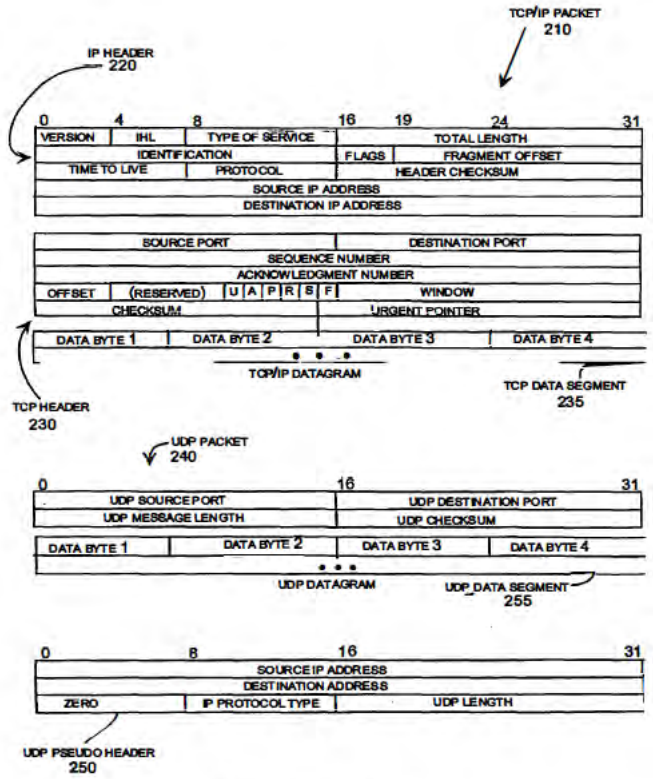
No.	'111 Patent Claim 18	Copeland
		origi-nator and destination to ensure proper routing. As shown, the stream of packets 101 sent by the intruder 120 can be interleaved with the packets 101 sent by other hosts. The packets 101 contain header information that enables the receiving host to reassemble the interleaved stream of pack-ets into the original messages as sent.")
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.	Copeland discloses wherein the packet-applicable criterion is that the first entity address, the second entity address, or both match a predetermined address or addresses. For example, Copeland discloses packet specific characteristics including source and destination IP addresses, that match predetermined packet and flow values. Copeland at Figure 2

No.	'111 Patent Claim 18	Copeland
		 FIG. 2 Copeland at [0024] ("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

No.	'111 Patent Claim 18	Copeland
		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.") 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.") Copeland at [0043] ("The operating environment of the intrusion detec-tion system 155 is contemplated to have numerous hosts connected by the Internet 199, e.g. Host #1, Host #2, Host #3 (also referred to as Hl-H3 respectively). Hosts are any computers that have full two-way access to other computers on the Internet 199 and have their own unique IP address. For example Host #1 has an exemplary IP address of 208.60.239.19. The Internet 199 connects clients 110 with a host server 130 in known client/server relationship.") Copeland at [0045] ("It is known in the art to send communications between hosts via the Internet 199. The Internet Protocol (IP) is the method by which data is sent from one host computer to another on the Internet 199. Each host on the Internet 199 has an IP address that uniquely identifies it from all other computers. When data is transmitted, the message gets divided into packets 101. Packets 101 are discussed in more detail in reference to FIG. 2.") Copeland at [0046] ("Each IP packet 101 includes a header that contains both the sender's Internet address and receiver's Internet address. The packets 101 are forwarded to the computer whose address is specified. Illustrated is a legitimate user/ client 110, host #1 (Hl), with an IP address of 208.60.239.19 and a server, host #2 (H2), with an IP address of 128.0.0.1.") Copeland at [0049] ("110.5.47.224. The intruder 120 also communicates by sending packets 101 via the Internet 199. As previously stated, the packets 101 contain the IP address of the

No.	'111 Patent Claim 18	Copeland
		origi-nator and destination to ensure proper routing. As shown, the stream of packets 101 sent by the intruder 120 can be interleaved with the packets 101 sent by other hosts. The packets 101 contain header information that enables the receiving host to reassemble the interleaved stream of pack-ets into the original messages as sent.")

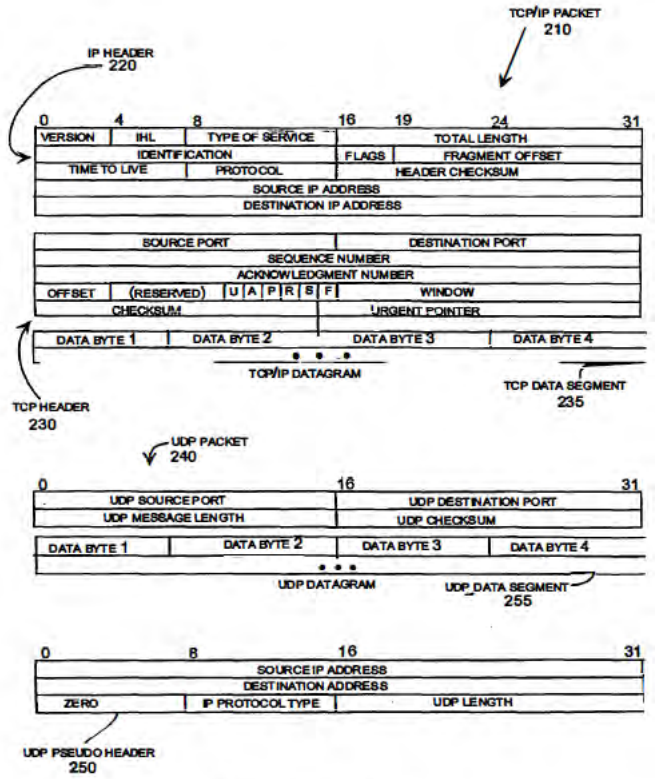
No.	'111 Patent Claim 19	Copeland
19	The method according to claim 18, wherein the addresses are Internet Protocol (IP) addresses.	Copeland discloses the method according to claim 18, wherein the addresses are Internet Protocol (IP) addresses. For example, Copeland discloses packet headers with source and destination IP addresses. *See supra* at Claim 18. Copeland at Figure 2

No.	'111 Patent Claim 19	Copeland
		 FIG. 2 Copeland at [0024] ("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

No.	'111 Patent Claim 19	Copeland
		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.") 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.") Copeland at [0043] ("The operating environment of the intrusion detec-tion system 155 is contemplated to have numerous hosts connected by the Internet 199, e.g. Host #1, Host #2, Host #3 (also referred to as Hl-H3 respectively). Hosts are any computers that have full two-way access to other computers on the Internet 199 and have their own unique IP address. For example Host #1 has an exemplary IP address of 208.60.239.19. The Internet 199 connects clients 110 with a host server 130 in known client/server relationship.") Copeland at [0045] ("It is known in the art to send communications between hosts via the Internet 199. The Internet Protocol (IP) is the method by which data is sent from one host computer to another on the Internet 199. Each host on the Internet 199 has an IP address that uniquely identifies it from all other computers. When data is transmitted, the message gets divided into packets 101. Packets 101 are discussed in more detail in reference to FIG. 2.") Copeland at [0046] ("Each IP packet 101 includes a header that contains both the sender's Internet address and receiver's Internet address. The packets 101 are forwarded to the computer whose address is specified. Illustrated is a legitimate user/ client 110, host #1 (Hl), with an IP address of 208.60.239.19 and a server, host #2 (H2), with an IP address of 128.0.0.1.") Copeland at [0049] ("110.5.47.224. The intruder 120 also communicates by sending packets 101 via the Internet 199. As previously stated, the packets 101 contain the IP address of the

No.	'111 Patent Claim 19	Copeland
		origi-nator and destination to ensure proper routing. As shown, the stream of packets 101 sent by the intruder 120 can be interleaved with the packets 101 sent by other hosts. The packets 101 contain header information that enables the receiving host to reassemble the interleaved stream of pack-ets into the original messages as sent.")

No.	'111 Patent Claim 20	Copeland
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	Copeland 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. *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 predetermined value or values.	Copeland 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. For example, Copeland discloses TCP packet specific characteristics that match predetermined packet and flow values. . Copeland at Figure 2

No.	'111 Patent Claim 20	Copeland
		 FIG. 2 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 20	Copeland
		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.") 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.") 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.") 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.") 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.") 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.")

No.	'111 Patent Claim 20	Copeland
		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.") 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.")

No.	'111 Patent Claim 21	Copeland
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.	Copeland 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. For example, Copeland discloses a packet network comprised of the Internet, which would be an Internet service provider. *See supra* at Claim 1. Copeland at [0042] ("Turning to the figures, in which like numerals indicate like elements throughout the several figures, FIG. 1 provides an overview of a flow-based intrusion detection system or engine 155 in accordance with an exemplary embodiment of the present invention. The flow-based intru-sion detection system 155 monitors network computer com-munications. The network computer communications are routed via a known global computer network commonly known as the Internet 199. In accordance with an aspect of the

No.	'111 Patent Claim 21	Copeland
		invention, the intrusion detection engine 155 is incor-porated into a monitoring appliance 150, together with a database 160 that stores information utilized in the intrusion detection methodology.") Copeland at [0043] ("The operating environment of the intrusion detec-tion system 155 is contemplated to have numerous hosts connected by the Internet 199, e.g. Host #1, Host #2, Host #3 (also referred to as Hl-H3 respectively). Hosts are any computers that have full two-way access to other computers on the Internet 199 and have their own unique IP address. For example Host #1 has an exemplary IP address of 208.60.239.19. The Internet 199 connects clients 110 with a host server 130 in known client/server relationship.") Copeland at [0045] ("It is known in the art to send communications between hosts via the Internet 199. The Internet Protocol (IP) is the method by which data is sent from one host computer to another on the Internet 199. Each host on the Internet 199 has an IP address that uniquely identifies it from all other computers. When data is transmitted, the message gets divided into packets 101. Packets 101 are discussed in more detail in reference to FIG. 2.")

No.	'111 Patent Claim 22	Copeland
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.	Copeland 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. For example, Copeland discloses client 110 and server 130 devices that communicate and exchange packets with each other. *See supra* at Claim 1. Copeland at [0044] ("In a typical configuration, some computers are referred to as "servers", while others are referred to as "clients." A server computer such as Host #2130 typically provides responses to requests from client computers and provides services, data, resources, and the like. While a client computer such as Host #1110 typically requests and utilizes the services, data, resources, and the like provided by the server.")

No.	'111 Patent Claim 22	Copeland
		Copeland at [0047] ("As shown, a client 110 communicates with a server 130 by sending packets 101 of data. A packet 101 is a unit of data that is routed between an origin and destination. As illustrated, messages are segmented into numerous packets 101 and routed via the Internet 199 to the receiving host. The receiving host reassembles the stream of packets 101 to recreate the original message, which is then handled by application programs running on the receiving computer system.") Copeland at [0050] ("Normal client/server (C/S) communication activity includes sending e-mails, Web traffic, file transfers, and the like. Communications via the Internet 199 need to be sent to a specific IP address and to a specific service contact port. A "port" is known to those skilled in the art as an arbitrarily assigned number to which a particular type of computing service is assigned in conventional Internet computer-to-computer communications, e.g. web traffic is conventionally on port 80, FTP traffic on ports 20 and 21, etc.")

No.	'111 Patent Claim 23	Copeland
23[a]	The method according to claim 22, wherein the server device comprises a web server, and	Copeland discloses the method according to claim 22, wherein the server device comprises a web server. For example, Copeland discloses server devices that communicate Web traffic via the Internet. *See supra* at Claim 22. Copeland at [0044] ("In a typical configuration, some computers are referred to as "servers", while others are referred to as "clients." A server computer such as Host #2130 typically provides responses to requests from client computers and provides services, data, resources, and the like. While a client computer such as Host #1110 typically requests and utilizes the services, data, resources, and the like provided by the server.") Copeland at [0050] ("Normal client/server (C/S) communication activity includes sending e-mails, Web traffic, file transfers, and the like. Communications via the Internet 199 need to be sent to a specific IP address and to a specific service contact port. A "port" is known to those skilled in the art as an arbitrarily assigned number to which a particular type of

No.	'111 Patent Claim 23	Copeland
		computing service is assigned in conventional Internet computer-to-computer communications, e.g. web traffic is conventionally on port 80, FTP traffic on ports 20 and 21, etc.") Copeland at [0052] ("A service port chart in FIG. I lists some common services that present day Internet-based computer systems may provide. Outgoing email typically utilizes the known Simple Mail Transfer Protocol (SMTP) that is implemented over the service contact port 25. For the Hypertext Transfer Protocol (HTTP) communications, Web browsers open an ephemeral high port number to initiate Web traffic that is sent to the host server port 80. File Transfer Protocol (FTP) control communications are sent to the server port 21, while FTP data transfer originates from port 20. The FINGER service utilizes service contact port 79, the domain name service (DNS) utilizes service contact port 53, and Telnet communications utilize service contact port 23. As illus-trated, common services are typically associated with spe-cific predetermined service contact ports.") Copeland at [0106] ("The first flow illustrated would be Web traffic (HTTP protocol) between the client at IP ADDRESS! and the server at IP ADDRESS0. The client Web browser opens a random ephemeral high port (51,132) as illustrated in the example. A high port is utilized because the low port numbers less than 1024 are pre-assigned for designated services. One of these designated services is port 80 for HTTP, which transfers displayable Web pages and related files in the known manner. The Web browser sends the request to the server's port 80. The server port responds by sending the requested Web page data in packets wherein the port number in the packets transmitted to the client sets the destination port to 51,132 of the client. All communications by clients utilizing HTTP is sent to port 80 of the server. One C/S flow would be the HTTP communications between port 51,132 of ADDRESS! and port 80 of ADDRESS0.")
23[b]	wherein the client device comprises a smartphone, a tablet computer, a personal computer, a laptop computer, or a	Copeland discloses wherein the client device comprises a smartphone, a tablet computer, a personal computer, a laptop computer, or a wearable computing device. For example, Copeland discloses client devices such as computers. Copeland at [0044] ("In a typical configuration, some computers are referred to as "servers", while others are referred to as "clients." A server computer such as Host #2130 typically

No.	'111 Patent Claim 23	Copeland
	wearable computing device.	provides responses to requests from client computers and provides services, data, resources, and the like. While a client computer such as Host #1110 typically requests and utilizes the services, data, resources, and the like provided by the server.") Copeland at [0050] ("Normal client/server (C/S) communication activity includes sending e-mails, Web traffic, file transfers, and the like. Communications via the Internet 199 need to be sent to a specific IP address and to a specific service contact port. A "port" is known to those skilled in the art as an arbitrarily assigned number to which a particular type of computing service is assigned in conventional Internet computer-to-computer communications, e.g. web traffic is conventionally on port 80, FTP traffic on ports 20 and 21, etc.") Copeland at [0139] ("Normal flows are those for which the correspond-ing statistics indicate a normal exchange of information between two hosts. The host that initiated the flow is considered the client (i.e. the computer that sent TCP SYN packets or sent an initial UDP packet). The other host is considered the server (i.e. the computer that sent TCP SYN-ACK packets or responded to a UDP packet). Some data is exchanged during a normal flow.")

No.	'111 Patent Claim 24	Copeland
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.	Copeland 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. For example, Copeland discloses communication between the network device, i.e., network node, and monitoring appliance using Netflow and sFlow standard protocols that are used in Software Defined Networks. *See supra* at Claim 22. Copeland at [0016] ("To overcome this limitation, a second embodiment is based on what is commonly referred to as Netflow data, wherein a router provides the Netflow data. A Netflow record contains information about each flow that passes through the router and provides a

No.	'111 Patent Claim 24	Copeland
		digest of the communi-cations showing hosts that were involved, services that were used, and how much data was exchanged. Through use of the router provided information, Netflow overcomes the limitations of requiring a separate packet capture IDS be installed at each location on the network. However, Netflow does not contain any packet level information, which limits the amount of detail available about the communications session. A possible source of data for network analysis is known as sFlow. As described in RFC-3176, sFlow is a technology designed for network monitoring based on packet sampling that is derived from work performed at the University of Geneva and CERN in 1991. sFlow operates by randomly sampling one out of every "n" data packets at a switch or router. The packet header along with forwarding tables and interface counters as well as additional packet information are delivered to an sFlow agent residing on the switch, wherein the sFlow agent forwards the captured information to an sFlow collector for processing. This technology is highly scalable, due to the fact that sample rates can be reduced on higher speed networks without causing a reduction in accuracy of traffic measurement.") Copeland at [0017] ("A possible source of data for network analysis is known as sFlow. As described in RFC-3176, sFlow is a technology designed for network monitoring based on packet sampling that is derived from work performed at the University of Geneva and CERN in 1991. sFlow operates by randomly sampling one out of every "n" data packets at a switch or router. The packet header along with forwarding tables and interface counters as well as additional packet information are delivered to an sFlow agent residing on the switch, wherein the sFlow agent forwards the captured information to an sFlow collector for processing. This technology is highly scalable, due to the fact that sample rates can be reduced on higher speed networks without causing a reduction in accuracy of traffic measurement.") 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.")

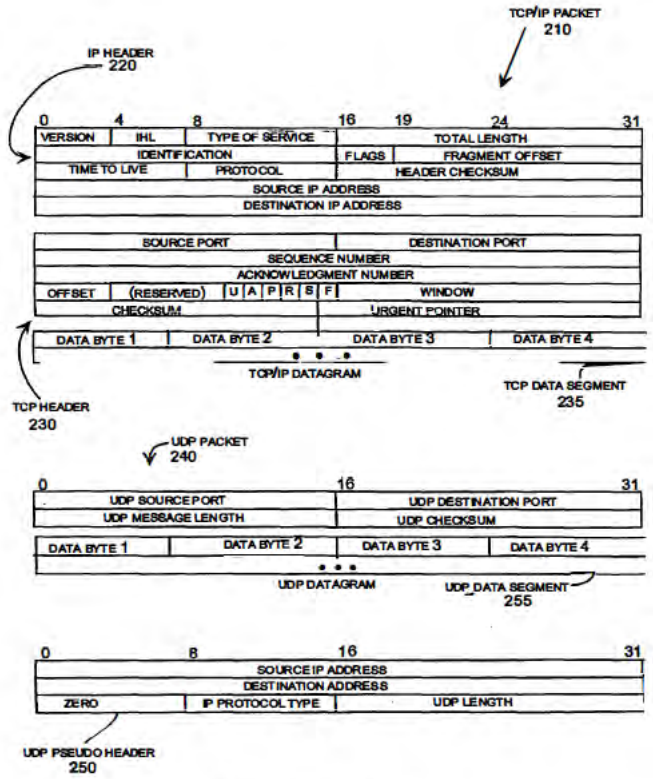
No.	'111 Patent Claim 24	Copeland
		Copeland at [0024] ("Generally speaking, the intrusion detection system analyzes network communication traffic for potential detri-mental activity. The system collects flow data from sampled packet headers between two hosts or Internet Protocol (IP) addresses. Collecting 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.") Copeland at [0025] ("The present invention provides an accurate and reliable method for detecting network attacks through the use of sampled packet headers provided by a source such as defined in sFlow and is based in large part on "flows" as opposed to signatures or anomalies. By utilizing the host and flow information structures inherent with flow-based analy-sis and applying rules of statistical significance and analysis, the intrusion detection system can operate with sampled data such as provided by sFlow to provide hybrid solution that combines many of the benefits of a packet capture imple-mentation with the distributed nature of an IDS that operates on Netflow to provide an enhanced wide-area IDS solution.") Copeland at [0052] ("A service port chart in FIG. I lists some common services that present day Internet-based computer systems may provide. Outgoing email typically utilizes the known Simple Mail Transfer Protocol (SMTP) that is implemented over the service contact port 25. For the Hypertext Transfer Protocol (HTTP) communications, Web browsers open an ephemeral high port number to initiate Web traffic that is sent to the host server port 80. File Transfer Protocol (FTP) control communications are sent to the server port 21, while FTP data transfer originates from port 20. The FINGER service utilizes service contact port 79, the domain name service (DNS) utilizes service contact port 53, and Telnet communications utilize service contact port 23. As illus-trated, common services are typically associated with spe-cific predetermined service contact ports.") Copeland at [0053] ("Also illustrated in FIG. 1 are four flows, Fl through F4, between by client host #1110 and service host #2130. Flow Fl is a file transfer utilizing the File Transfer

No.	'111 Patent Claim 24	Copeland
		Protocol (FTP). As shown, the file transfer (flow Fl) is delivered by a stream of packets 101 (Pl-P3) that will be reassembled by the receiving host 110.") 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.") Copeland at [0070] ("The sampled data format utilized within aspects of the present invention are based on the sFlow version 5.0 standard as published by sFlow.org and recognized by the Internet Engineering Task Force RFC-3176. As described in the RFC-3176, an sFlow capable device will randomly samples packet at a predetermined sampling rate that has been set by the user or device manufacturer. Addition, this sample rate can be controlled remotely, further, an IDS system could increase the scrutiny of the communications with a particular host or network segment by increasing the sample rate.") Copeland at [0104] ("The client is illustrated with an IP address of ADDRESS! while the server is illustrated with IP address ADDRESS0. As illustrated, three separate services-HTTP, SMTP, and FTP-are being invoked by the client. A Web browser application (not shown) running on the client machine utilizes the Hypertext Transfer Protocol (HTTP), an email application (also not shown) utilizes the Simple Mail Transfer Protocol (SMTP), and a file transfer application program (not shown) utilizes the File Transfer Protocol (FTP).") Copeland at [0105] ("Communications utilizing each of these protocols provide differing flow characteristics. Consequently, it is desirable to determine flows in which each of the services is analyzed separately. Therefore, as defined, the illustrated communications represent three distinct flows.") Copeland at [0106] ("The first flow illustrated would be Web traffic (HTTP protocol) between the client at IP ADDRESS! and the server at IP ADDRESS0. The client Web browser opens a random ephemeral high port (51,132) as illustrated in the example. A high port is utilized

No.	'111 Patent Claim 24	Copeland
		because the low port numbers less than 1024 are pre-assigned for designated services. One of these designated services is port 80 for HTTP, which transfers displayable Web pages and related files in the known manner. The Web browser sends the request to the server's port 80. The server port responds by sending the requested Web page data in packets wherein the port number in the packets transmitted to the client sets the destination port to 51,132 of the client. All communications by clients utilizing HTTP is sent to port 80 of the server. One C/S flow would be the HTTP communications between port 51,132 of ADDRESS! and port 80 of ADDRESS0.") Copeland at [0108] ("The next flow illustrated is email traffic between the client and server utilizing port 25. The client email application opens a random high ephemeral port, e.g. port 49,948 as illustrated in FIG. 4. The client's email applica-tion sends the email utilizing the Simple Mail Transfer Protocol (SMTP) to the server's port 25. Port 25 is conven-tionally designated for SMTP communications. A flow is terminated if no communications are delivered between the two IP addresses and the low port for 330 seconds. If the client sends another SMTP email packet or packets within 330 seconds of the end of the first email to the server, only one flow would exist.") Copeland at [0110] ("The File Transfer Protocol (FTP) is the simplest method to exchange files between hosts on the Internet. A client begins a session by sending a request to communicate to port 21 of designated server machine. The client also includes a second port number to be used when data is exchanged. The server initiates the exchange from its own port 20 (FTP DATA) to the port designated by the client, port 4993 as illustrated in FIG. 4. Although two ports on the server were utilized, port 20 and port 21, this unique case will be classified as one C/S flow. Preferably, embodiments of the present invention of a flow-based intrusion detection engine will have the ability to associate the FTP communi-cations on both ports as one flow.")

No.	'111 Patent Claim 27	Copeland
27	The method according to claim 1, wherein the network node comprises a router, a switch, or a bridge.	Copeland discloses the method according to claim 1, wherein the network node comprises a router, a switch, or a bridge. For example, Copeland discloses network devices, including routers and switches. *See supra* at Claim 1. 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 net-work traffic.")

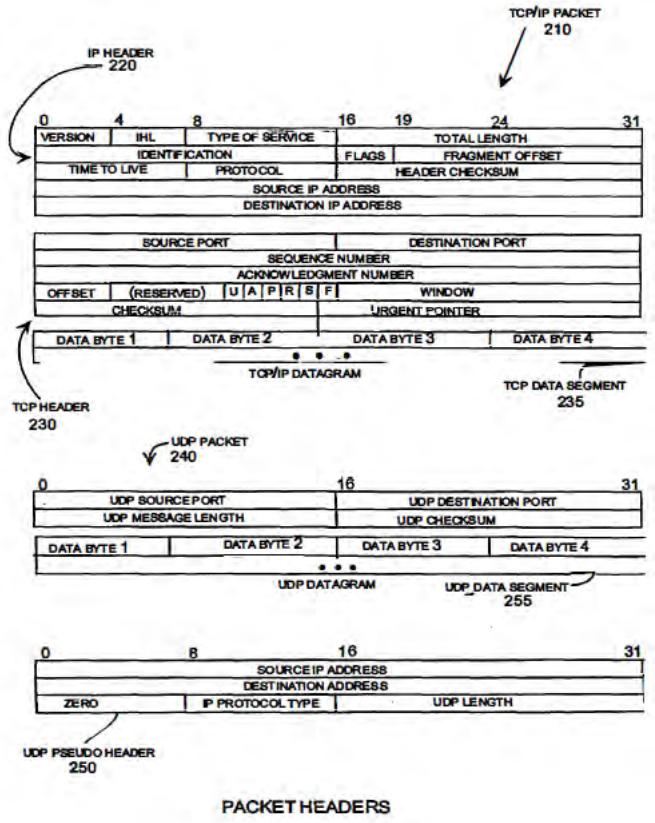
No.	'111 Patent Claim 28	Copeland
28	The method according to claim 1, wherein the packet network is an Internet Protocol (IP) network, and the packet is an IP packet.	Copeland discloses the method according to claim 1, wherein the packet network is an Internet Protocol (IP) network, and the packet is an IP packet. For example, Copeland discloses an Internet Protocol network and Internet Protocol packet. *See supra* at Claim 1. Copeland at Figure 2

No.	'111 Patent Claim 28	Copeland
		 FIG. 2 Copeland at [0024] ("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

No.	'111 Patent Claim 28	Copeland
		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.") 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.") Copeland at [0043] ("The operating environment of the intrusion detec-tion system 155 is contemplated to have numerous hosts connected by the Internet 199, e.g. Host #1, Host #2, Host #3 (also referred to as Hl-H3 respectively). Hosts are any computers that have full two-way access to other computers on the Internet 199 and have their own unique IP address. For example Host #1 has an exemplary IP address of 208.60.239.19. The Internet 199 connects clients 110 with a host server 130 in known client/server relationship.") Copeland at [0045] ("It is known in the art to send communications between hosts via the Internet 199. The Internet Protocol (IP) is the method by which data is sent from one host computer to another on the Internet 199. Each host on the Internet 199 has an IP address that uniquely identifies it from all other computers. When data is transmitted, the message gets divided into packets 101. Packets 101 are discussed in more detail in reference to FIG. 2.") Copeland at [0046] ("Each IP packet 101 includes a header that contains both the sender's Internet address and receiver's Internet address. The packets 101 are forwarded to the computer whose address is specified. Illustrated is a legitimate user/ client 110, host #1 (Hl), with an IP address of 208.60.239.19 and a server, host #2 (H2), with an IP address of 128.0.0.1.") Copeland at [0049] ("110.5.47.224. The intruder 120 also communicates by sending packets 101 via the Internet 199. As previously stated, the packets 101 contain the IP address of the

No.	'111 Patent Claim 28	Copeland
		origi-nator and destination to ensure proper routing. As shown, the stream of packets 101 sent by the intruder 120 can be interleaved with the packets 101 sent by other hosts. The packets 101 contain header information that enables the receiving host to reassemble the interleaved stream of pack-ets into the original messages as sent.")

No.	'111 Patent Claim 29	Copeland
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.	Copeland 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. For example, Copeland discloses a TCP network and TCP packet. *See supra* at Claim 28. Copeland at Figure 2

No.	'111 Patent Claim 29	Copeland
		 FIG. 2 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.")

No.	'111 Patent Claim 29	Copeland
		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.") 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 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.") 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.") 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

No.	'111 Patent Claim 29	Copeland
		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.")

No.	'111 Patent Claim 30	Copeland
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;	Copeland 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. For example, Copeland discloses the sending and receiving of additional packets within a communication flow from the client or server device. *See supra at* Claim 1, 1[c]. Copeland at [0109] ("For example, as shown in FIG. 4, if a second email packet originating from the ephemeral port 35,620 is sent within 330 seconds, only one flow would exist. If the second email packet arrives later than 330 seconds from the first sent email, it would be classified as another flow for analysis purposes. An analysis can determine if the flow is abnormal or not for SMTP communications.") Copeland at [0148] ("The packet classifier 510 thread collects informa-tion on network operations such as packets and bytes on a per-second, per-minute, and per-hour basis. This informa-tion is collected on all packets and on certain categories of packets such as TCP and UDP and subsets of these based on port number. Histograms of packet size and TCP or UDP port numbers are also collected. The alert manager thread 530 writes the updated data to various output files for use by the user interface, or for later off-line analysis.") Copeland at [0155] ("In FIG. 9A, the sampled packet collector begins with step 904. In step 904, the thread determines if a new sample datagram is available for processing. If a new

No.	’111 Patent Claim 30	Copeland
		sample is not available, the "No" branch of step 904 is followed to step 904, in which the thread awaits a new sample datagram. If a new sample datagram is available, the "Yes" branch of step 904 is followed to step 906, in which the thread extracts a single packet header and the associated information asso-ciated with sampled packet and places this information in a packet buffer. Subsequently, in step 908 the sample data-gram is tested to determine if another sample packet is available for processing. If another packet is available, the "Yes" branch of step 908 is followed to step 906 for additional processing, if no more packets are available in the datagram, then the "No" branch of step 908 is followed to step 904.”)
30[b]	checking, by the network node, if any one of the one or more additional packets satisfies the criterion;	Copeland discloses checking, by the network node, if any one of the one or more additional packets satisfies the criterion. *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	Copeland 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. *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.	Copeland 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. *See supra at* Claim 1[f], 30[a].

No.	'111 Patent Claim 31	Copeland
31[a]	The method according to claim 1, wherein the packet network is a Software Defined Network (SDN),	Copeland discloses the method according to claim 1, wherein the packet network is a Software Defined Network (SDN). For example, Copeland discloses a packet network using Netflow and sFlow standards that are used in Software Defined Networks. *See supra* at Claim 1. Copeland at [0016] ("To overcome this limitation, a second embodiment is based on what is commonly referred to as Netflow data, wherein a router provides the Netflow data. A Netflow record contains information about each flow that passes through the router and provides a digest of the communi-cations showing hosts that were involved, services that were used, and how much data was exchanged. Through use of the router provided information, Netflow overcomes the limitations of requiring a separate packet capture IDS be installed at each location on the network. However, Netflow does not contain any packet level information, which limits the amount of detail available about the communications session. A possible source of data for network analysis is known as sFlow. As described in RFC-3176, sFlow is a technology designed for network monitoring based on packet sampling that is derived from work performed at the University of Geneva and CERN in 1991. sFlow operates by randomly sampling one out of every "n" data packets at a switch or router. The packet header along with forwarding tables and interface counters as well as additional packet information are delivered to an sFlow agent residing on the switch, wherein the sFlow agent forwards the captured information to an sFlow collector for processing. This technology is highly scalable, due to the fact that sample rates can be reduced on higher speed networks without causing a reduction in accuracy of traffic measurement.") Copeland at [0017] ("A possible source of data for network analysis is known as sFlow. As described in RFC-3176, sFlow is a technology designed for network monitoring based on packet sampling that is derived from work performed at the University of Geneva and CERN in 1991. sFlow operates by randomly sampling one out of every "n" data packets at a switch or router. The packet header along with forwarding tables and interface counters as well as additional packet information are delivered to an sFlow agent residing on the switch, wherein the sFlow agent forwards the captured information to an sFlow collector for processing. This

No.	'111 Patent Claim 31	Copeland
		technology is highly scalable, due to the fact that sample rates can be reduced on higher speed networks without causing a reduction in accuracy of traffic measurement.") 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.") Copeland at [0025] ("The present invention provides an accurate and reliable method for detecting network attacks through the use of sampled packet headers provided by a source such as defined in sFlow and is based in large part on "flows" as opposed to signatures or anomalies. By utilizing the host and flow information structures inherent with flow-based analy-sis and applying rules of statistical significance and analysis, the intrusion detection system can operate with sampled data such as provided by sFlow to provide hybrid solution that combines many of the benefits of a packet capture imple-mentation with the distributed nature of an IDS that operates on Netflow to provide an enhanced wide-area IDS solution.") 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.") Copeland at [0070] ("The sampled data format utilized within aspects of the present invention are based on the sFlow version 5.0 standard as published by sFlow.org and recognized by the Internet Engineering Task Force RFC-3176. As described in the RFC-3176, an sFlow capable device will randomly samples packet at a predetermined sampling rate that has been set by

No.	'111 Patent Claim 31	Copeland
		the user or device manufacturer. Addition, this sample rate can be controlled remotely, further, an IDS system could increase the scrutiny of the communications with a particular host or network segment by increasing the sample rate.")
31[b]	the packet is routed as part of a data plane and	Copeland discloses the packet is routed as part of a data plane. For example, Copeland discloses routing packets between hosts via a global computer network, for example, the Internet. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Copeland 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. Copeland at [0042] ("Turning to the figures, in which like numerals indicate like elements throughout the several figures, FIG. 1 provides an overview of a flow-based intrusion detection system or engine 155 in accordance with an exemplary embodiment of the present invention. The flow-based intru-sion detection system 155 monitors network computer com-munications. The network computer communications are routed via a known global computer network commonly known as the Internet 199. In accordance with an aspect of the invention, the intrusion detection engine 155 is incor-porated into a monitoring appliance 150, together with a database 160 that stores information utilized in the intrusion detection methodology.") Copeland at [0047] ("As shown, a client 110 communicates with a server 130 by sending packets 101 of data. A packet 101 is a unit of data that is routed between an origin and destination. As illustrated, messages are segmented into numerous packets 101 and routed via the Internet 199 to the receiving host. The receiving host reassembles the stream of packets 101 to recreate the original message, which is then handled by application programs running on the receiving computer system.")

No.	'111 Patent Claim 31	Copeland
		Copeland at [0050] ("Normal client/server (C/S) communication activity includes sending e-mails, Web traffic, file transfers, and the like. Communications via the Internet 199 need to be sent to a specific IP address and to a specific service contact port. A "port" is known to those skilled in the art as an arbitrarily assigned number to which a particular type of computing service is assigned in conventional Internet computer-to-computer communications, e.g. web traffic is conventionally on port 80, FTP traffic on ports 20 and 21, etc.") Copeland at [0074] ("Referring now to FIG. 2, and inasmuch as an understanding of Internet data packets is helpful for con-structing embodiments of the present invention, a descrip-tion of such packets, also called "datagrams", will next be provided as an aid to understanding. A packet or datagram 101 is a self-contained, independent entity or unit of data carrying sufficient information to be routed from a source to a destination computer without reliance on earlier exchanges between the source and destination computer. Packets 101 have a header and a data segment as illustrated by FIG. 2. The term "packet" in present-day parlance has generally replaced the term "datagram".") Copeland at [0075] ("Restated, a packet 101 is the unit of data that is routed between an origin and destination on a packet-switched network such as the Internet 199. A packet-switch-ing scheme is an efficient method of handling transmissions on a connectionless network. However, connection-oriented protocols can be utilized to create a session. A session is a series of interactions between two communication end points that occur during the span of a single connection. A detailed discussion of a TCP/IP session is described in reference to FIG. 3. However, a host can send a message without establishing a connection with the recipient. That is, the host simply sends a packet 101 onto the network 199 with the destination address and hopes that it arrives.") Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Copeland 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. For example, Kempf discloses routing packets on a data plane using a control protocol

No.	'111 Patent Claim 31	Copeland
		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).") 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

No.	'111 Patent Claim 31	Copeland
		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.") 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") 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.") 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

No.	'111 Patent Claim 31	Copeland
		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.) 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.") 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.")

No.	'111 Patent Claim 31	Copeland
		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.") 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.") 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: Write-Metadata (GTP-TEID, 0x FFFFFFFF) Apply-Actions (Set-Output-Port LOCAL_GTP _U_ENCAP)")

No.	'111 Patent Claim 31	Copeland
		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.") As another example, Chua discloses forwarding packets over a data plane to various network destinations. 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.") 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.")

No.	'111 Patent Claim 31	Copeland
		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.") 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.") 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.")

No.	'111 Patent Claim 31	Copeland
31[c]	the network node communication with the controller serves as a control plane.	Copeland discloses the network node communication with the controller serves as a control. For example, Copeland discloses distributed flow communication between the network device and monitoring. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Copeland 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. 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.") Copeland at [0043] ("The operating environment of the intrusion detec-tion system 155 is contemplated to have numerous hosts connected by the Internet 199, e.g. Host #1, Host #2, Host #3 (also referred to as Hl-H3 respectively). Hosts are any computers that have full two-way access to other computers on the Internet 199 and have their own unique IP address. For example Host #1 has an exemplary IP address of 208.60.239.19. The Internet 199 connects clients 110 with a host server 130 in known client/server relationship.") 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.")

No.	'111 Patent Claim 31	Copeland
		Copeland at [0113] ("The computer programs or threads are executed on a computer system 800 constructed as described in reference to FIG. 8, which illustrates a suitable exemplary computer system that may be utilized to construct a monitoring appliance 150 including an intrusion detection engine 155, or a separately implemented intrusion detection engine. Although the described embodiments are generally described in reference to an Internet-accessible computer system that is dedicated to implementing the engine 155, those of ordinary skill in the art will recognize that the present invention can be implemented in computer program code that can execute in conjunction with other program modules in various types of general purpose, special pur-pose, or dedicated computers. Accordingly, it will be under-stood that the terms "computer,""operating system," and "application program" include all types of computers and the program modules designed to be implemented by the computers.") Copeland at [0150] ("A preferred hardware configuration 800 of an embodiment that executes the functions of the above-de-scribed 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 net-work traffic.) Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Copeland 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. For example, Kempf discloses communication between network elements and an OpenFlow controller over a control plane. 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

No.	'111 Patent Claim 31	Copeland
		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).") 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

No.	'111 Patent Claim 31	Copeland
		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.") 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") 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.") 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

No.	'111 Patent Claim 31	Copeland
		web site hosted by Stanford Uni-versity. As used herein, an "OpenFlow switch" refers to a network element implementing the OpenFlow protocol.) 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.") 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.") 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

No.	'111 Patent Claim 31	Copeland
		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.") 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.") 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: Write-Metadata (GTP-TEID, 0x FFFFFFFF) Apply-Actions (Set-Output-Port LOCAL_GTP _U_ENCAP)") 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

No.	'111 Patent Claim 31	Copeland
		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.") 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. 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.") 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.") 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

No.	'111 Patent Claim 31	Copeland
		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.") 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).")

Chart for U.S. Patent 10,652,111 ("the '111 Patent")
U.S. Publication No. 2011/0310901 A1 to Uchida et al. ("Uchida")

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 Uchida meets each element of those claims, and/or (2) 35 U.S.C. § 103 because Uchida 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 Uchida titled "Packet Forwarding Apparatus, Communication Ssytem, Process Rule Update Method, And Program", which was filed on July 26, 2011, and published on December 22, 2011.

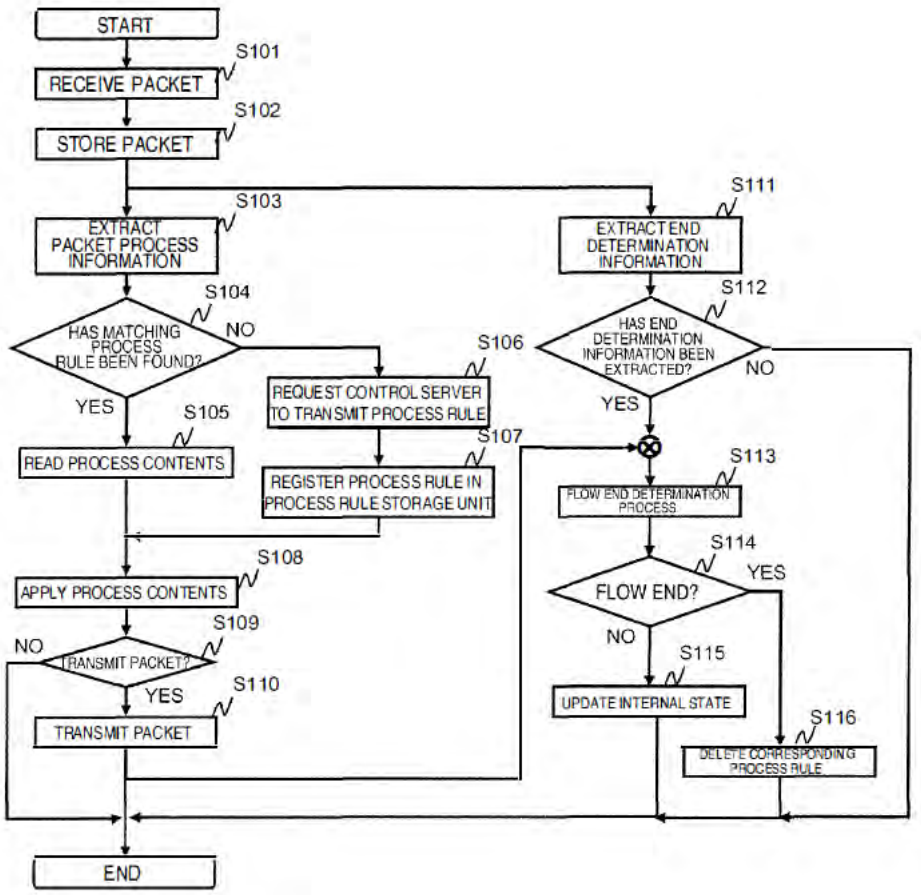
Motivations to combine the disclosures in Uchida with disclosures in other publications known in the art, as explained in this chart, include at least the similarity in subject matter between the references to the extent they concern methods relating to routing certain network traffic to entities for 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.

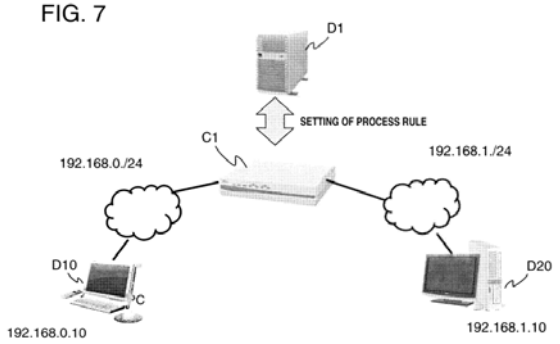
These invalidity contentions are based on Defendant's present understanding of the Asserted Claims, and Orckit's apparent construction of the claims in its November 3, 2022 and January 19, 2023 Preliminary Infringement Disclosures ("Orckit's Infringement Disclosures"), which is deficient at least insofar 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	Uchida
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:	Uchida 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. For example, Uchida discloses a packet forwarding apparatus C1 that relays TCP communication between a computer (client) D10 and a server D20 that use network configurations. The control apparatus (controller) D1 is connected to the packet forwarding apparatus C1 via a dedicated channel and manages connection between the two networks. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. Uchida at [0002] ("The present invention relates to a packet forwarding apparatus, a communication system, a process rule update method, and a computer program. In particular, it relates to a packet forwarding apparatus, a communication system, a process rule update method, and a computer program for processing a received packet in accordance with a process rule matching the received packet.") Uchida at [0104] ("Hereinafter, the description will be made, assuming that a packet forwarding apparatus C1 according to the present invention relays TCP communication between a computer (client) D10 and a server D20 that use network configurations illustrated in FIG. 7. In the example of FIG. 7, the computer D10 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 (controller) D1 is connected to the packet forwarding apparatus C1 via a dedicated channel and manages connection between the two networks. In the following description, the control apparatus (controller) D1 controls the packet forwarding apparatus C1 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 C17 monitors a protocol stack, including: fields in which the TCP is determined; and the FIN flag in the TCP header.")

No.	'111 Patent Claim 1	Uchida
		Uchida at [0109] ("Next, a process rule corresponding to the received packet is searched for. The packet process information extraction unit C12 transmits the extracted information (representing that the ingress port is 1; the source address is 192.168.0.10; the destination address is 192.168.1.10; and the protocol is TCP (the type is 0x0006)) to a process rule search unit C14. The process rule search unit C14 refers to the process rule storage unit C13. Since no process rule matching the packet exists, the process rule search unit C14 queries the control apparatus (controller) D1 (steps S104 and S106 in FIG. 3). The control apparatus (controller) D1 creates a process rule including process contents representing modification of the source IP and forward from a specified transport port and transmits the process rule to the packet forwarding apparatus C1. The process rule search unit C14 registers the process rule received from the control apparatus (controller) D1 as a new entry in the process rule storage unit C13 (step S107 in FIG. 3). At this stage, as illustrated in FIG. 9, a new process rule is added to the process rule storage unit C13.") Uchida at Figure 3

No.	'111 Patent Claim 1	Uchida
		 Uchida at Figure 7

No.	'111 Patent Claim 1	Uchida
		FIG. 7 
1[a]	sending, by the controller to the network node over the packet network, an instruction and a packet-applicable criterion;	Uchida discloses sending, by the controller to the network node over the packet network, an instruction and a packet-applicable criterion. For example, Uchida discloses the process rule search unit C14 transmits the received packet or information extracted from the received packet to a predetermined control apparatus (controller; see FIG. 18) to request creation and transmission of a process rule. Uchida at [0008] ("The following analyses are made by the present inventor. According to the above Non-Patent Documents 1 and 2, if a flow entry (hereinafter, information that is equiva-lent to the flow entry in Non-Patent Documents 1 and 2 and that the packet forwarding apparatus refers to when determin-ing a process executed on a received packet will be referred to as a process rule) is set by the controller at the start of com-munication and if a corresponding packet is not received during a period indicated by a separately set expiration time (the duration time in FIG. 19), the corresponding entry is deleted (see "Idle_Timeout and Hard_Timeout" on page 24 in Non-Patent Document 2).") Uchida at [0014] ("According to a third aspect of the present invention, there is provided a process rule update method causing a packet forwarding apparatus comprising: a process rule stor-age unit that stores process rules defining a process content for each flow; and a packet process unit that executes and processing a received packet by using a process rule matching the received packet among the process rules to execute: extracting information for checking a flow end from the received packet; and deleting, when a flow end is checked based on the

No.	'111 Patent Claim 1	Uchida
		extracted information, a process rule corre-sponding to the flow from the process rule storage unit. The present method is connected to a certain machine referred to as a packet forwarding apparatus comprising a process rule storage unit and a packet process unit.") Uchida at [0070] ("The process rule search unit C14 determines actions (process contents) to be applied to the received packet, based on the information transmitted from the packet process information extraction unit C12 and outputs the actions to the packet process unit C15. More specifically, the process rule search unit C14 searches the process rule storage unit C13 for a process rule matching the information extracted from the received packet transmitted from the packet process information extraction unit C12 and outputs the process rule to the packet process unit C15. If a process rule matching the received packet exists, the process rule search unit C14 extracts actions (process contents) defined in the action field of the process rule and outputs the actions to the packet process unit C15. If no process rule matching the received packet is found, the process rule search unit C14 transmits the received packet or information extracted from the received packet to a predetermined control apparatus (controller; see FIG. 18) to request creation and transmission of a process rule. Upon receiving a process rule from the predetermined control apparatus (controller; see FIG. 18), the process rule search unit C14 registers the received process rule in the process rule storage unit C13. Next, the process rule search unit C14 extracts actions (process contents) from the process rule and outputs the actions to the packet process unit C15.") Uchida at [0109] ("Next, a process rule corresponding to the received packet is searched for. The packet process information extraction unit C12 transmits the extracted information (representing that the ingress port is 1; the source address is 192.168.0.10; the destination address is 192.168.1.10; and the protocol is TCP (the type is 0x0006)) to a process rule search unit C14. The process rule search unit C14 refers to the process rule storage unit C13. Since no process rule matching the packet exists, the process rule search unit C14 queries the control apparatus (controller) D1 (steps S104 and S106 in FIG. 3). The control apparatus (controller) D1 creates a process rule including process contents representing modification of the source IP and forward from a specified transport port and transmits the process rule to the packet forwarding apparatus C1. The process rule search unit C14 registers the process rule received from the control apparatus (controller) D1 as a new entry in the process rule storage unit C13

No.	'111 Patent Claim 1	Uchida
		(step S107 in FIG. 3). At this stage, as illustrated in FIG. 9, a new process rule is added to the process rule storage unit C13.")
1[b]	receiving, by the network node from the controller, the instruction and the criterion;	Uchida discloses receiving, by the network node from the controller, the 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;	Uchida discloses receiving, by the network node from the first entity over the packet network, a packet addressed to the second entity. For example, Uchida discloses the computer D10 starts communication with the server D20. The packet reception unit C10 receives and stores this first packet in the packet storage unit C11]"). 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. Uchida at [0059] ("Next, a first exemplary embodiment of the present invention will be described in detail with reference to the drawings. First, terms will be defined. A "flow" signifies communication from a certain source to destination. Namely, a "flow" signifies a group of forwarded packets or the for-warding itself. Each of the "source" and "destination" does not necessarily signify a single physical or logical address. Namely, communication from a certain subnet to a different subnet may be collectively deemed as a single flow. Thus, a flow may be a collection of more detailed flows, one example of which corresponds to a flow entry in which a plurality of flows are unified by describing part of destination addresses, source addresses, or the like with wildcards, as in the case of OpenFlow in Non-Patent Documents 1 and 2.") Uchida at [0070] ("The process rule search unit C14 determines actions (process contents) to be applied to the received packet, based on the information transmitted from the packet process infor-mation extraction unit C12 and outputs the actions to the packet process unit C15. More specifically, the process rule search unit C14 searches the process rule storage unit C13 for a process rule matching the information extracted from the received packet

No.	'111 Patent Claim 1	Uchida
		transmitted from the packet process informa-tion extraction unit C12 and outputs the process rule to the packet process unit C15. If a process rule matching the received packet exists, the process rule search unit C14 extracts actions (process contents) defined in the action field of the process rule and outputs the actions to the packet process unit C15. If no process rule matching the received packet is found, the process rule search unit C14 transmits the received packet or information extracted from the received packet to a predetermined control apparatus (controller; see FIG. 18) to request creation and transmission of a process rule. Upon receiving a process rule from the predetermined control apparatus (controller; see FIG. 18), the process rule search unit C14 registers the received process rule in the process rule storage unit C13. Next, the process rule search unit C14 extracts actions (process contents) from the process rule and outputs the actions to the packet process unit C15.") Uchida at [0071] ("The first process rule in the flow table in FIG. 20 is applied to packets indicating that the physical port number of the packet forwarding apparatus is 1; the source network is 192.168.0./24; the protocol type is Ox0006 (TCP); the desti-nation network is 192.168.1./24; and the TCP destination port number is 80. Accordingly, the actions (process contents) of changing the IP source address of the packet to 192.168.1/24 and transmitting the packet from physical port number 2 of the forwarding apparatus are executed. Similarly, the second process rule in FIG. 20 is applied to packets indicating a destination port number other than HTTP port numbers. Accordingly, the actions (process contents) of changing the IP source address of the packet to 192.168.1/24 and transmit-ting the packet from physical port number 3 of the forwarding apparatus are executed.") Uchida at [0106] ("First, the computer D10 starts communication with the server D20. For an initial establishment of communication, a packet (SYN) is inputted to the packet forwarding apparatus C1 (start of ACTIVE OPEN through SYN forwarding in FIG. 6). The packet reception unit C10 receives and stores this first packet in the packet storage unit C11 (steps S101 to S102 in FIG. 3).") 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

No.	'111 Patent Claim 1	Uchida
		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.") Uchida at [0109] ("Next, a process rule corresponding to the received packet is searched for. The packet process information extraction unit C12 transmits the extracted information (representing that the ingress port is 1; the source address is 192.168.0.10; the destination address is 192.168.1.10; and the protocol is TCP (the type is 0x0006)) to a process rule search unit C14. The process rule search unit C14 refers to the process rule storage unit C13. Since no process rule matching the packet exists, the process rule search unit C14 queries the control apparatus (controller) D1 (steps S104 and S106 in FIG. 3). The control apparatus (controller) D1 creates a process rule including process contents representing modification of the source IP and forward from a specified transport port and transmits the process rule to the packet forwarding apparatus C1. The process rule search unit C14 registers the process rule received from the control apparatus (controller) D1 as a new entry in the process rule storage unit C13 (step S107 in FIG. 3). At this stage, as illustrated in FIG. 9, a new process rule is added to the process rule storage unit C13.")
1[d]	checking, by the network node, if the packet satisfies the criterion;	Uchida discloses checking, by the network node, if the packet satisfies the criterion. For example, Uchida discloses a packet process unit that executes processing a received packet by using a process rule matching the received packet. Uchida at [0012] ("According to a first aspect of the present invention, there is provided a packet forwarding apparatus comprising: a process rule storage unit that stores process rules defining a process content for each flow; a packet process unit that executes processing a received packet by using a process rule matching the received packet among the process rules and; an end determination information extraction unit that extracts information for checking a flow end from the received packet; and a flow end check unit that deletes, when a flow end is checked based on the extracted information, a process rule corresponding to the flow.") Uchida at [0014] ("According to a third aspect of the present invention, there is provided a process rule update method causing a packet forwarding apparatus comprising: a process rule stor-age unit that stores process rules defining a process content for each flow; and a packet

No.	'111 Patent Claim 1	Uchida
		process unit that executes and processing a received packet by using a process rule matching the received packet among the process rules to execute: extracting information for checking a flow received packet; and deleting, when a flow end is checked based on the extracted information, a process rule corre-sponding to the flow from the process rule storage unit. The present method is connected to a certain machine referred to as a packet forwarding apparatus comprising a process rule storage unit and a packet process unit.") Uchida at [0015] ("According to a fourth aspect of the present inven-tion, there is provided a program, causing a computer forming a packet forwarding apparatus comprising: a process rule storage unit that stores process rules defining a process con-tent for each flow; and a packet process unit that executes processing a received packet by using a process rule matching a received packet among the process rules to execute the processes of: extracting information for determining a flow end from the received packet; and deleting, when a flow end is determined based on the extracted information, a process rule corresponding to the flow from the process rule storage unit. The program can be stored in a computer-readable recording medium. Namely, the present invention can be realized as a computer program product. Also the program may be regarded as either a program product (i.e. manufac-ture) or a process embodying the load distribution method (process), expressed in a program language.") Uchida at [0039] ("First, an outline of an exemplary embodiment of the present invention will be described. Hereinafter, the reference characters used in this outline are used only as examples to facilitate comprehension and are not intended to limit the present invention to the illustrated modes. As illustrated in FIG. 1, a packet forwarding apparatus 1 according to an exemplary embodiment of the present invention includes: a process rule storage unit 11 storing process rules applied to received packets; a packet process unit 12 reading a process rule matching a received packet from the process rule storage unit to execute a forwarding process or the like; an end deter-mination information extraction unit 13 extracting informa-tion for checking a flow end from a received packet; and a flow end check unit 14 deleting, when a flow end is checked based on the extracted information, a process rule corre-sponding to the flow.") Uchida at [0070] ("The process rule search unit C14 determines actions (process contents) to be applied to the received packet, based on the information transmitted from the packet

No.	'111 Patent Claim 1	Uchida
		process infor-mation extraction unit C12 and outputs the actions to the packet process unit C15. More specifically, the process rule search unit C14 searches the process rule storage unit C13 for a process rule matching the information extracted from the received packet transmitted from the packet process informa-tion extraction unit C12 and outputs the process rule to the packet process unit C15. If a process rule matching the received packet exists, the process rule search unit C14 extracts actions (process contents) defined in the action field of the process rule and outputs the actions to the packet process unit C15. If no process rule matching the received packet is found, the process rule search unit C14 transmits the received packet or information extracted from the received packet to a predetermined control apparatus (controller; see FIG. 18) to request creation and transmission of a process rule. Upon receiving a process rule from the predetermined control apparatus (controller; see FIG. 18), the process rule search unit C14 registers the received process rule in the process rule storage unit C13. Next, the process rule search unit C14 extracts actions (process contents) from the process rule and outputs the actions to the packet process unit C15.") Uchida at [0071] ("The first process rule in the flow table in FIG. 20 is applied to packets indicating that the physical port number of the packet forwarding apparatus is 1; the source network is 192.168.0./24; the protocol type is Ox0006 (TCP); the desti-nation network is 192.168.1./24; and the TCP destination port number is 80. Accordingly, the actions (process contents) of changing the IP source address of the packet to 192.168.1/24 and transmitting the packet from physical port number 2 of the forwarding apparatus are executed. Similarly, the second process rule in FIG. 20 is applied to packets indicating a destination port number other than HTTP port numbers. Accordingly, the actions (process contents) of changing the IP source address of the packet to 192.168.1/24 and transmit-ting the packet from physical port number 3 of the forwarding apparatus are executed.") Uchida at [0083] ("Next, upon receiving the extracted information, the process rule search unit C14 searches the process rule storage unit C13 for a process rule matching the extracted informa-tion (step S104). If a process rule matching the extracted information is found, the process rule search unit C14 extracts actions (process contents) defined in the process rule and transmits the actions to the packet process unit C15 (step S105). If no process rule matching the extracted information is found, the process rule search unit C14 transmits the received packet or information extracted from the received packet to a predetermined control apparatus (controller; see FIG. 18) to request creation and transmission of a process rule (step

No.	'111 Patent Claim 1	Uchida
		S106). Upon receiving a process rule from the predeter-mined control apparatus (controller; see FIG. 18), the process rule search unit C14 registers the received process rule in the process rule storage unit C13, extracts actions (process con-tents) in the process rule, and outputs the actions to the packet process unit C15 (step S107).") Uchida at [0106] ("First, the computer D10 starts communication with the server D20. For an initial establishment of communication, a packet (SYN) is inputted to the packet forwarding apparatus C1 (start of ACTIVE OPEN through SYN forwarding in FIG. 6). The packet reception unit C10 receives and stores this first packet in the packet storage unit C11 (steps S101 to S102 in FIG. 3).") 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.") Uchida at [0109] ("Next, a process rule corresponding to the received packet is searched for. The packet process information extraction unit C12 transmits the extracted information (representing that the ingress port is 1; the source address is 192.168.0.10; the destination address is 192.168.1.10; and the protocol is TCP (the type is 0x0006)) to a process rule search unit C14. The process rule search unit C14 refers to the process rule storage unit C13. Since no process rule matching the packet exists, the process rule search unit C14 queries the control apparatus (controller) D1 (steps S104 and S106 in FIG. 3). The control apparatus (controller) D1 creates a process rule including process contents representing modification of the source IP and forward from a specified transport port and transmits the process rule to the packet forwarding apparatus C1. The process rule search unit C14 registers the process rule received from the control apparatus (controller) D1 as a new entry in the process rule storage unit C13 (step S107 in FIG. 3). At this stage, as illustrated in FIG. 9, a new process rule is added to the process rule storage unit C13.")

No.	'111 Patent Claim 1	Uchida
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	Uchida discloses responsive to the packet not satisfying the criterion, sending, by the network node over the packet network, the packet to the second entity. For example, Uchida discloses a process rule matching a received packet from the process rule storage unit to execute a forwarding process. If no process rule matching the received packet is found, the process rule search unit C14 transmits the received packet or information extracted from the received packet to a predetermined control apparatus. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Uchida 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. Uchida at [0012] ("According to a first aspect of the present invention, there is provided a packet forwarding apparatus comprising: a process rule storage unit that stores process rules defining a process content for each flow; a packet process unit that executes processing a received packet by using a process rule matching the received packet among the process rules and; an end determination information extraction unit that extracts information for checking a flow end from the received packet; and a flow end check unit that deletes, when a flow end is checked based on the extracted information, a process rule corresponding to the flow.") Uchida at [0014] ("According to a third aspect of the present invention, there is provided a process rule update method causing a packet forwarding apparatus comprising: a process rule stor-age unit that stores process rules defining a process content for each flow; and a packet process unit that executes and processing a received packet by using a process rule matching the received packet among the process rules to execute: extracting information for checking a flow received packet; and deleting, when a flow end is checked based on the extracted information, a process rule corre-sponding to the flow from the process rule storage unit. The present method is connected to a certain machine referred to as a packet forwarding apparatus comprising a process rule storage unit and a packet process unit.") Uchida at [0015] ("According to a fourth aspect of the present inven-tion, there is provided a program, causing a computer forming a packet forwarding apparatus comprising: a process rule storage unit that stores process rules defining a process con-tent for each flow; and a

No.	'111 Patent Claim 1	Uchida
		packet process unit that executes processing a received packet by using a process rule matching a received packet among the process rules to execute the processes of: extracting information for determining a flow end from the received packet; and deleting, when a flow end is determined based on the extracted information, a process rule corresponding to the flow from the process rule storage unit. The program can be stored in a computer-readable recording medium. Namely, the present invention can be realized as a computer program product. Also the program may be regarded as either a program product (i.e. manufac-ture) or a process embodying the load distribution method (process), expressed in a program language.") Uchida at [0039] ("First, an outline of an exemplary embodiment of the present invention will be described. Hereinafter, the reference characters used in this outline are used only as examples to facilitate comprehension and are not intended to limit the present invention to the illustrated modes. As illustrated in FIG. 1, a packet forwarding apparatus 1 according to an exemplary embodiment of the present invention includes: a process rule storage unit 11 storing process rules applied to received packets; a packet process unit 12 reading a process rule matching a received packet from the process rule storage unit to execute a forwarding process or the like; an end deter-mination information extraction unit 13 extracting informa-tion for checking a flow end from a received packet; and a flow end check unit 14 deleting, when a flow end is checked based on the extracted information, a process rule corre-sponding to the flow.") Uchida at [0068] ("In addition, in the actions in FIG. 19, "Forward" for forwarding the packet, "Drop" for dropping the packet, and "Modify-Field" for modifying information about the packet are present. An arbitrary number of actions can be registered in the actions (see the list in FIG. 21. However, in the case of Non-Patent Document 2, if the action field is blank, "Drop" is executed). For example, the Modify-Filed actions include: setting/changing of a VLAN ID or priority; stripping of a VLAN header; changing of an Ethernet (registered trademark) or IP source/destination address; and changing of a TCP/UDP source/destination port (other examples include changing of an IP ToS (Type of Service) field).") Uchida at [0070] ("The process rule search unit C14 determines actions (process contents) to be applied to the received packet, based on the information transmitted from the packet

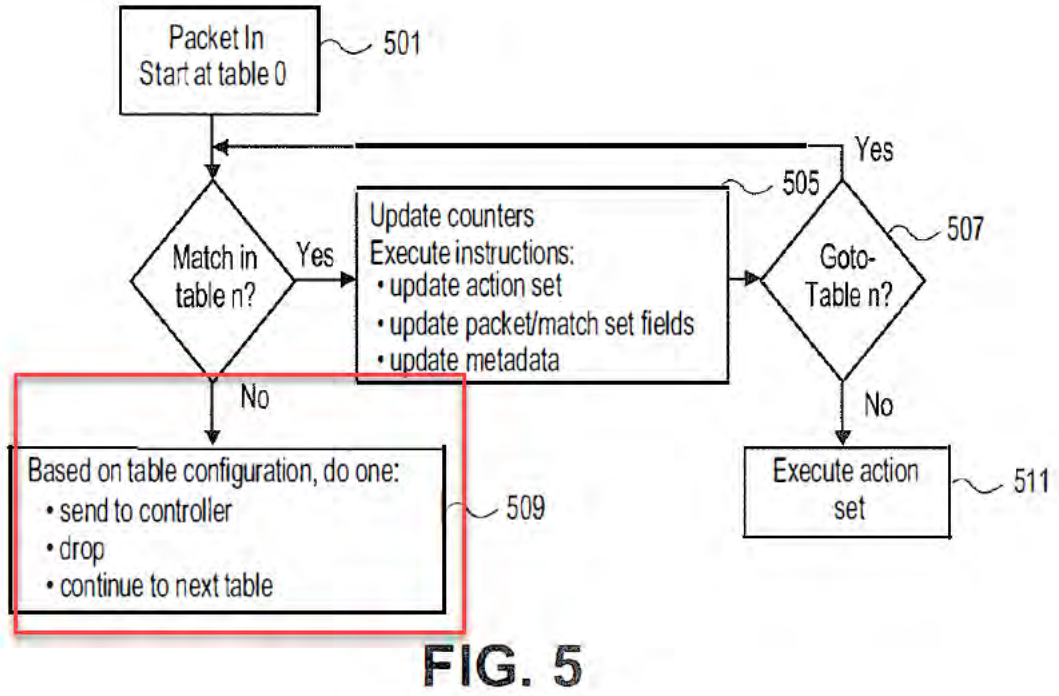
No.	'111 Patent Claim 1	Uchida
		process information extraction unit C12 and outputs the actions to the packet process unit C15. More specifically, the process rule search unit C14 searches the process rule storage unit C13 for a process rule matching the information extracted from the received packet transmitted from the packet process information extraction unit C12 and outputs the process rule to the packet process unit C15. If a process rule matching the received packet exists, the process rule search unit C14 extracts actions (process contents) defined in the action field of the process rule and outputs the actions to the packet process unit C15. If no process rule matching the received packet is found, the process rule search unit C14 transmits the received packet or information extracted from the received packet to a predetermined control apparatus (controller; see FIG. 18) to request creation and transmission of a process rule. Upon receiving a process rule from the predetermined control apparatus (controller; see FIG. 18), the process rule search unit C14 registers the received process rule in the process rule storage unit C13. Next, the process rule search unit C14 extracts actions (process contents) from the process rule and outputs the actions to the packet process unit C15.") Uchida at [0071] ("The first process rule in the flow table in FIG. 20 is applied to packets indicating that the physical port number of the packet forwarding apparatus is 1; the source network is 192.168.0./24; the protocol type is Ox0006 (TCP); the desti-nation network is 192.168.1./24; and the TCP destination port number is 80. Accordingly, the actions (process contents) of changing the IP source address of the packet to 192.168.1/24 and transmitting the packet from physical port number 2 of the forwarding apparatus are executed. Similarly, the second process rule in FIG. 20 is applied to packets indicating a destination port number other than HTTP port numbers. Accordingly, the actions (process contents) of changing the IP source address of the packet to 192.168.1/24 and transmit-ting the packet from physical port number 3 of the forwarding apparatus are executed.") Uchida at [0083] ("Next, upon receiving the extracted information, the process rule search unit C14 searches the process rule storage unit C13 for a process rule matching the extracted informa-tion (step S104). If a process rule matching the extracted information is found, the process rule search unit C14 extracts actions (process contents) defined in the process rule and transmits the actions to the packet process unit C15 (step S105). If no process rule matching the extracted information is found, the process rule search unit C14 transmits the received packet or information extracted from the received packet to a predetermined control apparatus (controller; see FIG. 18) to request creation and transmission of a process rule (step

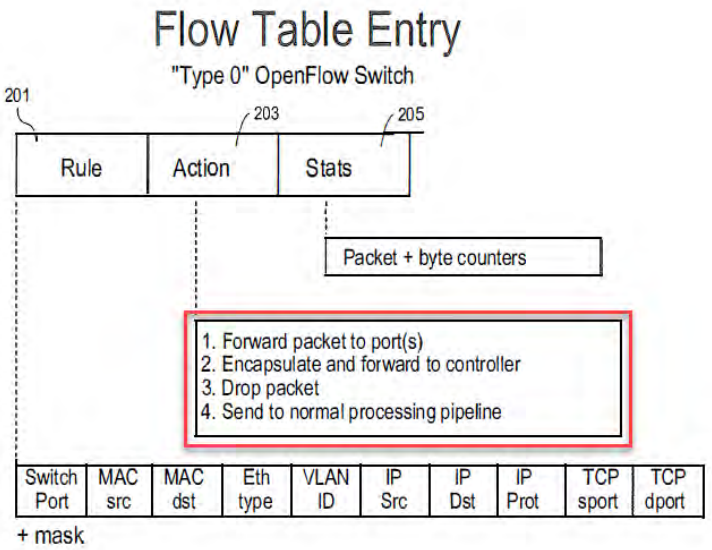
No.	'111 Patent Claim 1	Uchida
		S106). Upon receiving a process rule from the predeter-mined control apparatus (controller; see FIG. 18), the process rule search unit C14 registers the received process rule in the process rule storage unit C13, extracts actions (process con-tents) in the process rule, and outputs the actions to the packet process unit C15 (step S107).") Uchida at [0106] ("First, the computer D10 starts communication with the server D20. For an initial establishment of communication, a packet (SYN) is inputted to the packet forwarding apparatus C1 (start of ACTIVE OPEN through SYN forwarding in FIG. 6). The packet reception unit C10 receives and stores this first packet in the packet storage unit C11 (steps S101 to S102 in FIG. 3).") 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.") Uchida at [0109] ("Next, a process rule corresponding to the received packet is searched for. The packet process information extraction unit C12 transmits the extracted information (representing that the ingress port is 1; the source address is 192.168.0.10; the destination address is 192.168.1.10; and the protocol is TCP (the type is 0x0006)) to a process rule search unit C14. The process rule search unit C14 refers to the process rule storage unit C13. Since no process rule matching the packet exists, the process rule search unit C14 queries the control apparatus (controller) D1 (steps S104 and S106 in FIG. 3). The control apparatus (controller) D1 creates a process rule including process contents representing modification of the source IP and forward from a specified transport port and transmits the process rule to the packet forwarding apparatus C1. The process rule search unit C14 registers the process rule received from the control apparatus (controller) D1 as a new entry in the process rule storage unit C13 (step S107 in FIG. 3). At this stage, as illustrated in FIG. 9, a new process rule is added to the process rule storage unit C13.") Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Uchida in combination with (1) the knowledge of a person of ordinary skill in the art, alone

No.	'111 Patent Claim 1	Uchida
		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 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.") 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.") 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.") 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

No.	'111 Patent Claim 1	Uchida
		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.") 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.") 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.") 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

No.	'111 Patent Claim 1	Uchida
		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.") 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.") 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.") Kempf at Figure 5 (annotation added)

No.	'111 Patent Claim 1	Uchida
		 FIG. 5 Kempf at Figure 2 (annotation added)

No.	'111 Patent Claim 1	Uchida
		 **FIG. 2** 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. 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

No.	'111 Patent Claim 1	Uchida
		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.") 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.") 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

No.	'111 Patent Claim 1	Uchida
		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.") 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.") 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.") 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

No.	'111 Patent Claim 1	Uchida
		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.") 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.") 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.

No.	'111 Patent Claim 1	Uchida
		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.") 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.	Uchida 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. For example, Uchida discloses a process rule matching a received packet from the process rule storage unit to execute a forwarding process. If a process rule matching the received packet exists, the process rule search unit C14 extracts actions (process contents) defined in the action field of the process rule and outputs the actions to the packet process unit C15. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures,

No.	'111 Patent Claim 1	Uchida
		this limitation is met. To the extent that the Uchida is found to not meet this 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. Uchida at [0012] ("According to a first aspect of the present invention, there is provided a packet forwarding apparatus comprising: a process rule storage unit that stores process rules defining a process content for each flow; a packet process unit that executes processing a received packet by using a process rule matching the received packet among the process rules and; an end determination information extraction unit that extracts information for checking a flow end from the received packet; and a flow end check unit that deletes, when a flow end is checked based on the extracted information, a process rule corresponding to the flow.") Uchida at [0014] ("According to a third aspect of the present invention, there is provided a process rule update method causing a packet forwarding apparatus comprising: a process rule stor-age unit that stores process rules defining a process content for each flow; and a packet process unit that executes and processing a received packet by using a process rule matching the received packet among the process rules to execute: extracting information for checking a flow received packet; and deleting, when a flow end is checked based on the extracted information, a process rule corre-sponding to the flow from the process rule storage unit. The present method is connected to a certain machine referred to as a packet forwarding apparatus comprising a process rule storage unit and a packet process unit.") Uchida at [0015] ("According to a fourth aspect of the present inven-tion, there is provided a program, causing a computer forming a packet forwarding apparatus comprising: a process rule storage unit that stores process rules defining a process con-tent for each flow; and a packet process unit that executes processing a received packet by using a process rule matching a received packet among the process rules to execute the processes of: extracting information for determining a flow end from the received packet; and deleting, when a flow end is determined based on the extracted information, a process rule corresponding to the flow from the process rule storage unit. The program can be stored in a computer-readable recording medium. Namely, the present invention can be realized as a computer program product. Also the program may be regarded as either a program product (i.e. manufac-ture)

No.	'111 Patent Claim 1	Uchida
		or a process embodying the load distribution method (process), expressed in a program language.") Uchida at [0039] ("First, an outline of an exemplary embodiment of the present invention will be described. Hereinafter, the reference characters used in this outline are used only as examples to facilitate comprehension and are not intended to limit the present invention to the illustrated modes. As illustrated in FIG. 1, a packet forwarding apparatus 1 according to an exemplary embodiment of the present invention includes: a process rule storage unit 11 storing process rules applied to received packets; a packet process unit 12 reading a process rule matching a received packet from the process rule storage unit to execute a forwarding process or the like; an end deter-mination information extraction unit 13 extracting informa-tion for checking a flow end from a received packet; and a flow end check unit 14 deleting, when a flow end is checked based on the extracted information, a process rule corre-sponding to the flow.") Uchida at [0068] ("In addition, in the actions in FIG. 19, "Forward" for forwarding the packet, "Drop" for dropping the packet, and "Modify-Field" for modifying information about the packet are present. An arbitrary number of actions can be registered in the actions (see the list in FIG. 21. However, in the case of Non-Patent Document 2, if the action field is blank, "Drop" is executed). For example, the Modify-Filed actions include: setting/changing of a VLAN ID or priority; stripping of a VLAN header; changing of an Ethernet (registered trademark) or IP source/destination address; and changing of a TCP/UDP source/destination port (other examples include changing of an IP ToS (Type of Service) field).") Uchida at [0070] ("The process rule search unit C14 determines actions (process contents) to be applied to the received packet, based on the information transmitted from the packet process information extraction unit C12 and outputs the actions to the packet process unit C15. More specifically, the process rule search unit C14 searches the process rule storage unit C13 for a process rule matching the information extracted from the received packet transmitted from the packet process information extraction unit C12 and outputs the process rule to the packet process unit C15. If a process rule matching the received packet exists, the process rule search unit C14 extracts actions (process contents) defined in the action field of the process rule and outputs the actions to the packet process unit C15. If no process rule

No.	'111 Patent Claim 1	Uchida
		matching the received packet is found, the process rule search unit C14 transmits the received packet or information extracted from the received packet to a predetermined control apparatus (controller; see FIG. 18) to request creation and transmission of a process rule. Upon receiving a process rule from the predetermined control apparatus (controller; see FIG. 18), the process rule search unit C14 registers the received process rule in the process rule storage unit C13. Next, the process rule search unit C14 extracts actions (process contents) from the process rule and outputs the actions to the packet process unit C15.") Uchida at [0071] ("The first process rule in the flow table in FIG. 20 is applied to packets indicating that the physical port number of the packet forwarding apparatus is 1; the source network is 192.168.0./24; the protocol type is Ox0006 (TCP); the desti-nation network is 192.168.1./24; and the TCP destination port number is 80. Accordingly, the actions (process contents) of changing the IP source address of the packet to 192.168.1/24 and transmitting the packet from physical port number 2 of the forwarding apparatus are executed. Similarly, the second process rule in FIG. 20 is applied to packets indicating a destination port number other than HTTP port numbers. Accordingly, the actions (process contents) of changing the IP source address of the packet to 192.168.1/24 and transmit-ting the packet from physical port number 3 of the forwarding apparatus are executed.") Uchida at [0083] ("Next, upon receiving the extracted information, the process rule search unit C14 searches the process rule storage unit C13 for a process rule matching the extracted informa-tion (step S104). If a process rule matching the extracted information is found, the process rule search unit C14 extracts actions (process contents) defined in the process rule and transmits the actions to the packet process unit C15 (step S105). If no process rule matching the extracted information is found, the process rule search unit C14 transmits the received packet or information extracted from the received packet to a predetermined control apparatus (controller; see FIG. 18) to request creation and transmission of a process rule (step S106). Upon receiving a process rule from the predeter-mined control apparatus (controller; see FIG. 18), the process rule search unit C14 registers the received process rule in the process rule storage unit C13, extracts actions (process con-tents) in the process rule, and outputs the actions to the packet process unit C15 (step S107).") Uchida at [0106] ("First, the computer D10 starts communication with the server D20. For an initial establishment of communication, a packet (SYN) is inputted to the packet forwarding

No.	'111 Patent Claim 1	Uchida
		apparatus C1 (start of ACTIVE OPEN through SYN forwarding in FIG. 6). The packet reception unit C10 receives and stores this first packet in the packet storage unit C11 (steps S101 to S102 in FIG. 3).") 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.") Uchida at [0109] ("Next, a process rule corresponding to the received packet is searched for. The packet process information extraction unit C12 transmits the extracted information (representing that the ingress port is 1; the source address is 192.168.0.10; the destination address is 192.168.1.10; and the protocol is TCP (the type is 0x0006)) to a process rule search unit C14. The process rule search unit C14 refers to the process rule storage unit C13. Since no process rule matching the packet exists, the process rule search unit C14 queries the control apparatus (controller) D1 (steps S104 and S106 in FIG. 3). The control apparatus (controller) D1 creates a process rule including process contents representing modification of the source IP and forward from a specified transport port and transmits the process rule to the packet forwarding apparatus C1. The process rule search unit C14 registers the process rule received from the control apparatus (controller) D1 as a new entry in the process rule storage unit C13 (step S107 in FIG. 3). At this stage, as illustrated in FIG. 9, a new process rule is added to the process rule storage unit C13.") Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Uchida 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.

No.	'111 Patent Claim 1	Uchida
		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 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.") 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.")

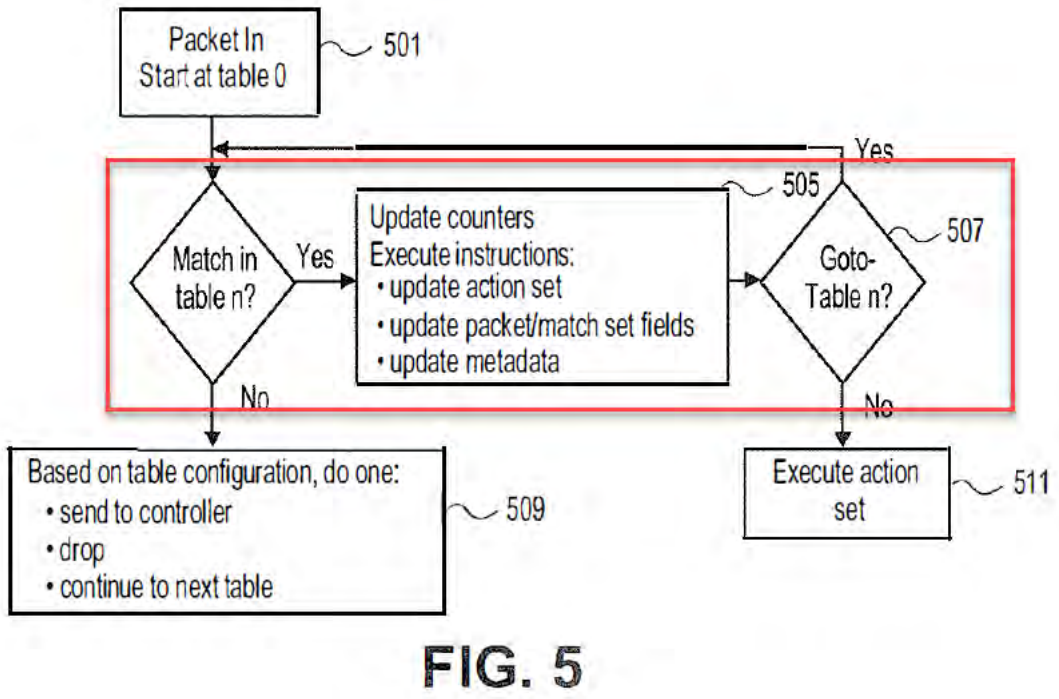
No.	'111 Patent Claim 1	Uchida
		Swenson at [0029] ("As a flow is sent to the network controller 140 for inspection, historical network traffic data stored at the net-work controller 140 may be searched. The historical network traffic data includes information such as subscriber informa-tion, the cell 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.") 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.") 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.")

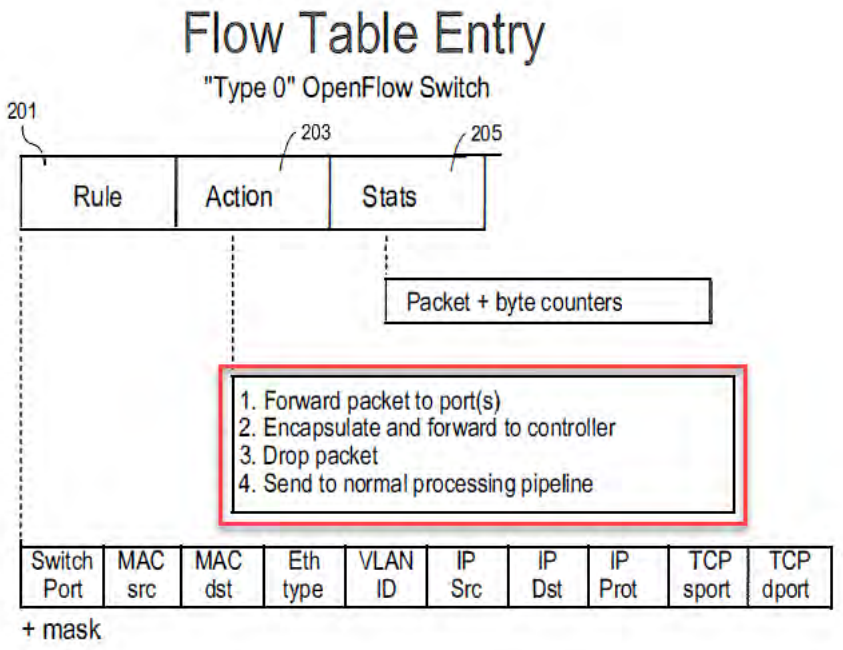
No.	'111 Patent Claim 1	Uchida
		Swenson at [0045] ("The steering device interface 316 interacts with an external routing appliance, such as the steering device 130 to divert portions of the network traffic (e.g., large object net-work flows). Existing routing appliances in most carrier net-works are 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.") 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.") 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

No.	'111 Patent Claim 1	Uchida
		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.") 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.") 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.

No.	'111 Patent Claim 1	Uchida
		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.") For example, Kempf discloses sending the packet from the network element to the controller or another table to an entity that is included in the instruction and is other than the second entity, 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 be provided by any type of network including a local area network (LAN) or a wide area network (WAN), such as the Internet.") 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.") 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.")

No.	'111 Patent Claim 1	Uchida
		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.") 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.") 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.") Kempf at [0106] ("This encapsulates the packet and sends it to the OpenFlow controller.")

No.	'111 Patent Claim 1	Uchida
		Kempf at Figure 5 (annotation added)  FIG. 5 Kempf at Figure 2 (annotation added)

No.	'111 Patent Claim 1	Uchida
		 FIG. 2

No.	'111 Patent Claim 2	Uchida
2[a]	The method according to claim 1, wherein the instruction is 'probe', 'mirror', or 'terminate' instruction, and	Uchida discloses the method according to claim 1, wherein the instruction is 'probe', 'mirror', or 'terminate' instruction. For example, Uchida discloses the actions such as "Forward" for forwarding the packet, "Drop" for dropping the packet, and "Modify-Field" for modifying information about the packet.

No.	'111 Patent Claim 2	Uchida
		Uchida at [0068] ("In addition, in the actions in FIG. 19, "Forward" for forwarding the packet, "Drop" for dropping the packet, and "Modify-Field" for modifying information about the packet are present. An arbitrary number of actions can be registered in the actions (see the list in FIG. 21. However, in the case of Non-Patent Document 2, if the action field is blank, "Drop" is executed). For example, the Modify-Filed actions include: setting/changing of a VLAN ID or priority; stripping of a VLAN header; changing of an Ethernet (registered trademark) or IP source/destination address; and changing of a TCP/UDP source/destination port (other examples include changing of an IP ToS (Type of Service) field).") Uchida at [0070] ("The process rule search unit C14 determines actions (process contents) to be applied to the received packet, based on the information transmitted from the packet process information extraction unit C12 and outputs the actions to the packet process unit C15. More specifically, the process rule search unit C14 searches the process rule storage unit C13 for a process rule matching the information extracted from the received packet transmitted from the packet process information extraction unit C12 and outputs the process rule to the packet process unit C15. If a process rule matching the received packet exists, the process rule search unit C14 extracts actions (process contents) defined in the action field of the process rule and outputs the actions to the packet process unit C15. If no process rule matching the received packet is found, the process rule search unit C14 transmits the received packet or information extracted from the received packet to a predetermined control apparatus (controller; see FIG. 18) to request creation and transmission of a process rule. Upon receiving a process rule from the predetermined control apparatus (controller; see FIG. 18), the process rule search unit C14 registers the received process rule in the process rule storage unit C13. Next, the process rule search unit C14 extracts actions (process contents) from the process rule and outputs the actions to the packet process unit C15.")

No.	'111 Patent Claim 2	Uchida
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.	Uchida 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. For example, Uchida discloses the action such as "Drop" for dropping the packet. Uchida at [0068] ("In addition, in the actions in FIG. 19, "Forward" for forwarding the packet, "Drop" for dropping the packet, and "Modify-Field" for modifying information about the packet are present. An arbitrary number of actions can be registered in the actions (see the list in FIG. 21. However, in the case of Non-Patent Document 2, if the action field is blank, "Drop" is executed). For example, the Modify-Filed actions include: setting/changing of a VLAN ID or priority; stripping of a VLAN header; changing of an Ethernet (registered trademark) or IP source/destination address; and changing of a TCP/UDP source/destination port (other examples include changing of an IP ToS (Type of Service) field).") Uchida at [0070] ("The process rule search unit C14 determines actions (process contents) to be applied to the received packet, based on the information transmitted from the packet process information extraction unit C12 and outputs the actions to the packet process unit C15. More specifically, the process rule search unit C14 searches the process rule storage unit C13 for a process rule matching the information extracted from the received packet transmitted from the packet process information extraction unit C12 and outputs the process rule to the packet process unit C15. If a process rule matching the received packet exists, the process rule search unit C14 extracts actions (process contents) defined in the action field of the process rule and outputs the actions to the packet process unit C15. If no process rule matching the received packet is found, the process rule search unit C14 transmits the received packet or information extracted from the received packet to a predetermined control apparatus (controller; see FIG. 18) to request creation and transmission of a process rule. Upon receiving a process rule from the predetermined control apparatus (controller; see FIG. 18), the process rule search unit C14 registers the received process rule in the process rule storage unit C13. Next, the process rule search unit C14 extracts actions (process contents) from the process rule and outputs the actions to the packet process unit C15.")

No.	'111 Patent Claim 3	Uchida
3[a]	The method according to claim 1, wherein the instruction is a 'probe', a 'mirror', or a 'terminate' instruction, and	Uchida discloses the method according to claim 1, wherein the instruction is a 'probe', a 'mirror', or a 'terminate' instruction. *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.	Uchida 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. For example, Uchida discloses forwarding packets to both the destination and the controller. A person of ordinary skill would understand that a process rule matching the received packet could correspond to an action of sending the packet to both the destination 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 Copeland 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. Uchida at [0012] ("According to a first aspect of the present invention, there is provided a packet forwarding apparatus comprising: a process rule storage unit that stores process rules defining a process content for each flow; a packet process unit that executes processing a received packet by using a process rule matching the received packet among the process rules and; an end determination information extraction unit that extracts information for checking a flow end from the received packet; and a flow end check unit that deletes, when a flow end is checked based on the extracted information, a process rule corresponding to the flow.") Uchida at [0014] ("According to a third aspect of the present invention, there is provided a process rule update method causing a packet forwarding apparatus comprising: a process rule stor-age unit that stores process rules defining a process content for each flow; and a packet process unit that executes and processing a received packet by using a process rule matching the received packet among the process rules to execute: extracting information for checking a flow received packet; and deleting, when a flow end is checked based on the extracted

No.	'111 Patent Claim 3	Uchida
		information, a process rule corre-sponding to the flow from the process rule storage unit. The present method is connected to a certain machine referred to as a packet forwarding apparatus comprising a process rule storage unit and a packet process unit.") Uchida at [0015] ("According to a fourth aspect of the present inven-tion, there is provided a program, causing a computer forming a packet forwarding apparatus comprising: a process rule storage unit that stores process rules defining a process con-tent for each flow; and a packet process unit that executes processing a received packet by using a process rule matching a received packet among the process rules to execute the processes of: extracting information for determining a flow end from the received packet; and deleting, when a flow end is determined based on the extracted information, a process rule corresponding to the flow from the process rule storage unit. The program can be stored in a computer-readable recording medium. Namely, the present invention can be realized as a computer program product. Also the program may be regarded as either a program product (i.e. manufac-ture) or a process embodying the load distribution method (process), expressed in a program language.") Uchida at [0039] ("First, an outline of an exemplary embodiment of the present invention will be described. Hereinafter, the reference characters used in this outline are used only as examples to facilitate comprehension and are not intended to limit the present invention to the illustrated modes. As illustrated in FIG. 1, a packet forwarding apparatus 1 according to an exemplary embodiment of the present invention includes: a process rule storage unit 11 storing process rules applied to received packets; a packet process unit 12 reading a process rule matching a received packet from the process rule storage unit to execute a forwarding process or the like; an end deter-mination information extraction unit 13 extracting informa-tion for checking a flow end from a received packet; and a flow end check unit 14 deleting, when a flow end is checked based on the extracted information, a process rule corre-sponding to the flow.") Uchida at [0061] ("The packet reception unit Cl0 receives a packet from another node and temporarily stores the received packet in the packet storage unit Cll. In addition, the packet recep-tion unit Cl0 can store reception status information repre-senting an ingress port (receiving port ID information) and the like simultaneously. After storing the received packet,

No.	'111 Patent Claim 3	Uchida
		the packet reception unit Cl0 notifies the packet process infor-mation extraction unit C12 and the end determination infor-mation extraction unit Cl 7 of reception of the packet.") Uchida at [0068] ("In addition, in the actions in FIG. 19, "Forward" for forwarding the packet, "Drop" for dropping the packet, and "Modify-Field" for modifying information about the packet are present. An arbitrary number of actions can be registered in the actions (see the list in FIG. 21. However, in the case of Non-Patent Document 2, if the action field is blank, "Drop" is executed). For example, the Modify-Filed actions include: setting/changing of a VLAN ID or priority; stripping of a VLAN header; changing of an Ethernet (registered trademark) or IP source/destination address; and changing of a TCP/UDP source/destination port (other examples include changing of an IP ToS (Type of Service) field).") Uchida at [0070] ("The process rule search unit C14 determines actions (process contents) to be applied to the received packet, based on the information transmitted from the packet process information extraction unit C12 and outputs the actions to the packet process unit C15. More specifically, the process rule search unit C14 searches the process rule storage unit C13 for a process rule matching the information extracted from the received packet transmitted from the packet process information extraction unit C12 and outputs the process rule to the packet process unit C15. If a process rule matching the received packet exists, the process rule search unit C14 extracts actions (process contents) defined in the action field of the process rule and outputs the actions to the packet process unit C15. If no process rule matching the received packet is found, the process rule search unit C14 transmits the received packet or information extracted from the received packet to a predetermined control apparatus (controller; see FIG. 18) to request creation and transmission of a process rule. Upon receiving a process rule from the predetermined control apparatus (controller; see FIG. 18), the process rule search unit C14 registers the received process rule in the process rule storage unit C13. Next, the process rule search unit C14 extracts actions (process contents) from the process rule and outputs the actions to the packet process unit C15.") Uchida at [0071] ("The first process rule in the flow table in FIG. 20 is applied to packets indicating that the physical port number of the packet forwarding apparatus is 1; the source network is 192.168.0./24; the protocol type is Ox0006 (TCP); the desti-nation network is 192.168.1./24; and the TCP destination port number is 80. Accordingly, the actions (process contents) of changing the IP source address of the packet to 192.168.1/24 and transmitting

No.	'111 Patent Claim 3	Uchida
		the packet from physical port number 2 of the forwarding apparatus are executed. Similarly, the second process rule in FIG. 20 is applied to packets indicating a destination port number other than HTTP port numbers. Accordingly, the actions (process contents) of changing the IP source address of the packet to 192.168.1/24 and transmit-ting the packet from physical port number 3 of the forwarding apparatus are executed.") Uchida at [0081] ("Next, the packet reception unit Cl0 notifies the packet process information extraction unit C12 and the end determination information extraction unit Cl 7 of reception of the packet. Accordingly, two processes will be executed simultaneously (branches after step S102).") Uchida at [0083] ("Next, upon receiving the extracted information, the process rule search unit C14 searches the process rule storage unit C13 for a process rule matching the extracted informa-tion (step S104). If a process rule matching the extracted information is found, the process rule search unit C14 extracts actions (process contents) defined in the process rule and transmits the actions to the packet process unit C15 (step S105). If no process rule matching the extracted information is found, the process rule search unit C14 transmits the received packet or information extracted from the received packet to a predetermined control apparatus (controller; see FIG. 18) to request creation and transmission of a process rule (step S106). Upon receiving a process rule from the predeter-mined control apparatus (controller; see FIG. 18), the process rule search unit C14 registers the received process rule in the process rule storage unit C13, extracts actions (process con-tents) in the process rule, and outputs the actions to the packet process unit C15 (step S107).") Uchida at [0106] ("First, the computer D10 starts communication with the server D20. For an initial establishment of communication, a packet (SYN) is inputted to the packet forwarding apparatus C1 (start of ACTIVE OPEN through SYN forwarding in FIG. 6). The packet reception unit C10 receives and stores this first packet in the packet storage unit C11 (steps S101 to S102 in FIG. 3).") 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

No.	'111 Patent Claim 3	Uchida
		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.") Uchida at [0109] ("Next, a process rule corresponding to the received packet is searched for. The packet process information extraction unit C12 transmits the extracted information (representing that the ingress port is 1; the source address is 192.168.0.10; the destination address is 192.168.1.10; and the protocol is TCP (the type is 0x0006)) to a process rule search unit C14. The process rule search unit C14 refers to the process rule storage unit C13. Since no process rule matching the packet exists, the process rule search unit C14 queries the control apparatus (controller) D1 (steps S104 and S106 in FIG. 3). The control apparatus (controller) D1 creates a process rule including process contents representing modification of the source IP and forward from a specified transport port and transmits the process rule to the packet forwarding apparatus C1. The process rule search unit C14 registers the process rule received from the control apparatus (controller) D1 as a new entry in the process rule storage unit C13 (step S107 in FIG. 3). At this stage, as illustrated in FIG. 9, a new process rule is added to the process rule storage unit C13.") Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Uchida 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. 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. 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

No.	'111 Patent Claim 3	Uchida
		perform pro-grammed actions on packets of a packet flow based on this data. Such programmed actions may include: Allow-explicitly allow a certain network flow to proceed to its destination Block-explicitly block a certain flow from traversing SDN 106 Mirror-allow the traffic, but send a copy of the traffic for deeper inspection or recording to, e.g., one of service devices 116 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. Transform-modify or translate values of headers of packets in the network flow Encapsulate-encapsulate packets in the network flow with a particular header") 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.") 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

No.	'111 Patent Claim 3	Uchida
		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.") 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. 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.") 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

No.	'111 Patent Claim 3	Uchida
		(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.") 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.") 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

No.	’111 Patent Claim 3	Uchida
		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.”) 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.”) 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	Uchida
4[a]	The method according to claim 1, wherein the instruction is 'probe', 'mirror', or 'terminate' instruction, and	Uchida discloses the method according to claim 1, wherein the instruction is 'probe', 'mirror', or 'terminate' instruction. *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;	Uchida 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. For example, Uchida discloses a process rule matching a received packet from the process rule storage unit to execute a forwarding process. If a process rule matching the received packet exists, the process rule search unit C14 extracts actions (process contents) defined in the action field of the process rule and outputs the actions to the packet process unit C15. A person of ordinary skill in the art would understand that a process rule matching could direct the packet to the controller for further processing. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Uchida 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. Uchida at [0012] ("According to a first aspect of the present invention, there is provided a packet forwarding apparatus comprising: a process rule storage unit that stores process rules defining a process content for each flow; a packet process unit that executes processing a received packet by using a process rule matching the received packet among the process rules and; an end determination information extraction unit that extracts information for checking a flow end from the received packet; and a flow end check unit that deletes, when a flow end is checked based on the extracted information, a process rule corresponding to the flow.") Uchida at [0014] ("According to a third aspect of the present invention, there is provided a process rule update method causing a packet forwarding apparatus comprising: a process rule stor-age unit that stores process rules defining a process content for each flow; and a packet process unit that executes and processing a received packet by using a process rule matching

No.	'111 Patent Claim 4	Uchida
		the received packet among the process rules to execute: extracting information for checking a flow received packet; and deleting, when a flow end is checked based on the extracted information, a process rule corre-sponding to the flow from the process rule storage unit. The present method is connected to a certain machine referred to as a packet forwarding apparatus comprising a process rule storage unit and a packet process unit.") Uchida at [0015] ("According to a fourth aspect of the present inven-tion, there is provided a program, causing a computer forming a packet forwarding apparatus comprising: a process rule storage unit that stores process rules defining a process con-tent for each flow; and a packet process unit that executes processing a received packet by using a process rule matching a received packet among the process rules to execute the processes of: extracting information for determining a flow end from the received packet; and deleting, when a flow end is determined based on the extracted information, a process rule corresponding to the flow from the process rule storage unit. The program can be stored in a computer-readable recording medium. Namely, the present invention can be realized as a computer program product. Also the program may be regarded as either a program product (i.e. manufac-ture) or a process embodying the load distribution method (process), expressed in a program language.") Uchida at [0039] ("First, an outline of an exemplary embodiment of the present invention will be described. Hereinafter, the reference characters used in this outline are used only as examples to facilitate comprehension and are not intended to limit the present invention to the illustrated modes. As illustrated in FIG. 1, a packet forwarding apparatus 1 according to an exemplary embodiment of the present invention includes: a process rule storage unit 11 storing process rules applied to received packets; a packet process unit 12 reading a process rule matching a received packet from the process rule storage unit to execute a forwarding process or the like; an end deter-mination information extraction unit 13 extracting informa-tion for checking a flow end from a received packet; and a flow end check unit 14 deleting, when a flow end is checked based on the extracted information, a process rule corre-sponding to the flow.") Uchida at [0068] ("In addition, in the actions in FIG. 19, "Forward" for forwarding the packet, "Drop" for dropping the packet, and "Modify-Field" for modifying information about the

No.	'111 Patent Claim 4	Uchida
		packet are present. An arbitrary number of actions can be registered in the actions (see the list in FIG. 21. However, in the case of Non-Patent Document 2, if the action field is blank, "Drop" is executed). For example, the Modify-Filed actions include: setting/changing of a VLAN ID or priority; stripping of a VLAN header; changing of an Ethernet (registered trademark) or IP source/destination address; and changing of a TCP/UDP source/destination port (other examples include changing of an IP ToS (Type of Service) field).") Uchida at [0070] ("The process rule search unit C14 determines actions (process contents) to be applied to the received packet, based on the information transmitted from the packet process information extraction unit C12 and outputs the actions to the packet process unit C15. More specifically, the process rule search unit C14 searches the process rule storage unit C13 for a process rule matching the information extracted from the received packet transmitted from the packet process information extraction unit C12 and outputs the process rule to the packet process unit C15. If a process rule matching the received packet exists, the process rule search unit C14 extracts actions (process contents) defined in the action field of the process rule and outputs the actions to the packet process unit C15. If no process rule matching the received packet is found, the process rule search unit C14 transmits the received packet or information extracted from the received packet to a predetermined control apparatus (controller; see FIG. 18) to request creation and transmission of a process rule. Upon receiving a process rule from the predetermined control apparatus (controller; see FIG. 18), the process rule search unit C14 registers the received process rule in the process rule storage unit C13. Next, the process rule search unit C14 extracts actions (process contents) from the process rule and outputs the actions to the packet process unit C15.") Uchida at [0071] ("The first process rule in the flow table in FIG. 20 is applied to packets indicating that the physical port number of the packet forwarding apparatus is 1; the source network is 192.168.0./24; the protocol type is Ox0006 (TCP); the desti-nation network is 192.168.1./24; and the TCP destination port number is 80. Accordingly, the actions (process contents) of changing the IP source address of the packet to 192.168.1/24 and transmitting the packet from physical port number 2 of the forwarding apparatus are executed. Similarly, the second process rule in FIG. 20 is applied to packets indicating a destination port number other than HTTP port numbers. Accordingly, the actions (process contents) of changing the IP source address of the packet to 192.168.1/24 and transmit-ting the packet from physical port number 3 of the forwarding apparatus are executed.")

No.	'111 Patent Claim 4	Uchida
		Uchida at [0083] ("Next, upon receiving the extracted information, the process rule search unit C14 searches the process rule storage unit C13 for a process rule matching the extracted informa-tion (step S104). If a process rule matching the extracted information is found, the process rule search unit C14 extracts actions (process contents) defined in the process rule and transmits the actions to the packet process unit C15 (step S105). If no process rule matching the extracted information is found, the process rule search unit C14 transmits the received packet or information extracted from the received packet to a predetermined control apparatus (controller; see FIG. 18) to request creation and transmission of a process rule (step S106). Upon receiving a process rule from the predeter-mined control apparatus (controller; see FIG. 18), the process rule search unit C14 registers the received process rule in the process rule storage unit C13, extracts actions (process con-tents) in the process rule, and outputs the actions to the packet process unit C15 (step S107).") Uchida at [0106] ("First, the computer D10 starts communication with the server D20. For an initial establishment of communication, a packet (SYN) is inputted to the packet forwarding apparatus C1 (start of ACTIVE OPEN through SYN forwarding in FIG. 6). The packet reception unit C10 receives and stores this first packet in the packet storage unit C11 (steps S101 to S102 in FIG. 3).") 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.") Uchida at [0109] ("Next, a process rule corresponding to the received packet is searched for. The packet process information extraction unit C12 transmits the extracted information (representing that the ingress port is 1; the source address is 192.168.0.10; the destination address is 192.168.1.10; and the protocol is TCP (the type is 0x0006)) to a process rule search unit C14. The process rule search unit C14 refers to the process rule storage unit C13. Since no process rule matching the packet exists, the process rule search unit C14 queries the control apparatus (controller) D1 (steps S104 and S106 in FIG. 3). The control apparatus (controller)

No.	'111 Patent Claim 4	Uchida
		D1 creates a process rule including process contents representing modification of the source IP and forward from a specified transport port and transmits the process rule to the packet forwarding apparatus C1. The process rule search unit C14 registers the process rule received from the control apparatus (controller) D1 as a new entry in the process rule storage unit C13 (step S107 in FIG. 3). At this stage, as illustrated in FIG. 9, a new process rule is added to the process rule storage unit C13.") Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Uchida 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. 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 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.") 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

No.	'111 Patent Claim 4	Uchida
		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.") 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.") 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

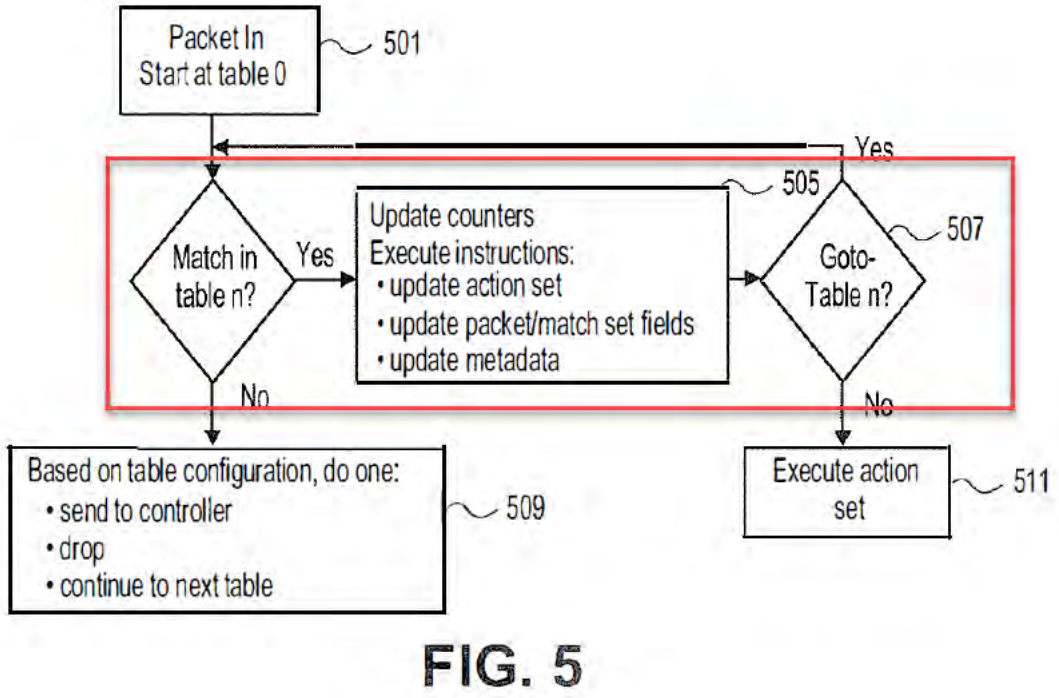
No.	'111 Patent Claim 4	Uchida
		traffic that benefits from optimization the most, thus achiev-ing both scalability and efficiency for optimization at a com-petitive price-point. The core element of the network control-ler 140 lies in its mechanisms for congestion detection and mitigation, which allows optimization resources to be utilized in the most efficient and surgical manner.") 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.") 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.") 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

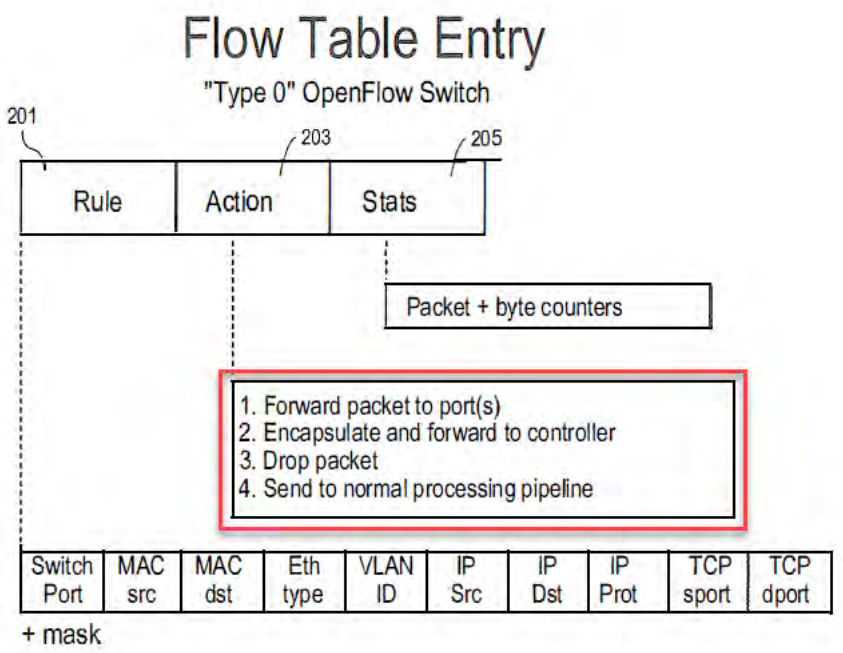
No.	'111 Patent Claim 4	Uchida
		flow inspection performed by the flow analyzer 312 may determine if the flow indeed contains large or medium object (e.g., larger than 50 kB), and/or if the source IP address of the flow is from a user or a group of users that are required to be monitored by policies. The flow ana-lyzer 312 may also determine if the flow needs to be opti-mized based on historical flow statistical data.") 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.") 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

No.	'111 Patent Claim 4	Uchida
		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.") 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.") 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 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 4	Uchida
		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.") 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.") 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.") 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.")

No.	'111 Patent Claim 4	Uchida
		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.") Kempf at [0106] ("This encapsulates the packet and sends it to the OpenFlow controller.") Kempf at Figure 5 (annotation added)

No.	'111 Patent Claim 4	Uchida
		 FIG. 5 Kempf at Figure 2 (annotation added)

No.	'111 Patent Claim 4	Uchida
		 FIG. 2
4[c]	responsive to receiving the packet, analyzing the packet, by the controller;	Uchida discloses responsive to receiving the packet, analyzing the packet, by the controller. For example, Uchida discloses the process rule search unit C14 transmits the received packet or information extracted from the received packet to a predetermined control apparatus to request creation and transmission of a process rule. Uchida at [0012] ("According to a first aspect of the present invention, there is provided a packet forwarding apparatus comprising: a process rule storage unit that stores process rules defining a process content for each flow; a packet process unit that executes processing

No.	'111 Patent Claim 4	Uchida
		received packet by using a process rule matching the received packet among the process rules and; an end determination information extraction unit that extracts information for checking a flow end from the received packet; and a flow end check unit that deletes, when a flow end is checked based on the extracted information, a process rule corresponding to the flow.") Uchida at [0014] ("According to a third aspect of the present invention, there is provided a process rule update method causing a packet forwarding apparatus comprising: a process rule stor-age unit that stores process rules defining a process content for each flow; and a packet process unit that executes and processing a received packet by using a process rule matching the received packet among the process rules to execute: extracting information for checking a flow received packet; and deleting, when a flow end is checked based on the extracted information, a process rule corre-sponding to the flow from the process rule storage unit. The present method is connected to a certain machine referred to as a packet forwarding apparatus comprising a process rule storage unit and a packet process unit.") Uchida at [0015] ("According to a fourth aspect of the present inven-tion, there is provided a program, causing a computer forming a packet forwarding apparatus comprising: a process rule storage unit that stores process rules defining a process con-tent for each flow; and a packet process unit that executes processing a received packet by using a process rule matching a received packet among the process rules to execute the processes of: extracting information for determining a flow end from the received packet; and deleting, when a flow end is determined based on the extracted information, a process rule corresponding to the flow from the process rule storage unit. The program can be stored in a computer-readable recording medium. Namely, the present invention can be realized as a computer program product. Also the program may be regarded as either a program product (i.e. manufac-ture) or a process embodying the load distribution method (process), expressed in a program language.") Uchida at [0039] ("First, an outline of an exemplary embodiment of the present invention will be described. Hereinafter, the reference characters used in this outline are used only as examples to facilitate comprehension and are not intended to limit the present invention to the illustrated modes. As illustrated in FIG. 1, a packet forwarding apparatus 1 according to an exemplary embodiment of the present invention includes: a process rule storage unit 11 storing process rules applied to received packets; a packet process unit 12 reading a process

No.	'111 Patent Claim 4	Uchida
		rule matching a received packet from the process rule storage unit to execute a forwarding process or the like; an end deter-mination information extraction unit 13 extracting informa-tion for checking a flow end from a received packet; and a flow end check unit 14 deleting, when a flow end is checked based on the extracted information, a process rule corre-sponding to the flow.") Uchida at [0068] ("In addition, in the actions in FIG. 19, "Forward" for forwarding the packet, "Drop" for dropping the packet, and "Modify-Field" for modifying information about the packet are present. An arbitrary number of actions can be registered in the actions (see the list in FIG. 21. However, in the case of Non-Patent Document 2, if the action field is blank, "Drop" is executed). For example, the Modify-Filed actions include: setting/changing of a VLAN ID or priority; stripping of a VLAN header; changing of an Ethernet (registered trademark) or IP source/destination address; and changing of a TCP/UDP source/destination port (other examples include changing of an IP ToS (Type of Service) field).") Uchida at [0070] ("The process rule search unit C14 determines actions (process contents) to be applied to the received packet, based on the information transmitted from the packet process information extraction unit C12 and outputs the actions to the packet process unit C15. More specifically, the process rule search unit C14 searches the process rule storage unit C13 for a process rule matching the information extracted from the received packet transmitted from the packet process information extraction unit C12 and outputs the process rule to the packet process unit C15. If a process rule matching the received packet exists, the process rule search unit C14 extracts actions (process contents) defined in the action field of the process rule and outputs the actions to the packet process unit C15. If no process rule matching the received packet is found, the process rule search unit C14 transmits the received packet or information extracted from the received packet to a predetermined control apparatus (controller; see FIG. 18) to request creation and transmission of a process rule. Upon receiving a process rule from the predetermined control apparatus (controller; see FIG. 18), the process rule search unit C14 registers the received process rule in the process rule storage unit C13. Next, the process rule search unit C14 extracts actions (process contents) from the process rule and outputs the actions to the packet process unit C15.")

No.	'111 Patent Claim 4	Uchida
		Uchida at [0071] ("The first process rule in the flow table in FIG. 20 is applied to packets indicating that the physical port number of the packet forwarding apparatus is 1; the source network is 192.168.0./24; the protocol type is Ox0006 (TCP); the desti-nation network is 192.168.1./24; and the TCP destination port number is 80. Accordingly, the actions (process contents) of changing the IP source address of the packet to 192.168.1/24 and transmitting the packet from physical port number 2 of the forwarding apparatus are executed. Similarly, the second process rule in FIG. 20 is applied to packets indicating a destination port number other than HTTP port numbers. Accordingly, the actions (process contents) of changing the IP source address of the packet to 192.168.1/24 and transmit-ting the packet from physical port number 3 of the forwarding apparatus are executed.") Uchida at [0083] ("Next, upon receiving the extracted information, the process rule search unit C14 searches the process rule storage unit C13 for a process rule matching the extracted informa-tion (step S104). If a process rule matching the extracted information is found, the process rule search unit C14 extracts actions (process contents) defined in the process rule and transmits the actions to the packet process unit C15 (step S105). If no process rule matching the extracted information is found, the process rule search unit C14 transmits the received packet or information extracted from the received packet to a predetermined control apparatus (controller; see FIG. 18) to request creation and transmission of a process rule (step S106). Upon receiving a process rule from the predeter-mined control apparatus (controller; see FIG. 18), the process rule search unit C14 registers the received process rule in the process rule storage unit C13, extracts actions (process con-tents) in the process rule, and outputs the actions to the packet process unit C15 (step S107).") Uchida at [0106] ("First, the computer D10 starts communication with the server D20. For an initial establishment of communication, a packet (SYN) is inputted to the packet forwarding apparatus C1 (start of ACTIVE OPEN through SYN forwarding in FIG. 6). The packet reception unit C10 receives and stores this first packet in the packet storage unit C11 (steps S101 to S102 in FIG. 3).") 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

No.	'111 Patent Claim 4	Uchida
		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.") Uchida at [0109] ("Next, a process rule corresponding to the received packet is searched for. The packet process information extraction unit C12 transmits the extracted information (representing that the ingress port is 1; the source address is 192.168.0.10; the destination address is 192.168.1.10; and the protocol is TCP (the type is 0x0006)) to a process rule search unit C14. The process rule search unit C14 refers to the process rule storage unit C13. Since no process rule matching the packet exists, the process rule search unit C14 queries the control apparatus (controller) D1 (steps S104 and S106 in FIG. 3). The control apparatus (controller) D1 creates a process rule including process contents representing modification of the source IP and forward from a specified transport port and transmits the process rule to the packet forwarding apparatus C1. The process rule search unit C14 registers the process rule received from the control apparatus (controller) D1 as a new entry in the process rule storage unit C13 (step S107 in FIG. 3). At this stage, as illustrated in FIG. 9, a new process rule is added to the process rule storage unit C13.")
4[d]	sending the packet, by the controller, to the network node; and	Uchida discloses sending the packet, by the controller, to the network node. For example, Uchida discloses the process rule search unit C14 transmits the received packet or information extracted from the received packet to a predetermined control apparatus to request creation and transmission of a process rule. Upon receiving a process rule from the predetermined control apparatus, the process rule search unit C14 registers the received process rule in the process rule storage unit C13. Next, the process rule search unit C14 extracts actions (process contents) from the process rule and outputs the actions to the packet process unit C15. Uchida at [0012] ("According to a first aspect of the present invention, there is provided a packet forwarding apparatus comprising: a process rule storage unit that stores process rules defining a process content for each flow; a packet process unit that executes processing a received packet by using a process rule matching the received packet among the process rules and; an end determination information extraction unit that extracts information for checking

No.	'111 Patent Claim 4	Uchida
		a flow end from the received packet; and a flow end check unit that deletes, when a flow end is checked based on the extracted information, a process rule corresponding to the flow.") Uchida at [0014] ("According to a third aspect of the present invention, there is provided a process rule update method causing a packet forwarding apparatus comprising: a process rule stor-age unit that stores process rules defining a process content for each flow; and a packet process unit that executes and processing a received packet by using a process rule matching the received packet among the process rules to execute: extracting information for checking a flow received packet; and deleting, when a flow end is checked based on the extracted information, a process rule corre-sponding to the flow from the process rule storage unit. The present method is connected to a certain machine referred to as a packet forwarding apparatus comprising a process rule storage unit and a packet process unit.") Uchida at [0015] ("According to a fourth aspect of the present inven-tion, there is provided a program, causing a computer forming a packet forwarding apparatus comprising: a process rule storage unit that stores process rules defining a process con-tent for each flow; and a packet process unit that executes processing a received packet by using a process rule matching a received packet among the process rules to execute the processes of: extracting information for determining a flow end from the received packet; and deleting, when a flow end is determined based on the extracted information, a process rule corresponding to the flow from the process rule storage unit. The program can be stored in a computer-readable recording medium. Namely, the present invention can be realized as a computer program product. Also the program may be regarded as either a program product (i.e. manufac-ture) or a process embodying the load distribution method (process), expressed in a program language.") Uchida at [0039] ("First, an outline of an exemplary embodiment of the present invention will be described. Hereinafter, the reference characters used in this outline are used only as examples to facilitate comprehension and are not intended to limit the present invention to the illustrated modes. As illustrated in FIG. 1, a packet forwarding apparatus 1 according to an exemplary embodiment of the present invention includes: a process rule storage unit 11 storing process rules applied to received packets; a packet process unit 12 reading a process rule matching a received packet from the process rule storage unit to execute a forwarding process or the like; an end deter-mination information extraction unit 13 extracting

No.	'111 Patent Claim 4	Uchida
		informa-tion for checking a flow end from a received packet; and a flow end check unit 14 deleting, when a flow end is checked based on the extracted information, a process rule corre-sponding to the flow.") Uchida at [0068] ("In addition, in the actions in FIG. 19, "Forward" for forwarding the packet, "Drop" for dropping the packet, and "Modify-Field" for modifying information about the packet are present. An arbitrary number of actions can be registered in the actions (see the list in FIG. 21. However, in the case of Non-Patent Document 2, if the action field is blank, "Drop" is executed). For example, the Modify-Filed actions include: setting/changing of a VLAN ID or priority; stripping of a VLAN header; changing of an Ethernet (registered trademark) or IP source/destination address; and changing of a TCP/UDP source/destination port (other examples include changing of an IP ToS (Type of Service) field).") Uchida at [0070] ("The process rule search unit C14 determines actions (process contents) to be applied to the received packet, based on the information transmitted from the packet process information extraction unit C12 and outputs the actions to the packet process unit C15. More specifically, the process rule search unit C14 searches the process rule storage unit C13 for a process rule matching the information extracted from the received packet transmitted from the packet process information extraction unit C12 and outputs the process rule to the packet process unit C15. If a process rule matching the received packet exists, the process rule search unit C14 extracts actions (process contents) defined in the action field of the process rule and outputs the actions to the packet process unit C15. If no process rule matching the received packet is found, the process rule search unit C14 transmits the received packet or information extracted from the received packet to a predetermined control apparatus (controller; see FIG. 18) to request creation and transmission of a process rule. Upon receiving a process rule from the predetermined control apparatus (controller; see FIG. 18), the process rule search unit C14 registers the received process rule in the process rule storage unit C13. Next, the process rule search unit C14 extracts actions (process contents) from the process rule and outputs the actions to the packet process unit C15.") Uchida at [0071] ("The first process rule in the flow table in FIG. 20 is applied to packets indicating that the physical port number of the packet forwarding apparatus is 1; the source network is 192.168.0./24; the protocol type is Ox0006 (TCP); the desti-nation network is

No.	'111 Patent Claim 4	Uchida
		192.168.1./24; and the TCP destination port number is 80. Accordingly, the actions (process contents) of changing the IP source address of the packet to 192.168.1/24 and transmitting the packet from physical port number 2 of the forwarding apparatus are executed. Similarly, the second process rule in FIG. 20 is applied to packets indicating a destination port number other than HTTP port numbers. Accordingly, the actions (process contents) of changing the IP source address of the packet to 192.168.1/24 and transmit-ting the packet from physical port number 3 of the forwarding apparatus are executed.") Uchida at [0083] ("Next, upon receiving the extracted information, the process rule search unit C14 searches the process rule storage unit C13 for a process rule matching the extracted informa-tion (step S104). If a process rule matching the extracted information is found, the process rule search unit C14 extracts actions (process contents) defined in the process rule and transmits the actions to the packet process unit C15 (step S105). If no process rule matching the extracted information is found, the process rule search unit C14 transmits the received packet or information extracted from the received packet to a predetermined control apparatus (controller; see FIG. 18) to request creation and transmission of a process rule (step S106). Upon receiving a process rule from the predeter-mined control apparatus (controller; see FIG. 18), the process rule search unit C14 registers the received process rule in the process rule storage unit C13, extracts actions (process con-tents) in the process rule, and outputs the actions to the packet process unit C15 (step S107).") Uchida at [0106] ("First, the computer D10 starts communication with the server D20. For an initial establishment of communication, a packet (SYN) is inputted to the packet forwarding apparatus C1 (start of ACTIVE OPEN through SYN forwarding in FIG. 6). The packet reception unit C10 receives and stores this first packet in the packet storage unit C11 (steps S101 to S102 in FIG. 3).") 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.")

No.	'111 Patent Claim 4	Uchida
		Uchida at [0109] ("Next, a process rule corresponding to the received packet is searched for. The packet process information extraction unit C12 transmits the extracted information (representing that the ingress port is 1; the source address is 192.168.0.10; the destination address is 192.168.1.10; and the protocol is TCP (the type is 0x0006)) to a process rule search unit C14. The process rule search unit C14 refers to the process rule storage unit C13. Since no process rule matching the packet exists, the process rule search unit C14 queries the control apparatus (controller) D1 (steps S104 and S106 in FIG. 3). The control apparatus (controller) D1 creates a process rule including process contents representing modification of the source IP and forward from a specified transport port and transmits the process rule to the packet forwarding apparatus C1. The process rule search unit C14 registers the process rule received from the control apparatus (controller) D1 as a new entry in the process rule storage unit C13 (step S107 in FIG. 3). At this stage, as illustrated in FIG. 9, a new process rule is added to the process rule storage unit C13.")
4[e]	responsive to receiving the packet, sending the packet, by the network node, to the second entity.	Uchida discloses responsive to receiving the packet, sending the packet, by the network node, to the second entity. For example, Uchida discloses 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. Uchida at [0012] ("According to a first aspect of the present invention, there is provided a packet forwarding apparatus comprising: a process rule storage unit that stores process rules defining a process content for each flow; a packet process unit that executes processing a received packet by using a process rule matching the received packet among the process rules and; an end determination information extraction unit that extracts information for checking a flow end from the received packet; and a flow end check unit that deletes, when a flow end is checked based on the extracted information, a process rule corresponding to the flow.") Uchida at [0014] ("According to a third aspect of the present invention, there is provided a process rule update method causing a packet forwarding apparatus comprising: a process rule stor-age unit that stores process rules defining a process content for each flow; and a packet

No.	'111 Patent Claim 4	Uchida
		process unit that executes and processing a received packet by using a process rule matching the received packet among the process rules to execute: extracting information for checking a flow received packet; and deleting, when a flow end is checked based on the extracted information, a process rule corre-sponding to the flow from the process rule storage unit. The present method is connected to a certain machine referred to as a packet forwarding apparatus comprising a process rule storage unit and a packet process unit.") Uchida at [0015] ("According to a fourth aspect of the present inven-tion, there is provided a program, causing a computer forming a packet forwarding apparatus comprising: a process rule storage unit that stores process rules defining a process con-tent for each flow; and a packet process unit that executes processing a received packet by using a process rule matching a received packet among the process rules to execute the processes of: extracting information for determining a flow end from the received packet; and deleting, when a flow end is determined based on the extracted information, a process rule corresponding to the flow from the process rule storage unit. The program can be stored in a computer-readable recording medium. Namely, the present invention can be realized as a computer program product. Also the program may be regarded as either a program product (i.e. manufac-ture) or a process embodying the load distribution method (process), expressed in a program language.") Uchida at [0039] ("First, an outline of an exemplary embodiment of the present invention will be described. Hereinafter, the reference characters used in this outline are used only as examples to facilitate comprehension and are not intended to limit the present invention to the illustrated modes. As illustrated in FIG. 1, a packet forwarding apparatus 1 according to an exemplary embodiment of the present invention includes: a process rule storage unit 11 storing process rules applied to received packets; a packet process unit 12 reading a process rule matching a received packet from the process rule storage unit to execute a forwarding process or the like; an end deter-mination information extraction unit 13 extracting informa-tion for checking a flow end from a received packet; and a flow end check unit 14 deleting, when a flow end is checked based on the extracted information, a process rule corre-sponding to the flow.") Uchida at [0068] ("In addition, in the actions in FIG. 19, "Forward" for forwarding the packet, "Drop" for dropping the packet, and "Modify-Field" for modifying information about the

No.	'111 Patent Claim 4	Uchida
		packet are present. An arbitrary number of actions can be registered in the actions (see the list in FIG. 21. However, in the case of Non-Patent Document 2, if the action field is blank, "Drop" is executed). For example, the Modify-Filed actions include: setting/changing of a VLAN ID or priority; stripping of a VLAN header; changing of an Ethernet (registered trademark) or IP source/destination address; and changing of a TCP/UDP source/destination port (other examples include changing of an IP ToS (Type of Service) field).") Uchida at [0070] ("The process rule search unit C14 determines actions (process contents) to be applied to the received packet, based on the information transmitted from the packet process information extraction unit C12 and outputs the actions to the packet process unit C15. More specifically, the process rule search unit C14 searches the process rule storage unit C13 for a process rule matching the information extracted from the received packet transmitted from the packet process information extraction unit C12 and outputs the process rule to the packet process unit C15. If a process rule matching the received packet exists, the process rule search unit C14 extracts actions (process contents) defined in the action field of the process rule and outputs the actions to the packet process unit C15. If no process rule matching the received packet is found, the process rule search unit C14 transmits the received packet or information extracted from the received packet to a predetermined control apparatus (controller; see FIG. 18) to request creation and transmission of a process rule. Upon receiving a process rule from the predetermined control apparatus (controller; see FIG. 18), the process rule search unit C14 registers the received process rule in the process rule storage unit C13. Next, the process rule search unit C14 extracts actions (process contents) from the process rule and outputs the actions to the packet process unit C15.") Uchida at [0071] ("The first process rule in the flow table in FIG. 20 is applied to packets indicating that the physical port number of the packet forwarding apparatus is 1; the source network is 192.168.0./24; the protocol type is Ox0006 (TCP); the desti-nation network is 192.168.1./24; and the TCP destination port number is 80. Accordingly, the actions (process contents) of changing the IP source address of the packet to 192.168.1/24 and transmitting the packet from physical port number 2 of the forwarding apparatus are executed. Similarly, the second process rule in FIG. 20 is applied to packets indicating a destination port number other than HTTP port numbers. Accordingly, the actions (process contents) of changing the IP source address of the packet to 192.168.1/24 and transmit-ting the packet from physical port number 3 of the forwarding apparatus are executed.")

No.	'111 Patent Claim 4	Uchida
		Uchida at [0083] ("Next, upon receiving the extracted information, the process rule search unit C14 searches the process rule storage unit C13 for a process rule matching the extracted informa-tion (step S104). If a process rule matching the extracted information is found, the process rule search unit C14 extracts actions (process contents) defined in the process rule and transmits the actions to the packet process unit C15 (step S105). If no process rule matching the extracted information is found, the process rule search unit C14 transmits the received packet or information extracted from the received packet to a predetermined control apparatus (controller; see FIG. 18) to request creation and transmission of a process rule (step S106). Upon receiving a process rule from the predeter-mined control apparatus (controller; see FIG. 18), the process rule search unit C14 registers the received process rule in the process rule storage unit C13, extracts actions (process con-tents) in the process rule, and outputs the actions to the packet process unit C15 (step S107).") Uchida at [0106] ("First, the computer D10 starts communication with the server D20. For an initial establishment of communication, a packet (SYN) is inputted to the packet forwarding apparatus C1 (start of ACTIVE OPEN through SYN forwarding in FIG. 6). The packet reception unit C10 receives and stores this first packet in the packet storage unit C11 (steps S101 to S102 in FIG. 3).") 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.") Uchida at [0109] ("Next, a process rule corresponding to the received packet is searched for. The packet process information extraction unit C12 transmits the extracted information (representing that the ingress port is 1; the source address is 192.168.0.10; the destination address is 192.168.1.10; and the protocol is TCP (the type is 0x0006)) to a process rule search unit C14. The process rule search unit C14 refers to the process rule storage unit C13. Since no process rule matching the packet exists, the process rule search unit C14 queries the control apparatus (controller) D1 (steps S104 and S106 in FIG. 3). The control apparatus (controller)

No.	'111 Patent Claim 4	Uchida
		D1 creates a process rule including process contents representing modification of the source IP and forward from a specified transport port and transmits the process rule to the packet forwarding apparatus C1. The process rule search unit C14 registers the process rule received from the control apparatus (controller) D1 as a new entry in the process rule storage unit C13 (step S107 in FIG. 3). At this stage, as illustrated in FIG. 9, a new process rule is added to the process rule storage unit C13.")

No.	'111 Patent Claim 5	Uchida
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.	Uchida 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. For example, Uchida discloses a process rule matching a received packet from the process rule storage unit to execute a forwarding process. If a process rule matching the received packet exists, the process rule search unit C14 extracts actions (process contents) defined in the action field of the process rule and outputs the actions to the packet process unit C15. A person of ordinary skill in the art would understand that a process rule matching could direct the packet to the controller for further processing. *See supra* at Claim 1. Uchida at [0012] ("According to a first aspect of the present invention, there is provided a packet forwarding apparatus comprising: a process rule storage unit that stores process rules defining a process content for each flow; a packet process unit that executes processing a received packet by using a process rule matching the received packet among the process rules and; an end determination information extraction unit that extracts information for checking a flow end from the received packet; and a flow end check unit that deletes, when a flow end is checked based on the extracted information, a process rule corresponding to the flow.") Uchida at [0014] ("According to a third aspect of the present invention, there is provided a process rule update method causing a packet forwarding apparatus comprising: a process rule stor-age unit that stores process rules defining a process content for each flow; and a packet process unit that executes and processing a received packet by using a process rule matching

No.	'111 Patent Claim 5	Uchida
		the received packet among the process rules to execute: extracting information for checking a flow received packet; and deleting, when a flow end is checked based on the extracted information, a process rule corre-sponding to the flow from the process rule storage unit. The present method is connected to a certain machine referred to as a packet forwarding apparatus comprising a process rule storage unit and a packet process unit.") Uchida at [0015] ("According to a fourth aspect of the present inven-tion, there is provided a program, causing a computer forming a packet forwarding apparatus comprising: a process rule storage unit that stores process rules defining a process con-tent for each flow; and a packet process unit that executes processing a received packet by using a process rule matching a received packet among the process rules to execute the processes of: extracting information for determining a flow end from the received packet; and deleting, when a flow end is determined based on the extracted information, a process rule corresponding to the flow from the process rule storage unit. The program can be stored in a computer-readable recording medium. Namely, the present invention can be realized as a computer program product. Also the program may be regarded as either a program product (i.e. manufac-ture) or a process embodying the load distribution method (process), expressed in a program language.") Uchida at [0039] ("First, an outline of an exemplary embodiment of the present invention will be described. Hereinafter, the reference characters used in this outline are used only as examples to facilitate comprehension and are not intended to limit the present invention to the illustrated modes. As illustrated in FIG. 1, a packet forwarding apparatus 1 according to an exemplary embodiment of the present invention includes: a process rule storage unit 11 storing process rules applied to received packets; a packet process unit 12 reading a process rule matching a received packet from the process rule storage unit to execute a forwarding process or the like; an end deter-mination information extraction unit 13 extracting informa-tion for checking a flow end from a received packet; and a flow end check unit 14 deleting, when a flow end is checked based on the extracted information, a process rule corre-sponding to the flow.") Uchida at [0068] ("In addition, in the actions in FIG. 19, "Forward" for forwarding the packet, "Drop" for dropping the packet, and "Modify-Field" for modifying information about the

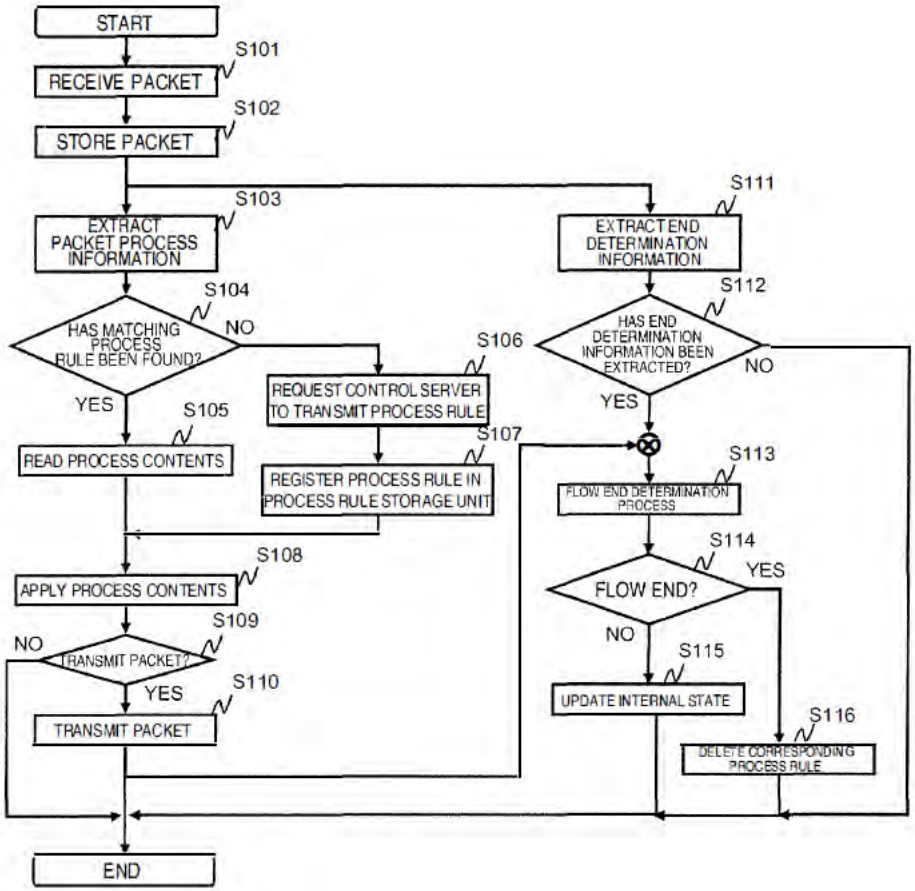
No.	'111 Patent Claim 5	Uchida
		packet are present. An arbitrary number of actions can be registered in the actions (see the list in FIG. 21. However, in the case of Non-Patent Document 2, if the action field is blank, "Drop" is executed). For example, the Modify-Filed actions include: setting/changing of a VLAN ID or priority; stripping of a VLAN header; changing of an Ethernet (registered trademark) or IP source/destination address; and changing of a TCP/UDP source/destination port (other examples include changing of an IP ToS (Type of Service) field).") Uchida at [0070] ("The process rule search unit C14 determines actions (process contents) to be applied to the received packet, based on the information transmitted from the packet process information extraction unit C12 and outputs the actions to the packet process unit C15. More specifically, the process rule search unit C14 searches the process rule storage unit C13 for a process rule matching the information extracted from the received packet transmitted from the packet process information extraction unit C12 and outputs the process rule to the packet process unit C15. If a process rule matching the received packet exists, the process rule search unit C14 extracts actions (process contents) defined in the action field of the process rule and outputs the actions to the packet process unit C15. If no process rule matching the received packet is found, the process rule search unit C14 transmits the received packet or information extracted from the received packet to a predetermined control apparatus (controller; see FIG. 18) to request creation and transmission of a process rule. Upon receiving a process rule from the predetermined control apparatus (controller; see FIG. 18), the process rule search unit C14 registers the received process rule in the process rule storage unit C13. Next, the process rule search unit C14 extracts actions (process contents) from the process rule and outputs the actions to the packet process unit C15.") Uchida at [0071] ("The first process rule in the flow table in FIG. 20 is applied to packets indicating that the physical port number of the packet forwarding apparatus is 1; the source network is 192.168.0./24; the protocol type is Ox0006 (TCP); the desti-nation network is 192.168.1./24; and the TCP destination port number is 80. Accordingly, the actions (process contents) of changing the IP source address of the packet to 192.168.1/24 and transmitting the packet from physical port number 2 of the forwarding apparatus are executed. Similarly, the second process rule in FIG. 20 is applied to packets indicating a destination port number other than HTTP port numbers. Accordingly, the actions (process contents) of changing the IP source address of the packet to 192.168.1/24 and transmit-ting the packet from physical port number 3 of the forwarding apparatus are executed.")

No.	'111 Patent Claim 5	Uchida
		Uchida at [0083] ("Next, upon receiving the extracted information, the process rule search unit C14 searches the process rule storage unit C13 for a process rule matching the extracted informa-tion (step S104). If a process rule matching the extracted information is found, the process rule search unit C14 extracts actions (process contents) defined in the process rule and transmits the actions to the packet process unit C15 (step S105). If no process rule matching the extracted information is found, the process rule search unit C14 transmits the received packet or information extracted from the received packet to a predetermined control apparatus (controller; see FIG. 18) to request creation and transmission of a process rule (step S106). Upon receiving a process rule from the predeter-mined control apparatus (controller; see FIG. 18), the process rule search unit C14 registers the received process rule in the process rule storage unit C13, extracts actions (process con-tents) in the process rule, and outputs the actions to the packet process unit C15 (step S107).") Uchida at [0106] ("First, the computer D10 starts communication with the server D20. For an initial establishment of communication, a packet (SYN) is inputted to the packet forwarding apparatus C1 (start of ACTIVE OPEN through SYN forwarding in FIG. 6). The packet reception unit C10 receives and stores this first packet in the packet storage unit C11 (steps S101 to S102 in FIG. 3).") 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.") Uchida at [0109] ("Next, a process rule corresponding to the received packet is searched for. The packet process information extraction unit C12 transmits the extracted information (representing that the ingress port is 1; the source address is 192.168.0.10; the destination address is 192.168.1.10; and the protocol is TCP (the type is 0x0006)) to a process rule search unit C14. The process rule search unit C14 refers to the process rule storage unit C13. Since no process rule matching the packet exists, the process rule search unit C14 queries the control apparatus (controller) D1 (steps S104 and S106 in FIG. 3). The control apparatus (controller)

No.	'111 Patent Claim 5	Uchida
		D1 creates a process rule including process contents representing modification of the source IP and forward from a specified transport port and transmits the process rule to the packet forwarding apparatus C1. The process rule search unit C14 registers the process rule received from the control apparatus (controller) D1 as a new entry in the process rule storage unit C13 (step S107 in FIG. 3). At this stage, as illustrated in FIG. 9, a new process rule is added to the process rule storage unit C13.")

No.	'111 Patent Claim 6	Uchida
6	The method according to claim 5, further comprising storing the received packet or a portion thereof, by the controller, in a memory.	Uchida discloses the method according to claim 5, further comprising storing the received packet or a portion thereof, by the controller, in a memory. For example, Uchida discloses the process rule search unit C14 transmits the received packet or information extracted from the received packet to a predetermined control apparatus. *See supra* at Claim 5. Uchida at [0070] ("The process rule search unit C14 determines actions (process contents) to be applied to the received packet, based on the information transmitted from the packet process information extraction unit C12 and outputs the actions to the packet process unit C15. More specifically, the process rule search unit C14 searches the process rule storage unit C13 for a process rule matching the information extracted from the received packet transmitted from the packet process information extraction unit C12 and outputs the process rule to the packet process unit C15. If a process rule matching the received packet exists, the process rule search unit C14 extracts actions (process contents) defined in the action field of the process rule and outputs the actions to the packet process unit C15. If no process rule matching the received packet is found, the process rule search unit C14 transmits the received packet or information extracted from the received packet to a predetermined control apparatus (controller; see FIG. 18) to request creation and transmission of a process rule. Upon receiving a process rule from the predetermined control apparatus (controller; see FIG. 18), the process rule search unit C14 registers the received process rule in the process rule storage unit C13.

No.	'111 Patent Claim 6	Uchida
		Next, the process rule search unit C14 extracts actions (process contents) from the process rule and outputs the actions to the packet process unit C15.") Uchida at [0106] ("First, the computer D10 starts communication with the server D20. For an initial establishment of communication, a packet (SYN) is inputted to the packet forwarding apparatus C1 (start of ACTIVE OPEN through SYN forwarding in FIG. 6). The packet reception unit C10 receives and stores this first packet in the packet storage unit C11 (steps S101 to S102 in FIG. 3).") 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.") Uchida at [0109] ("Next, a process rule corresponding to the received packet is searched for. The packet process information extraction unit C12 transmits the extracted information (representing that the ingress port is 1; the source address is 192.168.0.10; the destination address is 192.168.1.10; and the protocol is TCP (the type is 0x0006)) to a process rule search unit C14. The process rule search unit C14 refers to the process rule storage unit C13. Since no process rule matching the packet exists, the process rule search unit C14 queries the control apparatus (controller) D1 (steps S104 and S106 in FIG. 3). The control apparatus (controller) D1 creates a process rule including process contents representing modification of the source IP and forward from a specified transport port and transmits the process rule to the packet forwarding apparatus C1. The process rule search unit C14 registers the process rule received from the control apparatus (controller) D1 as a new entry in the process rule storage unit C13 (step S107 in FIG. 3). At this stage, as illustrated in FIG. 9, a new process rule is added to the process rule storage unit C13.") Uchida at Figure 3

No.	'111 Patent Claim 6	Uchida
		FIG. 3 

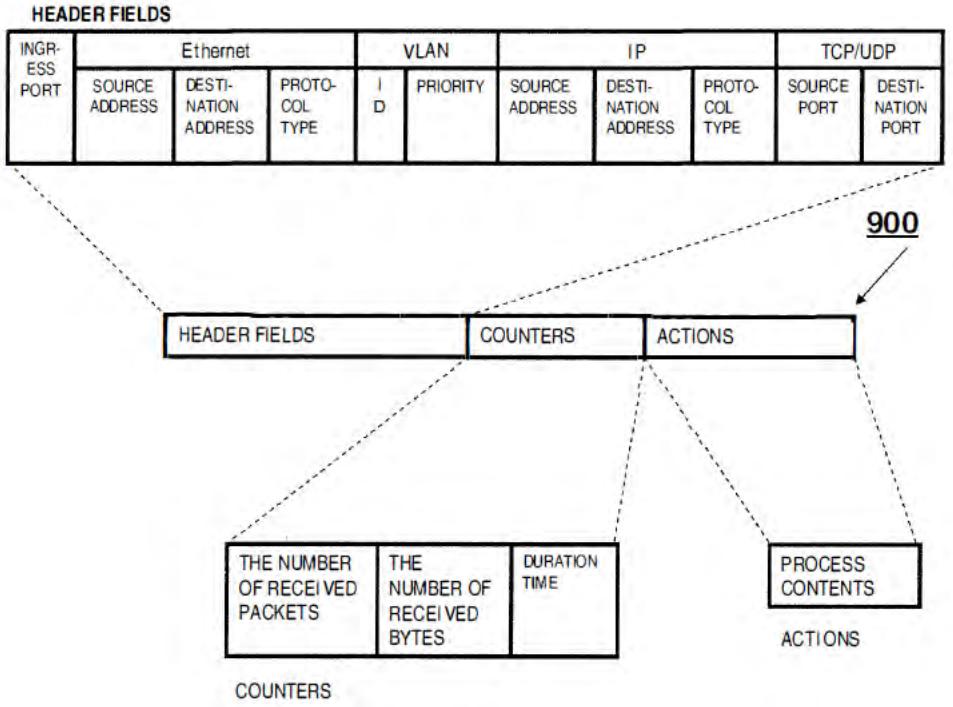
No.	'111 Patent Claim 7	Uchida
7	The method according to claim 5, further comprising responsive to the packet satisfying the criterion and to the instruction, sending a portion of the packet, by the network node, to the controller.	Uchida discloses the method according to claim 5, further comprising responsive to the packet satisfying the criterion and to the instruction, sending a portion of the packet, by the network node, to the controller. For example, Uchida discloses a process rule matching a received packet from the process rule storage unit to execute a forwarding process. If a process rule matching the received packet exists, the process rule search unit C14 extracts actions (process contents) defined in the action field of the process rule and outputs the actions to the packet process unit C15. A person of ordinary skill in the art would understand that a process rule matching could transmit the received packet or information extracted from the received packet to the controller for further processing. *See supra* at Claim 5. Uchida at [0012] ("According to a first aspect of the present invention, there is provided a packet forwarding apparatus comprising: a process rule storage unit that stores process rules defining a process content for each flow; a packet process unit that executes processing a received packet by using a process rule matching the received packet among the process rules and; an end determination information extraction unit that extracts information for checking a flow end from the received packet; and a flow end check unit that deletes, when a flow end is checked based on the extracted information, a process rule corresponding to the flow.") Uchida at [0014] ("According to a third aspect of the present invention, there is provided a process rule update method causing a packet forwarding apparatus comprising: a process rule stor-age unit that stores process rules defining a process content for each flow; and a packet process unit that executes and processing a received packet by using a process rule matching the received packet among the process rules to execute: extracting information for checking a flow received packet; and deleting, when a flow end is checked based on the extracted information, a process rule corre-sponding to the flow from the process rule storage unit. The present method is connected to a certain machine referred to as a packet forwarding apparatus comprising a process rule storage unit and a packet process unit.") Uchida at [0015] ("According to a fourth aspect of the present inven-tion, there is provided a program, causing a computer forming a packet forwarding apparatus comprising: a process

No.	'111 Patent Claim 7	Uchida
		rule storage unit that stores process rules defining a process con-tent for each flow; and a packet process unit that executes processing a received packet by using a process rule matching a received packet among the process rules to execute the processes of: extracting information for determining a flow end from the received packet; and deleting, when a flow end is determined based on the extracted information, a process rule corresponding to the flow from the process rule storage unit. The program can be stored in a computer-readable recording medium. Namely, the present invention can be realized as a computer program product. Also the program may be regarded as either a program product (i.e. manufac-ture) or a process embodying the load distribution method (process), expressed in a program language.") Uchida at [0039] ("First, an outline of an exemplary embodiment of the present invention will be described. Hereinafter, the reference characters used in this outline are used only as examples to facilitate comprehension and are not intended to limit the present invention to the illustrated modes. As illustrated in FIG. 1, a packet forwarding apparatus 1 according to an exemplary embodiment of the present invention includes: a process rule storage unit 11 storing process rules applied to received packets; a packet process unit 12 reading a process rule matching a received packet from the process rule storage unit to execute a forwarding process or the like; an end deter-mination information extraction unit 13 extracting informa-tion for checking a flow end from a received packet; and a flow end check unit 14 deleting, when a flow end is checked based on the extracted information, a process rule corre-sponding to the flow.") Uchida at [0068] ("In addition, in the actions in FIG. 19, "Forward" for forwarding the packet, "Drop" for dropping the packet, and "Modify-Field" for modifying information about the packet are present. An arbitrary number of actions can be registered in the actions (see the list in FIG. 21. However, in the case of Non-Patent Document 2, if the action field is blank, "Drop" is executed). For example, the Modify-Filed actions include: setting/changing of a VLAN ID or priority; stripping of a VLAN header; changing of an Ethernet (registered trademark) or IP source/destination address; and changing of a TCP/UDP source/destination port (other examples include changing of an IP ToS (Type of Service) field).")

No.	'111 Patent Claim 7	Uchida
		Uchida at [0070] ("The process rule search unit C14 determines actions (process contents) to be applied to the received packet, based on the information transmitted from the packet process information extraction unit C12 and outputs the actions to the packet process unit C15. More specifically, the process rule search unit C14 searches the process rule storage unit C13 for a process rule matching the information extracted from the received packet transmitted from the packet process information extraction unit C12 and outputs the process rule to the packet process unit C15. If a process rule matching the received packet exists, the process rule search unit C14 extracts actions (process contents) defined in the action field of the process rule and outputs the actions to the packet process unit C15. If no process rule matching the received packet is found, the process rule search unit C14 transmits the received packet or information extracted from the received packet to a predetermined control apparatus (controller; see FIG. 18) to request creation and transmission of a process rule. Upon receiving a process rule from the predetermined control apparatus (controller; see FIG. 18), the process rule search unit C14 registers the received process rule in the process rule storage unit C13. Next, the process rule search unit C14 extracts actions (process contents) from the process rule and outputs the actions to the packet process unit C15.") Uchida at [0071] ("The first process rule in the flow table in FIG. 20 is applied to packets indicating that the physical port number of the packet forwarding apparatus is 1; the source network is 192.168.0./24; the protocol type is Ox0006 (TCP); the desti-nation network is 192.168.1./24; and the TCP destination port number is 80. Accordingly, the actions (process contents) of changing the IP source address of the packet to 192.168.1/24 and transmitting the packet from physical port number 2 of the forwarding apparatus are executed. Similarly, the second process rule in FIG. 20 is applied to packets indicating a destination port number other than HTTP port numbers. Accordingly, the actions (process contents) of changing the IP source address of the packet to 192.168.1/24 and transmit-ting the packet from physical port number 3 of the forwarding apparatus are executed.") Uchida at [0083] ("Next, upon receiving the extracted information, the process rule search unit C14 searches the process rule storage unit C13 for a process rule matching the extracted informa-tion (step S104). If a process rule matching the extracted information is found, the process rule search unit C14 extracts actions (process contents) defined in the process rule and transmits the actions to the packet process unit C15 (step S105). If no process rule matching the extracted information is found, the process rule search unit C14 transmits the

No.	'111 Patent Claim 7	Uchida
		received packet or information extracted from the received packet to a predetermined control apparatus (controller; see FIG. 18) to request creation and transmission of a process rule (step S106). Upon receiving a process rule from the predeter-mined control apparatus (controller; see FIG. 18), the process rule search unit C14 registers the received process rule in the process rule storage unit C13, extracts actions (process con-tents) in the process rule, and outputs the actions to the packet process unit C15 (step S107).") Uchida at [0106] ("First, the computer D10 starts communication with the server D20. For an initial establishment of communication, a packet (SYN) is inputted to the packet forwarding apparatus C1 (start of ACTIVE OPEN through SYN forwarding in FIG. 6). The packet reception unit C10 receives and stores this first packet in the packet storage unit C11 (steps S101 to S102 in FIG. 3).") 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.") Uchida at [0109] ("Next, a process rule corresponding to the received packet is searched for. The packet process information extraction unit C12 transmits the extracted information (representing that the ingress port is 1; the source address is 192.168.0.10; the destination address is 192.168.1.10; and the protocol is TCP (the type is 0x0006)) to a process rule search unit C14. The process rule search unit C14 refers to the process rule storage unit C13. Since no process rule matching the packet exists, the process rule search unit C14 queries the control apparatus (controller) D1 (steps S104 and S106 in FIG. 3). The control apparatus (controller) D1 creates a process rule including process contents representing modification of the source IP and forward from a specified transport port and transmits the process rule to the packet forwarding apparatus C1. The process rule search unit C14 registers the process rule received from the control apparatus (controller) D1 as a new entry in the process rule storage unit C13 (step S107 in FIG. 3). At this stage, as illustrated in FIG. 9, a new process rule is added to the process rule storage unit C13.")

No.	'111 Patent Claim 8	Uchida
8[a]	The method according to claim 7, wherein the portion of the packet consists of multiple consecutive bytes, and	Uchida discloses the method according to claim 7, wherein the portion of the packet consists of multiple consecutive bytes. For example, Uchida discloses the number of received bytes which is used as statistics information about the flow to which the packet belongs. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Uchida 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. *See supra* at Claim 7. Uchida at [0068] ("For example, the header fields in FIG. 19 include: an ingress port of the packet forwarding apparatus; a source address, a destination address, and an upper protocol type of Ethernet (registered trademark); a VID and a priority (PCP; Priority Code Point) of an IEEE802.1Q-compliant VLAN; a source address, a destination address, and an upper protocol type of the IP; and a source port and a destination port of the TCP or the UDP. When a packet matching the contents of these header fields is received, process contents defined in the action field are executed. However, wildcards can be used in arbitrary information bits in the header fields.") Uchida at [0067] ("Further, the counters (fields) in FIG. 19 hold the number of received packets, the number of received bytes, and the rule duration time, which are used as statistics information about the flow to which the packet belongs.") Uchida at Figure 19

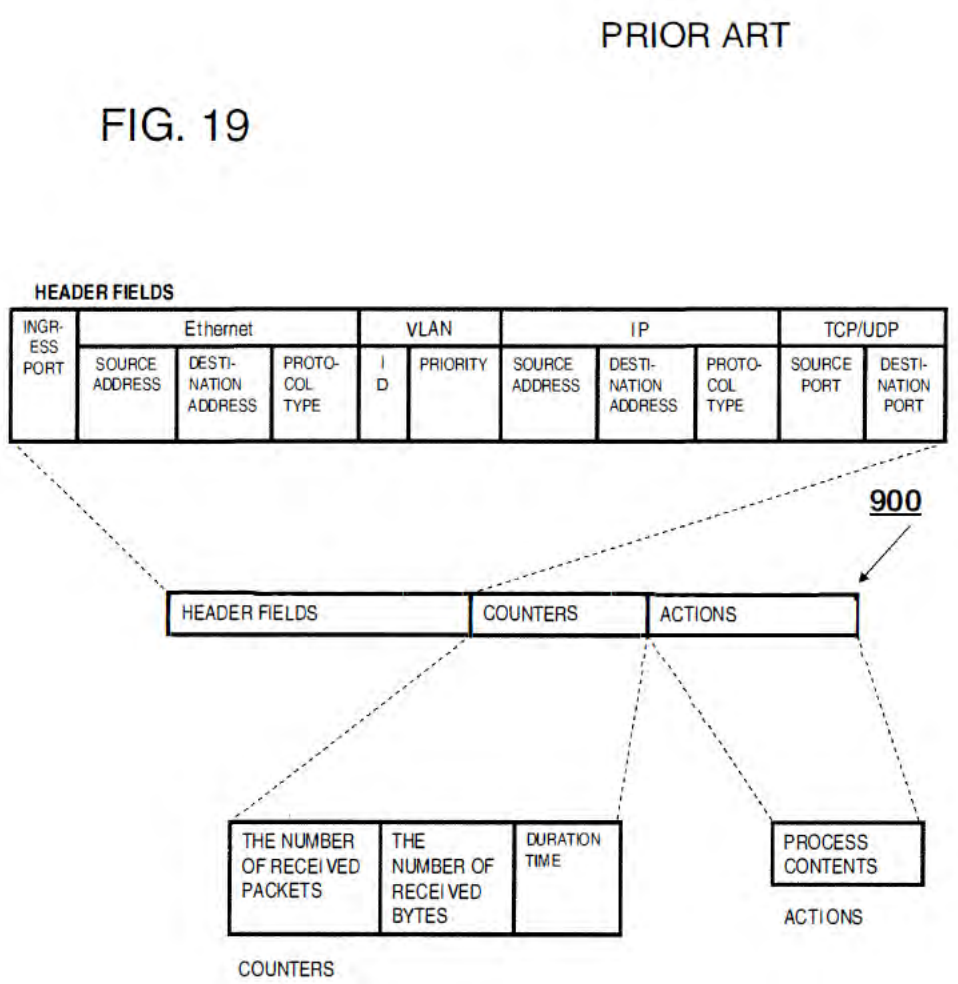
No.	'111 Patent Claim 8	Uchida
		PRIOR ART FIG. 19  Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Uchida in combination with (1) the knowledge of a person of ordinary skill in the art, alone or in further combination with (2) each (individually, as well as one 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.

No.	'111 Patent Claim 8	Uchida
		For example, Kempf discloses consecutive bytes of a packet header field. 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.") 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.") 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

No.	'111 Patent Claim 8	Uchida
		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.") 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.") 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.") 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

No.	'111 Patent Claim 8	Uchida
		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.") 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: Write-Metadata (GTP-TEID, 0x FFFFFFFF) Apply-Actions (Set-Output-Port LOCAL_GTP _U_ENCAP)") 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. 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.	Uchida discloses wherein the instruction comprises identification of the consecutive bytes in the packet. For example, Uchida discloses the number of received bytes which is used as statistics information about the flow to which the packet belongs. Thus, at least under the apparent

No.	'111 Patent Claim 8	Uchida
		claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Copeland 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. Uchida at [0066] ("For example, the header fields in FIG. 19 include: an ingress port of the packet forwarding apparatus; a source address, a destination address, and an upper protocol type of Ethernet (registered trademark); a VID and a priority (PCP; Priority Code Point) of an IEEE802.1Q-compliant VLAN; a source address, a destination address, and an upper protocol type of the IP; and a source port and a destination port of the TCP or the UDP. When a packet matching the contents of these header fields is received, process contents defined in the action field are executed. However, wildcards can be used in arbitrary information bits in the header fields.") Uchida at [0067] ("Further, the counters (fields) in FIG. 19 hold the number of received packets, the number of received bytes, and the rule duration time, which are used as statistics information about the flow to which the packet belongs.") Uchida at [0070] ("The process rule search unit C14 determines actions (process contents) to be applied to the received packet, based on the information transmitted from the packet process information extraction unit C12 and outputs the actions to the packet process unit C15. More specifically, the process rule search unit C14 searches the process rule storage unit C13 for a process rule matching the information extracted from the received packet transmitted from the packet process information extraction unit C12 and outputs the process rule to the packet process unit C15. If a process rule matching the received packet exists, the process rule search unit C14 extracts actions (process contents) defined in the action field of the process rule and outputs the actions to the packet process unit C15. If no process rule matching the received packet is found, the process rule search unit C14 transmits the received packet or information extracted from the received packet to a predetermined control apparatus (controller; see FIG. 18) to request creation and transmission of a process rule. Upon receiving a process rule from the predetermined control apparatus (controller; see FIG. 18), the process rule search unit C14 registers the received process rule in the process rule storage unit C13. Next, the process rule search unit C14 extracts actions (process contents) from the process rule and outputs the actions to the packet process unit C15.")

No.	'111 Patent Claim 8	Uchida
		Uchida at Figure 19  Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Uchida in combination with (1) the knowledge of a person of ordinary skill in the art, alone

No.	'111 Patent Claim 8	Uchida
		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. For example, Kempf discloses rules in which the flow table includes matching to the consecutive bytes of a packet header. 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.") 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.") 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

No.	'111 Patent Claim 8	Uchida
		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.") 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.") 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

No.	'111 Patent Claim 8	Uchida
		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.") 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.") 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: Write-Metadata (GTP-TEID, 0x FFFFFFFF) Apply-Actions (Set-Output-Port LOCAL_GTP _U_ENCAP)") For example, Copeland discloses identifying the sampled packet headers comprised of fragmented packets of smaller byte sizes. 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.")

No.	'111 Patent Claim 8	Uchida
		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.") 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	Uchida
9	The method according to claim 5, further comprising responsive to receiving the packet, analyzing the packet, by the controller.	Uchida discloses the method according to claim 5, further comprising responsive to receiving the packet, analyzing the packet, by the controller. For example, Uchida discloses the process rule search unit C14 transmits the received packet or information extracted from the received packet to a predetermined control apparatus which then uses the packet to create a process rule for transmission. *See supra* at Claim 5. Uchida at [0012] ("According to a first aspect of the present invention, there is provided a packet forwarding apparatus comprising: a process rule storage unit that stores process rules defining a process content for each flow; a packet process unit that executes processing a received packet by using a process rule matching the received packet among the process rules and; an end determination information extraction unit that extracts information for checking a flow end from the received packet; and a flow end check unit that deletes, when a flow end is checked based on the extracted information, a process rule corresponding to the flow.")

No.	'111 Patent Claim 9	Uchida
		Uchida at [0014] ("According to a third aspect of the present invention, there is provided a process rule update method causing a packet forwarding apparatus comprising: a process rule stor-age unit that stores process rules defining a process content for each flow; and a packet process unit that executes and processing a received packet by using a process rule matching the received packet among the process rules to execute: extracting information for checking a flow received packet; and deleting, when a flow end is checked based on the extracted information, a process rule corre-sponding to the flow from the process rule storage unit. The present method is connected to a certain machine referred to as a packet forwarding apparatus comprising a process rule storage unit and a packet process unit.") Uchida at [0015] ("According to a fourth aspect of the present inven-tion, there is provided a program, causing a computer forming a packet forwarding apparatus comprising: a process rule storage unit that stores process rules defining a process con-tent for each flow; and a packet process unit that executes processing a received packet by using a process rule matching a received packet among the process rules to execute the processes of: extracting information for determining a flow end from the received packet; and deleting, when a flow end is determined based on the extracted information, a process rule corresponding to the flow from the process rule storage unit. The program can be stored in a computer-readable recording medium. Namely, the present invention can be realized as a computer program product. Also the program may be regarded as either a program product (i.e. manufac-ture) or a process embodying the load distribution method (process), expressed in a program language.") Uchida at [0039] ("First, an outline of an exemplary embodiment of the present invention will be described. Hereinafter, the reference characters used in this outline are used only as examples to facilitate comprehension and are not intended to limit the present invention to the illustrated modes. As illustrated in FIG. 1, a packet forwarding apparatus 1 according to an exemplary embodiment of the present invention includes: a process rule storage unit 11 storing process rules applied to received packets; a packet process unit 12 reading a process rule matching a received packet from the process rule storage unit to execute a forwarding process or the like; an end deter-mination information extraction unit 13 extracting informa-tion for checking a flow end from a received packet; and a flow end check unit 14

No.	'111 Patent Claim 9	Uchida
		deleting, when a flow end is checked based on the extracted information, a process rule corre-sponding to the flow.") Uchida at [0068] ("In addition, in the actions in FIG. 19, "Forward" for forwarding the packet, "Drop" for dropping the packet, and "Modify-Field" for modifying information about the packet are present. An arbitrary number of actions can be registered in the actions (see the list in FIG. 21. However, in the case of Non-Patent Document 2, if the action field is blank, "Drop" is executed). For example, the Modify-Filed actions include: setting/changing of a VLAN ID or priority; stripping of a VLAN header; changing of an Ethernet (registered trademark) or IP source/destination address; and changing of a TCP/UDP source/destination port (other examples include changing of an IP ToS (Type of Service) field).") Uchida at [0070] ("The process rule search unit C14 determines actions (process contents) to be applied to the received packet, based on the information transmitted from the packet process information extraction unit C12 and outputs the actions to the packet process unit C15. More specifically, the process rule search unit C14 searches the process rule storage unit C13 for a process rule matching the information extracted from the received packet transmitted from the packet process information extraction unit C12 and outputs the process rule to the packet process unit C15. If a process rule matching the received packet exists, the process rule search unit C14 extracts actions (process contents) defined in the action field of the process rule and outputs the actions to the packet process unit C15. If no process rule matching the received packet is found, the process rule search unit C14 transmits the received packet or information extracted from the received packet to a predetermined control apparatus (controller; see FIG. 18) to request creation and transmission of a process rule. Upon receiving a process rule from the predetermined control apparatus (controller; see FIG. 18), the process rule search unit C14 registers the received process rule in the process rule storage unit C13. Next, the process rule search unit C14 extracts actions (process contents) from the process rule and outputs the actions to the packet process unit C15.") Uchida at [0071] ("The first process rule in the flow table in FIG. 20 is applied to packets indicating that the physical port number of the packet forwarding apparatus is 1; the source network is 192.168.0./24; the protocol type is Ox0006 (TCP); the desti-nation network is 192.168.1./24; and the TCP destination port number is 80. Accordingly, the actions (process

No.	'111 Patent Claim 9	Uchida
		contents) of changing the IP source address of the packet to 192.168.1/24 and transmitting the packet from physical port number 2 of the forwarding apparatus are executed. Similarly, the second process rule in FIG. 20 is applied to packets indicating a destination port number other than HTTP port numbers. Accordingly, the actions (process contents) of changing the IP source address of the packet to 192.168.1/24 and transmit-ting the packet from physical port number 3 of the forwarding apparatus are executed.") Uchida at [0083] ("Next, upon receiving the extracted information, the process rule search unit C14 searches the process rule storage unit C13 for a process rule matching the extracted informa-tion (step S104). If a process rule matching the extracted information is found, the process rule search unit C14 extracts actions (process contents) defined in the process rule and transmits the actions to the packet process unit C15 (step S105). If no process rule matching the extracted information is found, the process rule search unit C14 transmits the received packet or information extracted from the received packet to a predetermined control apparatus (controller; see FIG. 18) to request creation and transmission of a process rule (step S106). Upon receiving a process rule from the predeter-mined control apparatus (controller; see FIG. 18), the process rule search unit C14 registers the received process rule in the process rule storage unit C13, extracts actions (process con-tents) in the process rule, and outputs the actions to the packet process unit C15 (step S107).") Uchida at [0106] ("First, the computer D10 starts communication with the server D20. For an initial establishment of communication, a packet (SYN) is inputted to the packet forwarding apparatus C1 (start of ACTIVE OPEN through SYN forwarding in FIG. 6). The packet reception unit C10 receives and stores this first packet in the packet storage unit C11 (steps S101 to S102 in FIG. 3).") 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.")

No.	'111 Patent Claim 9	Uchida
		Uchida at [0109] ("Next, a process rule corresponding to the received packet is searched for. The packet process information extraction unit C12 transmits the extracted information (representing that the ingress port is 1; the source address is 192.168.0.10; the destination address is 192.168.1.10; and the protocol is TCP (the type is 0x0006)) to a process rule search unit C14. The process rule search unit C14 refers to the process rule storage unit C13. Since no process rule matching the packet exists, the process rule search unit C14 queries the control apparatus (controller) D1 (steps S104 and S106 in FIG. 3). The control apparatus (controller) D1 creates a process rule including process contents representing modification of the source IP and forward from a specified transport port and transmits the process rule to the packet forwarding apparatus C1. The process rule search unit C14 registers the process rule received from the control apparatus (controller) D1 as a new entry in the process rule storage unit C13 (step S107 in FIG. 3). At this stage, as illustrated in FIG. 9, a new process rule is added to the process rule storage unit C13.")

No.	'111 Patent Claim 12	Uchida
12	The method according to claim 9, wherein the analyzing comprises applying security or data analytic application.	Uchida discloses the method according to claim 9, wherein the analyzing comprises applying security or data analytic application. For example, Uchida discloses the process rule search unit C14 transmits the received packet or information extracted from the received packet to a predetermined control apparatus which then uses the packet to create a process rule for transmission. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Copeland is found to not meet this limitation, wherein the analyzing comprises applying security or data analytic application would have been obvious to a person having ordinary skill in the art, as explained below. *See supra* at Claim 9. Uchida at [0012] ("According to a first aspect of the present invention, there is provided a packet forwarding apparatus comprising: a process rule storage unit that stores process rules defining a process content for each flow; a packet process unit that executes processing a received packet by using a process rule matching the received packet among the process rules and; an end determination information extraction unit that extracts information for checking

No.	'111 Patent Claim 12	Uchida
		a flow end from the received packet; and a flow end check unit that deletes, when a flow end is checked based on the extracted information, a process rule corresponding to the flow.") Uchida at [0014] ("According to a third aspect of the present invention, there is provided a process rule update method causing a packet forwarding apparatus comprising: a process rule stor-age unit that stores process rules defining a process content for each flow; and a packet process unit that executes and processing a received packet by using a process rule matching the received packet among the process rules to execute: extracting information for checking a flow received packet; and deleting, when a flow end is checked based on the extracted information, a process rule corre-sponding to the flow from the process rule storage unit. The present method is connected to a certain machine referred to as a packet forwarding apparatus comprising a process rule storage unit and a packet process unit.") Uchida at [0015] ("According to a fourth aspect of the present inven-tion, there is provided a program, causing a computer forming a packet forwarding apparatus comprising: a process rule storage unit that stores process rules defining a process con-tent for each flow; and a packet process unit that executes processing a received packet by using a process rule matching a received packet among the process rules to execute the processes of: extracting information for determining a flow end from the received packet; and deleting, when a flow end is determined based on the extracted information, a process rule corresponding to the flow from the process rule storage unit. The program can be stored in a computer-readable recording medium. Namely, the present invention can be realized as a computer program product. Also the program may be regarded as either a program product (i.e. manufac-ture) or a process embodying the load distribution method (process), expressed in a program language.") Uchida at [0039] ("First, an outline of an exemplary embodiment of the present invention will be described. Hereinafter, the reference characters used in this outline are used only as examples to facilitate comprehension and are not intended to limit the present invention to the illustrated modes. As illustrated in FIG. 1, a packet forwarding apparatus 1 according to an exemplary embodiment of the present invention includes: a process rule storage unit 11 storing process rules applied to received packets; a packet process unit 12 reading a process rule matching a received packet from the process rule storage unit to execute a forwarding process or the like; an end deter-mination information extraction unit 13 extracting

No.	'111 Patent Claim 12	Uchida
		informa-tion for checking a flow end from a received packet; and a flow end check unit 14 deleting, when a flow end is checked based on the extracted information, a process rule corre-sponding to the flow.") Uchida at [0068] ("In addition, in the actions in FIG. 19, "Forward" for forwarding the packet, "Drop" for dropping the packet, and "Modify-Field" for modifying information about the packet are present. An arbitrary number of actions can be registered in the actions (see the list in FIG. 21. However, in the case of Non-Patent Document 2, if the action field is blank, "Drop" is executed). For example, the Modify-Filed actions include: setting/changing of a VLAN ID or priority; stripping of a VLAN header; changing of an Ethernet (registered trademark) or IP source/destination address; and changing of a TCP/UDP source/destination port (other examples include changing of an IP ToS (Type of Service) field).") Uchida at [0070] ("The process rule search unit C14 determines actions (process contents) to be applied to the received packet, based on the information transmitted from the packet process information extraction unit C12 and outputs the actions to the packet process unit C15. More specifically, the process rule search unit C14 searches the process rule storage unit C13 for a process rule matching the information extracted from the received packet transmitted from the packet process information extraction unit C12 and outputs the process rule to the packet process unit C15. If a process rule matching the received packet exists, the process rule search unit C14 extracts actions (process contents) defined in the action field of the process rule and outputs the actions to the packet process unit C15. If no process rule matching the received packet is found, the process rule search unit C14 transmits the received packet or information extracted from the received packet to a predetermined control apparatus (controller; see FIG. 18) to request creation and transmission of a process rule. Upon receiving a process rule from the predetermined control apparatus (controller; see FIG. 18), the process rule search unit C14 registers the received process rule in the process rule storage unit C13. Next, the process rule search unit C14 extracts actions (process contents) from the process rule and outputs the actions to the packet process unit C15.") Uchida at [0071] ("The first process rule in the flow table in FIG. 20 is applied to packets indicating that the physical port number of the packet forwarding apparatus is 1; the source network is 192.168.0./24; the protocol type is Ox0006 (TCP); the desti-nation network is

No.	'111 Patent Claim 12	Uchida
		192.168.1./24; and the TCP destination port number is 80. Accordingly, the actions (process contents) of changing the IP source address of the packet to 192.168.1/24 and transmitting the packet from physical port number 2 of the forwarding apparatus are executed. Similarly, the second process rule in FIG. 20 is applied to packets indicating a destination port number other than HTTP port numbers. Accordingly, the actions (process contents) of changing the IP source address of the packet to 192.168.1/24 and transmit-ting the packet from physical port number 3 of the forwarding apparatus are executed.") Uchida at [0083] ("Next, upon receiving the extracted information, the process rule search unit C14 searches the process rule storage unit C13 for a process rule matching the extracted informa-tion (step S104). If a process rule matching the extracted information is found, the process rule search unit C14 extracts actions (process contents) defined in the process rule and transmits the actions to the packet process unit C15 (step S105). If no process rule matching the extracted information is found, the process rule search unit C14 transmits the received packet or information extracted from the received packet to a predetermined control apparatus (controller; see FIG. 18) to request creation and transmission of a process rule (step S106). Upon receiving a process rule from the predeter-mined control apparatus (controller; see FIG. 18), the process rule search unit C14 registers the received process rule in the process rule storage unit C13, extracts actions (process con-tents) in the process rule, and outputs the actions to the packet process unit C15 (step S107).") Uchida at [0106] ("First, the computer D10 starts communication with the server D20. For an initial establishment of communication, a packet (SYN) is inputted to the packet forwarding apparatus C1 (start of ACTIVE OPEN through SYN forwarding in FIG. 6). The packet reception unit C10 receives and stores this first packet in the packet storage unit C11 (steps S101 to S102 in FIG. 3).") 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.")

No.	'111 Patent Claim 12	Uchida
		Uchida at [0109] ("Next, a process rule corresponding to the received packet is searched for. The packet process information extraction unit C12 transmits the extracted information (representing that the ingress port is 1; the source address is 192.168.0.10; the destination address is 192.168.1.10; and the protocol is TCP (the type is 0x0006)) to a process rule search unit C14. The process rule search unit C14 refers to the process rule storage unit C13. Since no process rule matching the packet exists, the process rule search unit C14 queries the control apparatus (controller) D1 (steps S104 and S106 in FIG. 3). The control apparatus (controller) D1 creates a process rule including process contents representing modification of the source IP and forward from a specified transport port and transmits the process rule to the packet forwarding apparatus C1. The process rule search unit C14 registers the process rule received from the control apparatus (controller) D1 as a new entry in the process rule storage unit C13 (step S107 in FIG. 3). At this stage, as illustrated in FIG. 9, a new process rule is added to the process rule storage unit C13.") Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Uchida 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 12 of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references. For example, Copeland discloses analysis by the intrusion detection engine of the monitoring appliance to include security analytics and determining suspicious activity. Copeland at [0022] ("This novel detection system does not require an updated database of signatures. Instead, the intrusion detec-tion system inspects all inbound and outbound activity and identifies suspicious patterns that denote non-normal flows and may indicate an attack. The computational simplicity of the technique allows for operation at much higher speeds than is possible with a signature-based system on compa-rable hardware. The ability of the system to operate with sampled data provides the ability to operate at much higher network speeds and cover a wide network area with a high degree of accuracy and with more detailed information than would otherwise be possible.") Copeland at [0026] ("This novel detection system does not require an updated database of signatures. Instead, the intrusion detec-tion system inspects all inbound and outbound activity

No.	'111 Patent Claim 12	Uchida
		and identifies suspicious patterns that denote non-normal flows and may indicate an attack. The computational simplicity of the technique allows for operation at much higher speeds than is possible with a signature-based system on compa-rable hardware. The ability of the system to operate with sampled data provides the ability to operate at much higher network speeds and cover a wide network area with a high degree of accuracy and with more detailed information than would otherwise be possible.") 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.") Copeland at [0143] ("For security monitoring, a single event multiplied by a sample rate should not be allowed to dominate a calculation. In the above example, a host sending a single SYN packet that is sampled at 1/10,000 would be inaccurately identified as generating a SYN flood if we sampled a single packet containing the SYN flag and multiplied it by 10,000. As such, multiple samples must be obtained for threshold based event alarms such as this to add statistical significance to the data. Common approaches for this comprise requiring an absolute minimum number of incidents via sampling (e.g., 5). A second approach would be based on an algorithm that determines a minimum number of samples that would be required based on the sample rate and the threshold value in order to maintain a good balance of sensitivity and accuracy.") 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

No.	'111 Patent Claim 12	Uchida
		to drop packets going to or from an offending host. It will thus be 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.") For example, Chua '877 discloses analyzing, including determining trusted packet flows for security purposes, performed by the controller on packets received. 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.") 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. 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.") 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

No.	'111 Patent Claim 12	Uchida
		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.") 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 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.") 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-

No.	'111 Patent Claim 12	Uchida
		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.") 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.") 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 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.") 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

No.	'111 Patent Claim 12	Uchida
		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).") 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.") 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.")

No.	'111 Patent Claim 13	Uchida
13	The method according to claim 9, wherein the analyzing comprises applying security application that comprises firewall or	Uchida discloses the method according to claim 9, wherein the analyzing comprises applying security application that comprises firewall or intrusion detection functionality. For example, Uchida discloses the process rule search unit C14 transmits the received packet or information extracted from the received packet to a predetermined control apparatus which then uses the packet to create a process rule for transmission. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent

No.	'111 Patent Claim 13	Uchida
	intrusion detection functionality.	that the Copeland 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. *See supra* at Claim 9. Uchida at [0012] ("According to a first aspect of the present invention, there is provided a packet forwarding apparatus comprising: a process rule storage unit that stores process rules defining a process content for each flow; a packet process unit that executes processing a received packet by using a process rule matching the received packet among the process rules and; an end determination information extraction unit that extracts information for checking a flow end from the received packet; and a flow end check unit that deletes, when a flow end is checked based on the extracted information, a process rule corresponding to the flow.") Uchida at [0014] ("According to a third aspect of the present invention, there is provided a process rule update method causing a packet forwarding apparatus comprising: a process rule stor-age unit that stores process rules defining a process content for each flow; and a packet process unit that executes and processing a received packet by using a process rule matching the received packet among the process rules to execute: extracting information for checking a flow received packet; and deleting, when a flow end is checked based on the extracted information, a process rule corre-sponding to the flow from the process rule storage unit. The present method is connected to a certain machine referred to as a packet forwarding apparatus comprising a process rule storage unit and a packet process unit.") Uchida at [0015] ("According to a fourth aspect of the present inven-tion, there is provided a program, causing a computer forming a packet forwarding apparatus comprising: a process rule storage unit that stores process rules defining a process con-tent for each flow; and a packet process unit that executes processing a received packet by using a process rule matching a received packet among the process rules to execute the processes of: extracting information for determining a flow end from the received packet; and deleting, when a flow end is determined based on the extracted information, a process rule corresponding to the flow from the process rule storage unit. The program can be stored in a computer-readable recording medium. Namely, the present invention can be realized as a computer program product. Also the program may be regarded as either a program product (i.e. manufac-ture)

No.	'111 Patent Claim 13	Uchida
		or a process embodying the load distribution method (process), expressed in a program language.") Uchida at [0039] ("First, an outline of an exemplary embodiment of the present invention will be described. Hereinafter, the reference characters used in this outline are used only as examples to facilitate comprehension and are not intended to limit the present invention to the illustrated modes. As illustrated in FIG. 1, a packet forwarding apparatus 1 according to an exemplary embodiment of the present invention includes: a process rule storage unit 11 storing process rules applied to received packets; a packet process unit 12 reading a process rule matching a received packet from the process rule storage unit to execute a forwarding process or the like; an end deter-mination information extraction unit 13 extracting informa-tion for checking a flow end from a received packet; and a flow end check unit 14 deleting, when a flow end is checked based on the extracted information, a process rule corre-sponding to the flow.") Uchida at [0068] ("In addition, in the actions in FIG. 19, "Forward" for forwarding the packet, "Drop" for dropping the packet, and "Modify-Field" for modifying information about the packet are present. An arbitrary number of actions can be registered in the actions (see the list in FIG. 21. However, in the case of Non-Patent Document 2, if the action field is blank, "Drop" is executed). For example, the Modify-Filed actions include: setting/changing of a VLAN ID or priority; stripping of a VLAN header; changing of an Ethernet (registered trademark) or IP source/destination address; and changing of a TCP/UDP source/destination port (other examples include changing of an IP ToS (Type of Service) field).") Uchida at [0070] ("The process rule search unit C14 determines actions (process contents) to be applied to the received packet, based on the information transmitted from the packet process information extraction unit C12 and outputs the actions to the packet process unit C15. More specifically, the process rule search unit C14 searches the process rule storage unit C13 for a process rule matching the information extracted from the received packet transmitted from the packet process information extraction unit C12 and outputs the process rule to the packet process unit C15. If a process rule matching the received packet exists, the process rule search unit C14 extracts actions (process contents) defined in the action field of the process rule and outputs the actions to the packet process unit C15. If no process rule

No.	'111 Patent Claim 13	Uchida
		matching the received packet is found, the process rule search unit C14 transmits the received packet or information extracted from the received packet to a predetermined control apparatus (controller; see FIG. 18) to request creation and transmission of a process rule. Upon receiving a process rule from the predetermined control apparatus (controller; see FIG. 18), the process rule search unit C14 registers the received process rule in the process rule storage unit C13. Next, the process rule search unit C14 extracts actions (process contents) from the process rule and outputs the actions to the packet process unit C15.") Uchida at [0071] ("The first process rule in the flow table in FIG. 20 is applied to packets indicating that the physical port number of the packet forwarding apparatus is 1; the source network is 192.168.0./24; the protocol type is Ox0006 (TCP); the desti-nation network is 192.168.1./24; and the TCP destination port number is 80. Accordingly, the actions (process contents) of changing the IP source address of the packet to 192.168.1/24 and transmitting the packet from physical port number 2 of the forwarding apparatus are executed. Similarly, the second process rule in FIG. 20 is applied to packets indicating a destination port number other than HTTP port numbers. Accordingly, the actions (process contents) of changing the IP source address of the packet to 192.168.1/24 and transmit-ting the packet from physical port number 3 of the forwarding apparatus are executed.") Uchida at [0083] ("Next, upon receiving the extracted information, the process rule search unit C14 searches the process rule storage unit C13 for a process rule matching the extracted informa-tion (step S104). If a process rule matching the extracted information is found, the process rule search unit C14 extracts actions (process contents) defined in the process rule and transmits the actions to the packet process unit C15 (step S105). If no process rule matching the extracted information is found, the process rule search unit C14 transmits the received packet or information extracted from the received packet to a predetermined control apparatus (controller; see FIG. 18) to request creation and transmission of a process rule (step S106). Upon receiving a process rule from the predeter-mined control apparatus (controller; see FIG. 18), the process rule search unit C14 registers the received process rule in the process rule storage unit C13, extracts actions (process con-tents) in the process rule, and outputs the actions to the packet process unit C15 (step S107).") Uchida at [0106] ("First, the computer D10 starts communication with the server D20. For an initial establishment of communication, a packet (SYN) is inputted to the packet forwarding

No.	'111 Patent Claim 13	Uchida
		apparatus C1 (start of ACTIVE OPEN through SYN forwarding in FIG. 6). The packet reception unit C10 receives and stores this first packet in the packet storage unit C11 (steps S101 to S102 in FIG. 3).") 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.") Uchida at [0109] ("Next, a process rule corresponding to the received packet is searched for. The packet process information extraction unit C12 transmits the extracted information (representing that the ingress port is 1; the source address is 192.168.0.10; the destination address is 192.168.1.10; and the protocol is TCP (the type is 0x0006)) to a process rule search unit C14. The process rule search unit C14 refers to the process rule storage unit C13. Since no process rule matching the packet exists, the process rule search unit C14 queries the control apparatus (controller) D1 (steps S104 and S106 in FIG. 3). The control apparatus (controller) D1 creates a process rule including process contents representing modification of the source IP and forward from a specified transport port and transmits the process rule to the packet forwarding apparatus C1. The process rule search unit C14 registers the process rule received from the control apparatus (controller) D1 as a new entry in the process rule storage unit C13 (step S107 in FIG. 3). At this stage, as illustrated in FIG. 9, a new process rule is added to the process rule storage unit C13.") Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Uchida 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. 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.

No.	'111 Patent Claim 13	Uchida
		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.") 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.") 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.") 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

No.	'111 Patent Claim 13	Uchida
		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.") 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.") Copeland at Figure 5

No.	'111 Patent Claim 13	Uchida

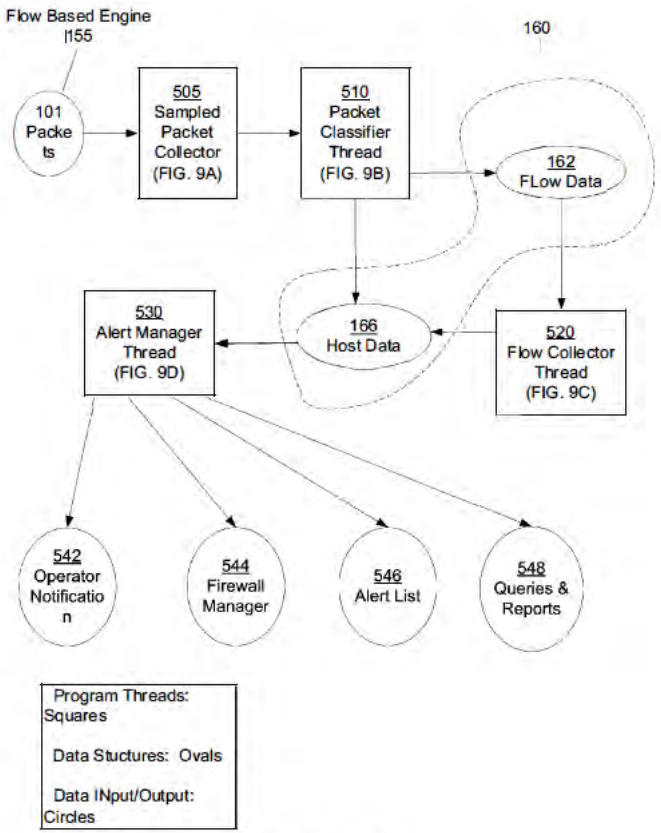


FIG. 5

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 appreciated that the present

No.	'111 Patent Claim 13	Uchida
		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.") 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.") For example, Chua '877 discloses security determinations and analysis that involve the use of firewalls or intrusion detection services. 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.") 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. 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

No.	'111 Patent Claim 13	Uchida
		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.") 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.") 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 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,

No.	'111 Patent Claim 13	Uchida
		domain name service (DNS) and global DNS server devices, mail proxies, and anti-spam devices.") 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.") 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.") 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 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

No.	'111 Patent Claim 13	Uchida
		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.") 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).") 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.") 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,

No.	'111 Patent Claim 13	Uchida
		blocking corresponding traf-fic, mirroring corresponding traffic, redirecting correspond-ing traffic.") 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	Uchida
14	The method according to claim 9, wherein the analyzing comprises performing Deep Packet Inspection (DPI) or using a DPI engine on the packet.	Uchida discloses the method according to claim 9, wherein the analyzing comprises performing Deep Packet Inspection (DPI) or using a DPI engine on the packet. For example, Uchida discloses the process rule search unit C14 transmits the received packet or information extracted from the received packet to a predetermined control apparatus which then uses the packet to create a process rule for transmission. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the Copeland 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. *See supra* at Claim 9. Uchida at [0012] ("According to a first aspect of the present invention, there is provided a packet forwarding apparatus comprising: a process rule storage unit that stores process rules defining a process content for each flow; a packet process unit that executes processing a received packet by using a process rule matching the received packet among the process rules and; an end determination information extraction unit that extracts information for checking

No.	'111 Patent Claim 14	Uchida
		a flow end from the received packet; and a flow end check unit that deletes, when a flow end is checked based on the extracted information, a process rule corresponding to the flow.") Uchida at [0014] ("According to a third aspect of the present invention, there is provided a process rule update method causing a packet forwarding apparatus comprising: a process rule stor-age unit that stores process rules defining a process content for each flow; and a packet process unit that executes and processing a received packet by using a process rule matching the received packet among the process rules to execute: extracting information for checking a flow received packet; and deleting, when a flow end is checked based on the extracted information, a process rule corre-sponding to the flow from the process rule storage unit. The present method is connected to a certain machine referred to as a packet forwarding apparatus comprising a process rule storage unit and a packet process unit.") Uchida at [0015] ("According to a fourth aspect of the present inven-tion, there is provided a program, causing a computer forming a packet forwarding apparatus comprising: a process rule storage unit that stores process rules defining a process con-tent for each flow; and a packet process unit that executes processing a received packet by using a process rule matching a received packet among the process rules to execute the processes of: extracting information for determining a flow end from the received packet; and deleting, when a flow end is determined based on the extracted information, a process rule corresponding to the flow from the process rule storage unit. The program can be stored in a computer-readable recording medium. Namely, the present invention can be realized as a computer program product. Also the program may be regarded as either a program product (i.e. manufac-ture) or a process embodying the load distribution method (process), expressed in a program language.") Uchida at [0039] ("First, an outline of an exemplary embodiment of the present invention will be described. Hereinafter, the reference characters used in this outline are used only as examples to facilitate comprehension and are not intended to limit the present invention to the illustrated modes. As illustrated in FIG. 1, a packet forwarding apparatus 1 according to an exemplary embodiment of the present invention includes: a process rule storage unit 11 storing process rules applied to received packets; a packet process unit 12 reading a process rule matching a received packet from the process rule storage unit to execute a forwarding process or the like; an end deter-mination information extraction unit 13 extracting

No.	'111 Patent Claim 14	Uchida
		informa-tion for checking a flow end from a received packet; and a flow end check unit 14 deleting, when a flow end is checked based on the extracted information, a process rule corre-sponding to the flow.") Uchida at [0068] ("In addition, in the actions in FIG. 19, "Forward" for forwarding the packet, "Drop" for dropping the packet, and "Modify-Field" for modifying information about the packet are present. An arbitrary number of actions can be registered in the actions (see the list in FIG. 21. However, in the case of Non-Patent Document 2, if the action field is blank, "Drop" is executed). For example, the Modify-Filed actions include: setting/changing of a VLAN ID or priority; stripping of a VLAN header; changing of an Ethernet (registered trademark) or IP source/destination address; and changing of a TCP/UDP source/destination port (other examples include changing of an IP ToS (Type of Service) field).") Uchida at [0070] ("The process rule search unit C14 determines actions (process contents) to be applied to the received packet, based on the information transmitted from the packet process information extraction unit C12 and outputs the actions to the packet process unit C15. More specifically, the process rule search unit C14 searches the process rule storage unit C13 for a process rule matching the information extracted from the received packet transmitted from the packet process information extraction unit C12 and outputs the process rule to the packet process unit C15. If a process rule matching the received packet exists, the process rule search unit C14 extracts actions (process contents) defined in the action field of the process rule and outputs the actions to the packet process unit C15. If no process rule matching the received packet is found, the process rule search unit C14 transmits the received packet or information extracted from the received packet to a predetermined control apparatus (controller; see FIG. 18) to request creation and transmission of a process rule. Upon receiving a process rule from the predetermined control apparatus (controller; see FIG. 18), the process rule search unit C14 registers the received process rule in the process rule storage unit C13. Next, the process rule search unit C14 extracts actions (process contents) from the process rule and outputs the actions to the packet process unit C15.") Uchida at [0071] ("The first process rule in the flow table in FIG. 20 is applied to packets indicating that the physical port number of the packet forwarding apparatus is 1; the source network is 192.168.0./24; the protocol type is Ox0006 (TCP); the desti-nation network is

No.	'111 Patent Claim 14	Uchida
		192.168.1./24; and the TCP destination port number is 80. Accordingly, the actions (process contents) of changing the IP source address of the packet to 192.168.1/24 and transmitting the packet from physical port number 2 of the forwarding apparatus are executed. Similarly, the second process rule in FIG. 20 is applied to packets indicating a destination port number other than HTTP port numbers. Accordingly, the actions (process contents) of changing the IP source address of the packet to 192.168.1/24 and transmit-ting the packet from physical port number 3 of the forwarding apparatus are executed.") Uchida at [0083] ("Next, upon receiving the extracted information, the process rule search unit C14 searches the process rule storage unit C13 for a process rule matching the extracted informa-tion (step S104). If a process rule matching the extracted information is found, the process rule search unit C14 extracts actions (process contents) defined in the process rule and transmits the actions to the packet process unit C15 (step S105). If no process rule matching the extracted information is found, the process rule search unit C14 transmits the received packet or information extracted from the received packet to a predetermined control apparatus (controller; see FIG. 18) to request creation and transmission of a process rule (step S106). Upon receiving a process rule from the predeter-mined control apparatus (controller; see FIG. 18), the process rule search unit C14 registers the received process rule in the process rule storage unit C13, extracts actions (process con-tents) in the process rule, and outputs the actions to the packet process unit C15 (step S107).") Uchida at [0106] ("First, the computer D10 starts communication with the server D20. For an initial establishment of communication, a packet (SYN) is inputted to the packet forwarding apparatus C1 (start of ACTIVE OPEN through SYN forwarding in FIG. 6). The packet reception unit C10 receives and stores this first packet in the packet storage unit C11 (steps S101 to S102 in FIG. 3).") 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.")

No.	'111 Patent Claim 14	Uchida
		Uchida at [0109] ("Next, a process rule corresponding to the received packet is searched for. The packet process information extraction unit C12 transmits the extracted information (representing that the ingress port is 1; the source address is 192.168.0.10; the destination address is 192.168.1.10; and the protocol is TCP (the type is 0x0006)) to a process rule search unit C14. The process rule search unit C14 refers to the process rule storage unit C13. Since no process rule matching the packet exists, the process rule search unit C14 queries the control apparatus (controller) D1 (steps S104 and S106 in FIG. 3). The control apparatus (controller) D1 creates a process rule including process contents representing modification of the source IP and forward from a specified transport port and transmits the process rule to the packet forwarding apparatus C1. The process rule search unit C14 registers the process rule received from the control apparatus (controller) D1 as a new entry in the process rule storage unit C13 (step S107 in FIG. 3). At this stage, as illustrated in FIG. 9, a new process rule is added to the process rule storage unit C13.") Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, Uchida 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. For example, Chua '877 discloses analyzing, including deep packet inspection, performed by the controller on packets received. 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.")

No.	'111 Patent Claim 14	Uchida
		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.") 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.") 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.") 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.

No.	'111 Patent Claim 14	Uchida
		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.") 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.") 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

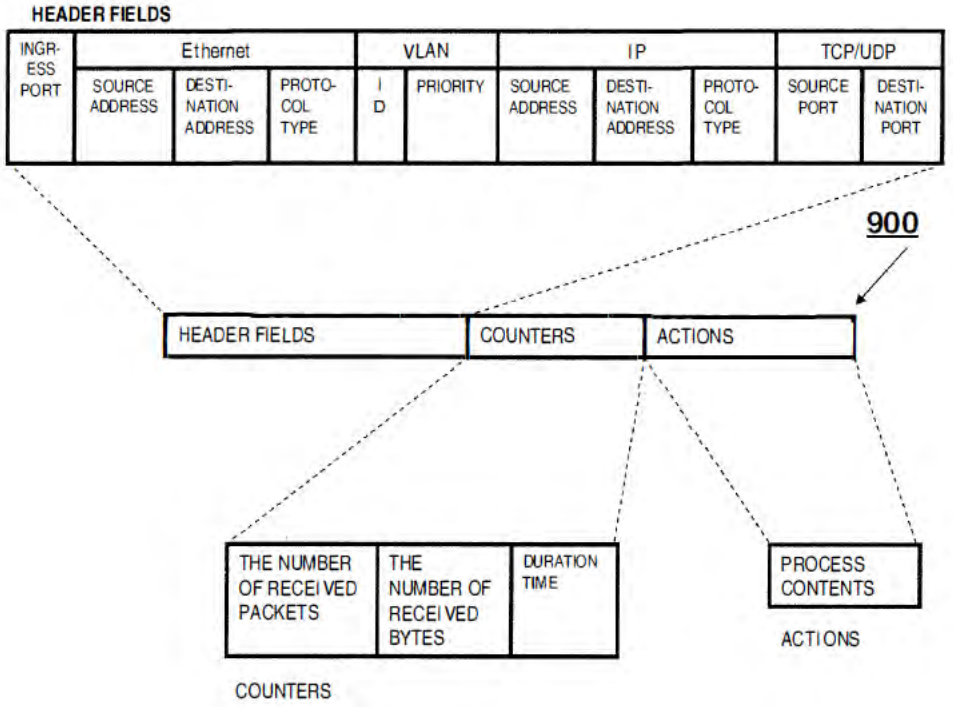
No.	'111 Patent Claim 14	Uchida
		encrypted protocols. The classifiers 18, 22 are configured to perform deep packet inspection (DPI), which provides the ability to look into the packet past basic header information so that the contents of a particular packet can be determined.") Chandrasekaran at [0021] ("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.") 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.") 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.") 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

No.	'111 Patent Claim 14	Uchida
		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).") 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.")

No.	'111 Patent Claim 15	Uchida
15[a]	The method according to claim 9, wherein the packet comprises distinct header and payload fields, and	Uchida discloses the method according to claim 9, wherein the packet comprises distinct header and payload fields. For example, Uchida discloses packets composed of header and process content fields. *See supra* at Claim 9. Uchida at [0063] ("When notified of reception of the packet by the packet reception unit C10, to determine process contents to be applied to the packet, the packet process information extraction unit C12 uses the notification as a trigger to extract header field information and reception status information (the above ingress port (receiving port ID information) and the like) about the received packet from the packet storage unit C11. The packet process information extraction unit C12 outputs the extracted packet information to the process rule search unit C14.")

No.	'111 Patent Claim 15	Uchida
		Uchida at [0064] ("As the information to be extracted to determine the process contents to be applied to the received packet by the packet process information extraction unit C12, information identical to that used in an OpenFlow switch in Non-Patent Documents 1 and 2 can be used. For example, the following information can be used: an ingress port (receiving port ID information); a source address, a destination address, and an upper protocol type of Ethernet (registered trademark); a VID (VLAN-ID) and a PCP (Priority Code Point) in TCI (Tag Control Information) of an IEEE802.1Q-compliant VLAN (Virtual LAN); a source address, a destination address, and an upper protocol type of the IP (Internet Protocol); and a source port and a destination port of the TCP (Transmission Control Protocol) or the UDP (User Datagram Protocol). Among the above information, the ingress port (receiving port ID information) of the packet forwarding apparatus can be obtained from the reception status information, and the other information can be obtained from header fields of the received packet, upper protocol headers of the packet, or the like.") Uchida at [0065] ("For each flow, the process rule storage unit C13 stores a process rule defining process contents applied to a received packet. The process rule includes contents similar to those in a flow table in Non-Patent Documents 1 and 2 in FIGS. 18 and 19. The process rule may be an entry, in which header fields that are matched against a received packet to determine the flow; statistics information (counters); and process contents (actions) are associated with each other.") Uchida at [0066] ("For example, the header fields in FIG. 19 include: an ingress port of the packet forwarding apparatus; a source address, a destination address, and an upper protocol type of Ethernet (registered trademark); a VID and a priority (PCP; Priority Code Point) of an IEEE802.1Q-compliant VLAN; a source address, a destination address, and an upper protocol type of the IP; and a source port and a destination port of the TCP or the UDP. When a packet matching the contents of these header fields is received, process contents defined in the action field are executed. However, wildcards can be used in arbitrary information bits in the header fields.") Uchida at [0067] ("Further, the counters (fields) in FIG. 19 hold the number of received packets, the number of received bytes, and the rule duration time, which are used as statistics information about the flow to which the packet belongs.")

No.	'111 Patent Claim 15	Uchida
		Uchida at [0070] ("The process rule search unit C14 determines actions (process contents) to be applied to the received packet, based on the information transmitted from the packet process information extraction unit C12 and outputs the actions to the packet process unit C15. More specifically, the process rule search unit C14 searches the process rule storage unit C13 for a process rule matching the information extracted from the received packet transmitted from the packet process information extraction unit C12 and outputs the process rule to the packet process unit C15. If a process rule matching the received packet exists, the process rule search unit C14 extracts actions (process contents) defined in the action field of the process rule and outputs the actions to the packet process unit C15. If no process rule matching the received packet is found, the process rule search unit C14 transmits the received packet or information extracted from the received packet to a predetermined control apparatus (controller; see FIG. 18) to request creation and transmission of a process rule. Upon receiving a process rule from the predetermined control apparatus (controller; see FIG. 18), the process rule search unit C14 registers the received process rule in the process rule storage unit C13. Next, the process rule search unit C14 extracts actions (process contents) from the process rule and outputs the actions to the packet process unit C15.") Uchida at Figure 19

No.	'111 Patent Claim 15	Uchida
		PRIOR ART FIG. 19 

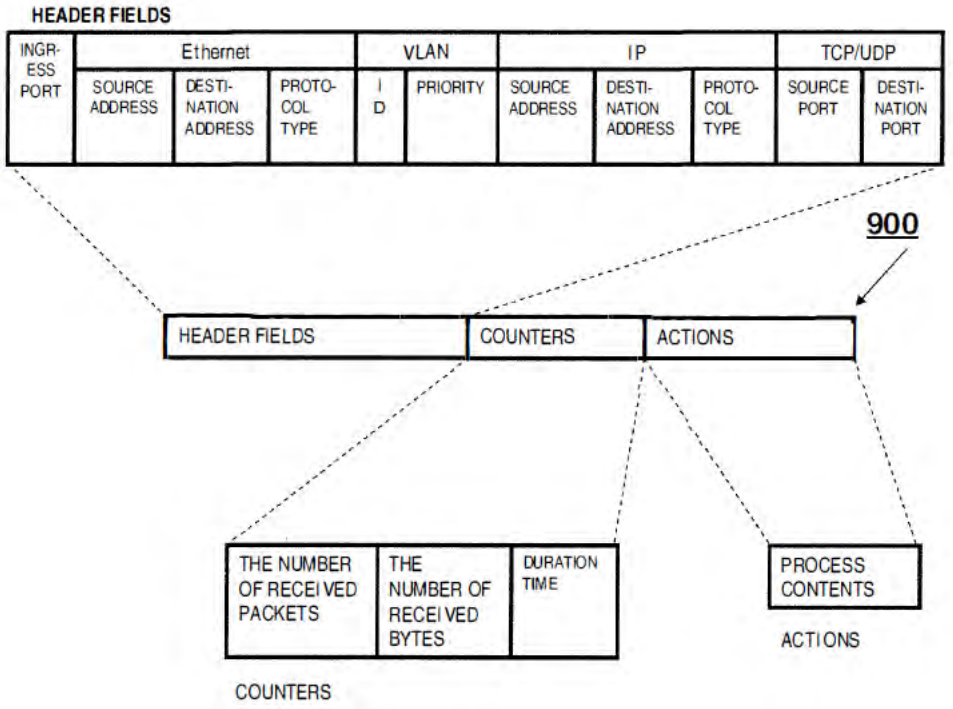
No.	'111 Patent Claim 15	Uchida
15[b]	wherein the analyzing comprises checking part of, or whole of, the payload field.	Uchida discloses wherein the analyzing comprises checking part of, or whole of, the payload field. For example, Uchida discloses the process rule search unit C14 transmits the received packet or information extracted from the received packet to a predetermined control apparatus to be used to create a process rule. *See supra* at Claim 9. Uchida at [0063] ("When notified of reception of the packet by the packet reception unit C10, to determine process contents to be applied to the packet, the packet process information extraction unit C12 uses the notification as a trigger to extract header field information and reception status information (the above ingress port (receiving port ID information) and the like) about the received packet from the packet storage unit C11. The packet process information extraction unit C12 outputs the extracted packet information to the process rule search unit C14.") Uchida at [0064] ("As the information to be extracted to determine the process contents to be applied to the received packet by the packet process information extraction unit C12, information identical to that used in an OpenFlow switch in Non-Patent Documents 1 and 2 can be used. For example, the following information can be used: an ingress port (receiving port ID information); a source address, a destination address, and an upper protocol type of Ethernet (registered trademark); a VID (VLAN-ID) and a PCP (Priority Code Point) in TCI (Tag Control Information) of an IEEE802.1Q-compliant VLAN (Virtual LAN); a source address, a destination address, and an upper protocol type of the IP (Internet Protocol); and a source port and a destination port of the TCP (Transmission Control Protocol) or the UDP (User Datagram Protocol). Among the above information, the ingress port (receiving port ID information) of the packet forwarding apparatus can be obtained from the reception status information, and the other information can be obtained from header fields of the received packet, upper protocol headers of the packet, or the like.") Uchida at [0065] ("For each flow, the process rule storage unit C13 stores a process rule defining process contents applied to a received packet. The process rule includes contents similar to those in a flow table in Non-Patent Documents 1 and 2 in FIGS. 18 and 19. The

No.	'111 Patent Claim 15	Uchida
		process rule may be an entry, in which header fields that are matched against a received packet to determine the flow; statistics information (counters); and process contents (actions) are associated with each other.") Uchida at [0066] ("For example, the header fields in FIG. 19 include: an ingress port of the packet forwarding apparatus; a source address, a destination address, and an upper protocol type of Ethernet (registered trademark); a VID and a priority (PCP; Priority Code Point) of an IEEE802.1Q-compliant VLAN; a source address, a destination address, and an upper protocol type of the IP; and a source port and a destination port of the TCP or the UDP. When a packet matching the contents of these header fields is received, process contents defined in the action field are executed. However, wildcards can be used in arbitrary information bits in the header fields.") Uchida at [0067] ("Further, the counters (fields) in FIG. 19 hold the number of received packets, the number of received bytes, and the rule duration time, which are used as statistics information about the flow to which the packet belongs.") Uchida at [0070] ("The process rule search unit C14 determines actions (process contents) to be applied to the received packet, based on the information transmitted from the packet process information extraction unit C12 and outputs the actions to the packet process unit C15. More specifically, the process rule search unit C14 searches the process rule storage unit C13 for a process rule matching the information extracted from the received packet transmitted from the packet process information extraction unit C12 and outputs the process rule to the packet process unit C15. If a process rule matching the received packet exists, the process rule search unit C14 extracts actions (process contents) defined in the action field of the process rule and outputs the actions to the packet process unit C15. If no process rule matching the received packet is found, the process rule search unit C14 transmits the received packet or information extracted from the received packet to a predetermined control apparatus (controller; see FIG. 18) to request creation and transmission of a process rule. Upon receiving a process rule from the predetermined control apparatus (controller; see FIG. 18), the process rule search unit C14 registers the received process rule in the process rule storage unit C13. Next, the process rule search unit C14 extracts actions (process contents) from the process rule and outputs the actions to the packet process unit C15.")

No.	'111 Patent Claim 15	Uchida
		Uchida at Figure 19

PRIOR ART

FIG. 19



No.	'111 Patent Claim 16	Uchida
16[a]	The method according to claim 1, wherein the packet comprises distinct header and payload fields,	Uchida discloses the method according to claim 1, wherein the packet comprises distinct header and payload fields. *See supra* at Claim 1, 15[a].
16[b]	the header comprises one or more flag bits, and	Uchida discloses the header comprises one or more flag bits. For example, Uchida discloses packet headers that are comprised of a TCP (Transmission Control Protocol) FIN flag, RST flag, ACK flag and/or SYN flag. 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.") 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.") 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.")

No.	'111 Patent Claim 16	Uchida
		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.") 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.") 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.")

No.	'111 Patent Claim 16	Uchida
		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.") 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).") 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.") 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).") 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

No.	'111 Patent Claim 16	Uchida
		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).") 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.") 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.") 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).") 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

No.	'111 Patent Claim 16	Uchida
		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).") 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.") 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.") 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.") 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.") 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.")

No.	'111 Patent Claim 16	Uchida
		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).") 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).") 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.") 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.") 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.")

No.	'111 Patent Claim 16	Uchida
		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.") 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).") 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).")
16[c]	wherein the packet-applicable criterion is that one or more of the flag bits is set.	Uchida discloses wherein the packet-applicable criterion is that one or more of the flag bits is set. For example, Uchida discloses checking by the flow end check unit whether a flag bit is set in a predetermined packet header position. 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

No.	'111 Patent Claim 16	Uchida
		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.") 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.") 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.") 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.") 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.") Uchida at [0104] ("Hereinafter, the description will be made, assuming that a packet forwarding apparatus Cl according to the present invention relays TCP communication

No.	'111 Patent Claim 16	Uchida
		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.") 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.") 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).") 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

No.	'111 Patent Claim 16	Uchida
		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.") 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).") 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).") 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.") 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.")

No.	'111 Patent Claim 16	Uchida
		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).") 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).") 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.") 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.") 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

No.	'111 Patent Claim 16	Uchida
		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.") 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.") 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.") 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).") 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).")

No.	'111 Patent Claim 16	Uchida
		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.") 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.") 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.") 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.") 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).") 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).")

No.	'111 Patent Claim 16	Uchida
		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 17	Uchida
17[a]	The method according to claim 16, wherein the packet is an Transmission Control Protocol (TCP) packet, and	Uchida discloses the method according to claim 16, wherein the packet is an Transmission Control Protocol (TCP) packet. For example, Uchida discloses a TCP (Transmission Control Protocol) connection in which packets are transmitted. *See supra* at Claim 16. 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.") 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.")

No.	'111 Patent Claim 17	Uchida
		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.") 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.") 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.") 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

No.	'111 Patent Claim 17	Uchida
		monitors a protocol stack, including: fields in which the TCP is determined; and the FIN flag in the TCP header.") 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.") 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).") 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.") 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).") 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.

No.	'111 Patent Claim 17	Uchida
		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).") 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.") 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.") 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).") 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

No.	'111 Patent Claim 17	Uchida
		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).") 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.") 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.") 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.") 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.")

No.	'111 Patent Claim 17	Uchida
		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.") 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).") 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).") 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.") 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.") 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

No.	'111 Patent Claim 17	Uchida
		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.") 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.") 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).") 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).") 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 17	Uchida
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.	Uchida 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. For example, Uchida discloses header flag bits that are comprised of the TCP (Transmission Control Protocol) FIN flag, RST flag, ACK flag and/or SYN flag. 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.") 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.") 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.") 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.") 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

No.	'111 Patent Claim 17	Uchida
		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.") 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.") 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,

No.	'111 Patent Claim 17	Uchida
		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.") 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).") 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.") 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).") 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

No.	'111 Patent Claim 17	Uchida
		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).") 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.") 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.") 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).") 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).")

No.	'111 Patent Claim 17	Uchida
		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.") 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.") 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.") 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.") 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.") 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).")

No.	'111 Patent Claim 17	Uchida
		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).") 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.") 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.") 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.") 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.")

No.	'111 Patent Claim 17	Uchida
		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).") 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).") 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	Uchida
18[a]	The method according to claim 1, wherein the packet comprises distinct header and payload fields,	Uchida discloses the method according to claim 1, wherein the packet comprises distinct header and payload fields. *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	Uchida discloses the header comprises at least the first and second entities addresses in the packet network. For example, Uchida discloses a source address, a destination address, a source port and a destination port of the TCP (Transmission Control Protocol) information is obtained from the reception status information.

No.	'111 Patent Claim 18	Uchida
		Uchida at [0059] ("Next, a first exemplary embodiment of the present invention will be described in detail with reference to the drawings. First, terms will be defined. A "flow" signifies communication from a certain source to destination. Namely, a "flow" signifies a group of forwarded packets or the for-warding itself. Each of the "source" and "destination" does not necessarily signify a single physical or logical address. Namely, communication from a certain subnet to a different subnet may be collectively deemed as a single flow. Thus, a flow may be a collection of more detailed flows, one example of which corresponds to a flow entry in which a plurality of flows are unified by describing part of destination addresses, source addresses, or the like with wildcards, as in the case of OpenFlow in Non-Patent Documents 1 and 2.") Uchida at [0064] ("As the information to be extracted to determine the process contents to be applied to the received packet by the packet process information extraction unit C12, information identical to that used in an OpenFlow switch in Non-Patent Documents 1 and 2 can be used. For example, the following information can be used: an ingress port (receiving port ID information); a source address, a destination address, and an upper protocol type of Ethernet (registered trademark); a VID (VLAN-ID) and a PCP (Priority Code Point) in TCI (Tag Control Information) of an IEEE802.1Q-compliant VLAN (Virtual LAN); a source address, a destination address, and an upper protocol type of the IP (Internet Protocol); and a source port and a destination port of the TCP (Transmission Control Protocol) or the UDP (User Datagram Protocol). Among the above information, the ingress port (receiving port ID information) of the packet forwarding apparatus can be obtained from the reception status information, and the other information can be obtained from header fields of the received packet, upper protocol headers of the packet, or the like.") Uchida at [0066] ("For example, the header fields in FIG.19 include: an ingress port of the packet forwarding apparatus; a source address, a destination address, and an upper protocol type of Ethernet (registered trademark); a YID and a priority (PCP; Priority Code Point) of an IEEE802.lQ-compliant YLAN; a source address, a destination address, and an upper protocol type of the IP; and a source port and a destination port of the TCP or the UDP. When a packet matching the contents of these header fields is received, process contents defined in the action field are executed. However, wildcards can be used in arbitrary information bits in the header fields.")

No.	'111 Patent Claim 18	Uchida
		Uchida at [0109] ("Next, a process rule corresponding to the received packet is searched for. The packet process information extrac-tion unit C12 transmits the extracted information (represent-ing that the ingress port is 1; the source address is 192.168. 0.10; the destination address is 192.168.1.10; and the protocol is TCP (the type is 0x0006)) to a process rule search unit C14. The process rule search unit C14 refers to the process rule storage unit C13. Since no process rule matching the packet exists, the process rule search unit C14 queries the control apparatus (controller) Dl (steps S104 and S106 in FIG. 3). The control apparatus (controller) Dl creates a pro-cess rule including process contents representing modifica-tion of the source IP and forward from a specified transport port and transmits the process rule to the packet forwarding apparatus Cl. The process rule search unit C14 registers the process rule received from the control apparatus (controller) Dl as a new entry in the process rule storage unit C13 (step S107 in FIG. 3). At this stage, as illustrated in FIG. 9, a new process rule is added to the process rule storage unit C13.")
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.	Uchida discloses wherein the packet-applicable criterion is that the first entity address, the second entity address, or both match a predetermined address or addresses. For example, Uchida discloses header fields that can be comprised of source and destination addresses that are matched against a received packet to determine the flow. Uchida at [0059] ("Next, a first exemplary embodiment of the present invention will be described in detail with reference to the drawings. First, terms will be defined. A "flow" signifies communication from a certain source to destination. Namely, a "flow" signifies a group of forwarded packets or the for-warding itself. Each of the "source" and "destination" does not necessarily signify a single physical or logical address. Namely, communication from a certain subnet to a different subnet may be collectively deemed as a single flow. Thus, a flow may be a collection of more detailed flows, one example of which corresponds to a flow entry in which a plurality of flows are unified by describing part of destination addresses, source addresses, or the like with wildcards, as in the case of OpenFlow in Non-Patent Documents 1 and 2.") Uchida at [0064] ("As the information to be extracted to determine the process contents to be applied to the received packet by the packet process information extraction unit C12,

No.	'111 Patent Claim 18	Uchida
		information identical to that used in an OpenFlow switch in Non-Patent Documents 1 and 2 can be used. For example, the following information can be used: an ingress port (receiving port ID information); a source address, a destination address, and an upper protocol type of Ethernet (registered trademark); a VID (VLAN-ID) and a PCP (Priority Code Point) in TCI (Tag Control Information) of an IEEE802.1Q-compliant VLAN (Virtual LAN); a source address, a destination address, and an upper protocol type of the IP (Internet Protocol); and a source port and a destination port of the TCP (Transmission Control Protocol) or the UDP (User Datagram Protocol). Among the above information, the ingress port (receiving port ID information) of the packet forwarding apparatus can be obtained from the reception status information, and the other information can be obtained from header fields of the received packet, upper protocol headers of the packet, or the like.") Uchida at [0066] ("For example, the header fields inFIG.19 include: an ingress port of the packet forwarding apparatus; a source address, a destination address, and an upper protocol type of Ethernet (registered trademark); a YID and a priority (PCP; Priority Code Point) of an IEEE802.lQ-compliant YLAN; a source address, a destination address, and an upper protocol type of the IP; and a source port and a destination port of the TCP or the UDP. When a packet matching the contents of these header fields is received, process contents defined in the action field are executed. However, wildcards can be used in arbitrary information bits in the header fields.") 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.") Uchida at [0109] ("Next, a process rule corresponding to the received packet is searched for. The packet process information extrac-tion unit C12 transmits the extracted information (represent-ing that the ingress port is 1; the source address is 192.168. 0.10; the destination address is 192.168.1.10; and the protocol is TCP (the type is 0x0006)) to a process rule search unit C14. The process rule search unit C14 refers to the process rule storage unit C13. Since no process rule matching the packet exists, the process rule search unit C14 queries the control

No.	'111 Patent Claim 18	Uchida
		apparatus (controller) Dl (steps S104 and S106 in FIG. 3). The control apparatus (controller) Dl creates a pro-cess rule including process contents representing modifica-tion of the source IP and forward from a specified transport port and transmits the process rule to the packet forwarding apparatus Cl. The process rule search unit C14 registers the process rule received from the control apparatus (controller) Dl as a new entry in the process rule storage unit C13 (step S107 in FIG. 3). At this stage, as illustrated in FIG. 9, a new process rule is added to the process rule storage unit C13.")

No.	'111 Patent Claim 19	Uchida
19	The method according to claim 18, wherein the addresses are Internet Protocol (IP) addresses.	Uchida discloses the method according to claim 18, wherein the addresses are Internet Protocol (IP) addresses. For example, Uchida discloses IP source and destination addresses. *See supra* at Claim 18. Uchida at [0059] ("Next, a first exemplary embodiment of the present invention will be described in detail with reference to the drawings. First, terms will be defined. A "flow" signifies communication from a certain source to destination. Namely, a "flow" signifies a group of forwarded packets or the for-warding itself. Each of the "source" and "destination" does not necessarily signify a single physical or logical address. Namely, communication from a certain subnet to a different subnet may be collectively deemed as a single flow. Thus, a flow may be a collection of more detailed flows, one example of which corresponds to a flow entry in which a plurality of flows are unified by describing part of destination addresses, source addresses, or the like with wildcards, as in the case of OpenFlow in Non-Patent Documents 1 and 2.") Uchida at [0064] ("As the information to be extracted to determine the process contents to be applied to the received packet by the packet process information extraction unit C12, information identical to that used in an OpenFlow switch in Non-Patent Documents 1 and 2 can be used. For example, the following information can be used: an ingress port (receiving port ID information); a source address, a destination address, and an upper protocol type of Ethernet (registered trademark); a VID (VLAN-ID) and a PCP (Priority Code Point) in TCI

No.	'111 Patent Claim 19	Uchida
		(Tag Control Information) of an IEEE802.1Q-compliant VLAN (Virtual LAN); a source address, a destination address, and an upper protocol type of the IP (Internet Protocol); and a source port and a destination port of the TCP (Transmission Control Protocol) or the UDP (User Datagram Protocol). Among the above information, the ingress port (receiving port ID information) of the packet forwarding apparatus can be obtained from the reception status information, and the other information can be obtained from header fields of the received packet, upper protocol headers of the packet, or the like.") Uchida at [0066] ("For example, the header fields inFIG.19 include: an ingress port of the packet forwarding apparatus; a source address, a destination address, and an upper protocol type of Ethernet (registered trademark); a YID and a priority (PCP; Priority Code Point) of an IEEE802.lQ-compliant YLAN; a source address, a destination address, and an upper protocol type of the IP; and a source port and a destination port of the TCP or the UDP. When a packet matching the contents of these header fields is received, process contents defined in the action field are executed. However, wildcards can be used in arbitrary information bits in the header fields.") 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.") Uchida at [0109] ("Next, a process rule corresponding to the received packet is searched for. The packet process information extrac-tion unit C12 transmits the extracted information (represent-ing that the ingress port is 1; the source address is 192.168. 0.10; the destination address is 192.168.1.10; and the protocol is TCP (the type is 0x0006)) to a process rule search unit C14. The process rule search unit C14 refers to the process rule storage unit C13. Since no process rule matching the packet exists, the process rule search unit C14 queries the control apparatus (controller) Dl (steps S104 and S106 in FIG. 3). The control apparatus (controller) Dl creates a pro-cess rule including process contents representing modifica-tion of the source IP and forward from a specified transport port and transmits the process rule to the packet forwarding apparatus Cl. The process rule search unit C14 registers the process rule received

No.	'111 Patent Claim 19	Uchida
		from the control apparatus (controller) Dl as a new entry in the process rule storage unit C13 (step S107 in FIG. 3). At this stage, as illustrated in FIG. 9, a new process rule is added to the process rule storage unit C13.") Uchida at [0072] ("The packet process unit C15 processes a received packet stored in the packet storage unit C11 in accordance with the actions (process contents) outputted from the process rule search unit C14. For example, upon receiving a Forward action for forwarding a packet, the packet process unit C15 requests the packet forwarding unit C16 to transmit the packet. In addition to the actions (process contents) outputted from the process rule search unit C14, the packet process unit C15 appropriately executes basic processes and the like relating to communication. For example, in the case of the first process rule in FIG. 20, the packet process unit C15 first changes the source IP address to 192.168.1.1 and next transmits the packet, as the actions (process contents). Simultaneously, as basic processes relating to communication, the packet process unit C15 also executes basic processes relating to the IP and the like, such as rewriting of the TTL (Time To Live) field, the checksum field, and the like in the IP headers, as needed. The basic processes relating to communication include determining the end of packet forwarding when the TTL value reaches 0 or the like.")

No.	'111 Patent Claim 20	Uchida
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	Uchida 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. For example, Uchida discloses a source port and a destination port of the TCP (Transmission Control Protocol). *See supra* at Claim 1, 17[a]. 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

No.	'111 Patent Claim 20	Uchida
		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.") 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.") 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.") 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.") 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.")

No.	'111 Patent Claim 20	Uchida
		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.") 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.") 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).") 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

No.	'111 Patent Claim 20	Uchida
		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.") 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).") 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).") 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.") 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

No.	'111 Patent Claim 20	Uchida
		determi-nation process from steps Slll to S116 is started, as in the case of the above start of ACTIVE CLOSE.") 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).") 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).") 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.") 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.")

No.	'111 Patent Claim 20	Uchida
		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.") 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.") 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.") 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).") 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

No.	'111 Patent Claim 20	Uchida
		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).") 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.") 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.") 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.") 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.") 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).") 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

No.	'111 Patent Claim 20	Uchida
		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).") 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.")
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.	Uchida 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. For example, Uchida discloses a source port and a destination port of an TCP packet that are checked against a flow entry. 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.")

No.	'111 Patent Claim 20	Uchida
		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.") 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.") 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.") 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.") 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

No.	'111 Patent Claim 20	Uchida
		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.") 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.") 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).") 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.") 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).")

No.	'111 Patent Claim 20	Uchida
		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).") 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.") 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.") 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).") 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.

No.	'111 Patent Claim 20	Uchida
		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).") 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.") 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.") 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.")

No.	'111 Patent Claim 20	Uchida
		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.") 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.") 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).") 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).") 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.")

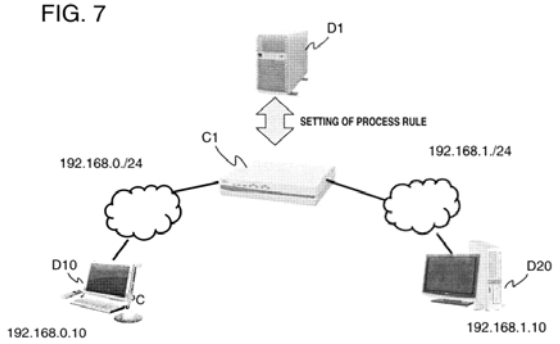
No.	'111 Patent Claim 20	Uchida
		Uchida at [0148] (" Next, a third specific example in which a flow end determination is executed by using a TCP RST (reset) flag will be described.") Uchida at [0149] ("Referring back to the state transition diagram of TCP connection in FIG. 6, there is a transition 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.") 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.") 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).") 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).") 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 21	Uchida
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.	Uchida 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. For example, Uchida discloses the VLAN (Virtual LAN) and IP addresses that denotes a network comprised of an Internet service provider. *See supra* at Claim 1. Uchida at [0064] ("As the information to be extracted to determine the process contents to be applied to the received packet by the packet process information extraction unit C12, information identical to that used in an OpenFlow switch in Non-Patent Documents 1 and 2 can be used. For example, the following information can be used: an ingress port (receiving port ID information); a source address, a destination address, and an upper protocol type of Ethernet (registered trademark); a VID (VLAN-ID) and a PCP (Priority Code Point) in TCI (Tag Control Information) of an IEEE802.1Q-compliant VLAN (Virtual LAN); a source address, a destination address, and an upper protocol type of the IP (Internet Protocol); and a source port and a destination port of the TCP (Transmission Control Protocol) or the UDP (User Datagram Protocol). Among the above information, the ingress port (receiving port ID information) of the packet forwarding apparatus can be obtained from the reception status information, and the other information can be obtained from header fields of the received packet, upper protocol headers of the packet, or the like.")

No.	'111 Patent Claim 22	Uchida
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	Uchida 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. For example, Uchida discloses communication between a computer (client) D10 and a server D20. *See supra* at Claim 1.

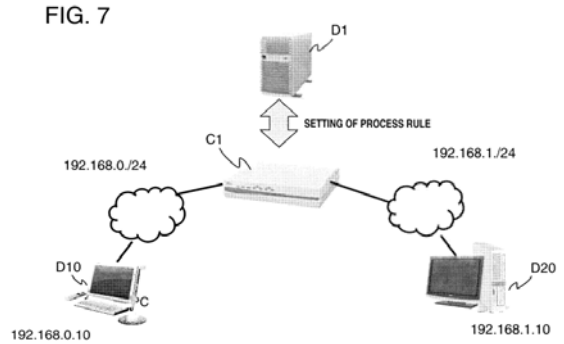
No.	'111 Patent Claim 22	Uchida
	entity is a server device.	Uchida at [0104] ("Hereinafter, the description will be made, assuming that a packet forwarding apparatus C1 according to the present invention relays TCP communication between a computer (client) D10 and a server D20 that use network configurations illustrated in FIG. 7. In the example of FIG. 7, the computer D10 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 (controller) D1 is connected to the packet forwarding apparatus C1 via a dedicated channel and manages connection between the two networks. In the following description, the control apparatus (controller) D1 controls the packet forwarding apparatus C1 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 C17 monitors a protocol stack, including: fields in which the TCP is determined; and the FIN flag in the TCP header.") 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).") Uchida at [0112] ("Next, for an establishment of communication with the computer Dl0, an acknowledgement (SYN/ACK packet) is inputted to the packet forwarding apparatus Cl from the server D20 (SYN/ ACK forwarding in FIG. 6). As in the case of the above establishment of communication from the com-puter DlO to the server D20, the process rule storage unit C13 is referred to. However, since no process rule matching the acknowledgement packet exists, the control apparatus (con-troller) Dl is queried, and a new process rule defining a process contents to be applied to a reverse-direction flow is added(steps S104, S106, and S107 in FIG. 3).At this stage, as illustrated in FIG. 10, a new process rule is added to the process rule storage unit C13.")

No.	'111 Patent Claim 22	Uchida
		Uchida at [0113] ("Next, for an establishment of communication, the computer Dl0 transmits the final acknowledgement (ACK packet) to the server D20 (ACK forwarding in FIG. 6). At this point, a process rule matching the flow from the computer Dl Oto the server D20 is already registered in the process rule storage unit C13. Thus, the process rule search unit C14 extracts the process contents (actions) defined in the corre-sponding process rule from the process rule storage unit C13 (step Sl 05). Subsequently, the packet exchanged between the computer Dl 0 and the server D20 is modified and relayed as described above ("ESTABLISHED" in FIG. 6).") Uchida at [0114] ("Subsequently, when ending the communication, the computer DlO transmits a FIN packet representing the end of the communication to the server D20 (start of ACTIVE CLOSE through FIN forwarding in FIG. 6). This FIN packet is inputted to the packet forwarding apparatus Cl. The FIN packet is forwarded in the same way as the above ACK packet, description thereof will be omitted. Hereinafter, steps Slll to S116 in FIG. 3 will be described with reference to FIG. 8.") 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.") Uchida at Figure 7

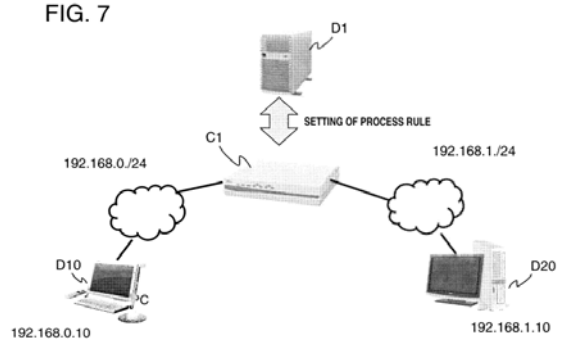
No.	'111 Patent Claim 22	Uchida
		FIG. 7 

No.	'111 Patent Claim 23	Uchida
23[a]	The method according to claim 22, wherein the server device comprises a web server, and	Uchida discloses the method according to claim 22, wherein the server device comprises a web server. For example, Uchida discloses a server D20 belonging to a network communicating with a computer, which would constitute a web server. *See supra* at Claim 22. Uchida at [0104] ("Hereinafter, the description will be made, assuming that a packet forwarding apparatus C1 according to the present invention relays TCP communication between a computer (client) D10 and a server D20 that use network configurations illustrated in FIG. 7. In the example of FIG. 7, the computer D10 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 (controller) D1 is connected to the packet forwarding apparatus C1 via a dedicated channel and manages connection between the two networks. In the following description, the control apparatus (controller) D1 controls the packet forwarding apparatus C1 so that connection from other networks appears as communication from network number 1 (192.168.1.1) of the respective networks (see

No.	'111 Patent Claim 23	Uchida
		process rule actions in FIG. 19). In addition, in the present specific example, since FIN packets are monitored, the end determination information extraction unit C17 monitors a protocol stack, including: fields in which the TCP is determined; and the FIN flag in the TCP header.") 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).") Uchida at [0112] ("Next, for an establishment of communication with the computer Dl0, an acknowledgement (SYN/ACK packet) is inputted to the packet forwarding apparatus Cl from the server D20 (SYN/ ACK forwarding in FIG. 6). As in the case of the above establishment of communication from the com-puter DlO to the server D20, the process rule storage unit C13 is referred to. However, since no process rule matching the acknowledgement packet exists, the control apparatus (con-troller) Dl is queried, and a new process rule defining a process contents to be applied to a reverse-direction flow is added(steps S104, S106, and S107 in FIG. 3).At this stage, as illustrated in FIG. 10, a new process rule is added to the process rule storage unit C13.") Uchida at [0113] ("Next, for an establishment of communication, the computer Dl0 transmits the final acknowledgement (ACK packet) to the server D20 (ACK forwarding in FIG. 6). At this point, a process rule matching the flow from the computer Dl Oto the server D20 is already registered in the process rule storage unit C13. Thus, the process rule search unit C14 extracts the process contents (actions) defined in the corre-sponding process rule from the process rule storage unit C13 (step Sl 05). Subsequently, the packet exchanged between the computer Dl 0 and the server D20 is modified and relayed as described above ("ESTABLISHED" in FIG. 6).") Uchida at [0114] ("Subsequently, when ending the communication, the computer DlO transmits a FIN packet representing the end of the communication to the server D20 (start of ACTIVE CLOSE through FIN forwarding in FIG. 6). This FIN packet is inputted to the packet forwarding apparatus Cl. The FIN packet is forwarded in the same way as the above

No.	'111 Patent Claim 23	Uchida
		ACK packet, description thereof will be omitted. Hereinafter, steps Slll to S116 in FIG. 3 will be described with reference to FIG. 8.") 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.") Uchida at Figure 7 
23[b]	wherein the client device comprises a smartphone, a tablet computer, a personal computer, a laptop computer, or a wearable computing device.	Uchida discloses wherein the client device comprises a smartphone, a tablet computer, a personal computer, a laptop computer, or a wearable computing device. For example, Uchida discloses a computer as client device D10. Uchida at [0104] ("Hereinafter, the description will be made, assuming that a packet forwarding apparatus C1 according to the present invention relays TCP communication between a computer (client) D10 and a server D20 that use network configurations illustrated in FIG. 7. In the example of FIG. 7, the computer D10 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

No.	'111 Patent Claim 23	Uchida
		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 (controller) D1 is connected to the packet forwarding apparatus C1 via a dedicated channel and manages connection between the two networks. In the following description, the control apparatus (controller) D1 controls the packet forwarding apparatus C1 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 C17 monitors a protocol stack, including: fields in which the TCP is determined; and the FIN flag in the TCP header.") 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).") Uchida at [0112] ("Next, for an establishment of communication with the computer Dl0, an acknowledgement (SYN/ACK packet) is inputted to the packet forwarding apparatus Cl from the server D20 (SYN/ ACK forwarding in FIG. 6). As in the case of the above establishment of communication from the com-puter DlO to the server D20, the process rule storage unit C13 is referred to. However, since no process rule matching the acknowledgement packet exists, the control apparatus (con-troller) Dl is queried, and a new process rule defining a process contents to be applied to a reverse-direction flow is added(steps S104, S106, and S107 in FIG. 3).At this stage, as illustrated in FIG. 10, a new process rule is added to the process rule storage unit C13.") Uchida at [0113] ("Next, for an establishment of communication, the computer Dl0 transmits the final acknowledgement (ACK packet) to the server D20 (ACK forwarding in FIG. 6). At this point, a process rule matching the flow from the computer Dl Oto the server D20 is already registered in the process rule storage unit C13. Thus, the process rule search unit C14 extracts the process contents (actions) defined in the corre-sponding process rule from the process rule storage unit C13 (step Sl 05). Subsequently, the packet exchanged between the

No.	'111 Patent Claim 23	Uchida
		computer Dl 0 and the server D20 is modified and relayed as described above ("ESTABLISHED" in FIG. 6).") Uchida at [0114] ("Subsequently, when ending the communication, the computer DlO transmits a FIN packet representing the end of the communication to the server D20 (start of ACTIVE CLOSE through FIN forwarding in FIG. 6). This FIN packet is inputted to the packet forwarding apparatus Cl. The FIN packet is forwarded in the same way as the above ACK packet, description thereof will be omitted. Hereinafter, steps Slll to S116 in FIG. 3 will be described with reference to FIG. 8.") 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.") Uchida at Figure 7 FIG. 7 

No.	'111 Patent Claim 24	Uchida
24	The method according to claim 22, wherein the communication	Uchida 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.

No.	'111 Patent Claim 24	Uchida
	between the network node and the controller is based on, or uses, a standard protocol.	For example, Uchida discloses an OpenFlow switch, which uses the OpenFlow protocol to communicate with an OpenFlow controller. *See supra* at Claim 22. Uchida at [0003] ("In recent years, a technique referred to as OpenFlow is proposed (see Non-Patent Document 1). In OpenFlow, communication is deemed as an end-to-end flow, and routing control, failure recovery, load distribution, and optimization are executed for each flow. FIG. 18 is a conceptual diagram of an OpenFlow switch. The OpenFlow switch, which serves as a forwarding node, includes a secure channel for communi-cation with an OpenFlow controller serving as a control appa-ratus. The OpenFlow switch operates in accordance with a flow table appropriately added or rewritten by the OpenFlow controller. In the flow table, a group of: matching rules (header fields) matched against packet headers; actions defin-ing process contents; and flow statistics information (counters) is defined for each flow (see FIGS. 19 and 20).") Uchida at [0005] ("For example, upon receiving a packet, the Open-Flow switch searches the flow table for an entry having matching rules (see header fields in FIGS. 19 and 20) that match header information of the received packet. As a result of the search, if an entry matching the received packet is found, the OpenFlow switch executes process contents described in action fields of the entry on the received packet. If, as a result of the search, no entry matching the received packet is found, the OpenFlow switch forwards the received packet to the OpenFlow controller via the secure channel to request the OpenFlow controller to determine a packet route based on the source/destination of the received packet. Upon receiving a flow entry realizing the route, the OpenFlow switch updates the flow table. In this way, the OpenFlow switch uses an entry stored in the flow table as a process rule to forward a packet.") Uchida at [0059] ("Next, a first exemplary embodiment of the present invention will be described in detail with reference to the drawings. First, terms will be defined. A "flow" signifies communication from a certain source to destination. Namely, a "flow" signifies a group of forwarded packets or the for-warding itself. Each of the "source" and "destination" does not necessarily signify a single physical or logical address. Namely, communication from a certain subnet to a different subnet may be collectively deemed as a single flow. Thus, a flow may be a collection of more detailed flows, one example of which corresponds to a

No.	'111 Patent Claim 24	Uchida
		flow entry in which a plurality of flows are unified by describing part of destination addresses, source addresses, or the like with wildcards, as in the case of OpenFlow in Non-Patent Documents 1 and 2.") Uchida at [0064] ("As the information to be extracted to determine the process contents to be applied to the received packet by the packet process information extraction unit C12, information identical to that used in an OpenFlow switch in Non-Patent Documents 1 and 2 can be used. For example, the following information can be used: an ingress port (receiving port ID information); a source address, a destination address, and an upper protocol type of Ethernet (registered trademark); a VID (VLAN-ID) and a PCP (Priority Code Point) in TCI (Tag Control Information) of an IEEE802.1Q-compliant VLAN (Virtual LAN); a source address, a destination address, and an upper protocol type of the IP (Internet Protocol); and a source port and a destination port of the TCP (Transmission Control Protocol) or the UDP (User Datagram Protocol). Among the above information, the ingress port (receiving port ID information) of the packet forwarding apparatus can be obtained from the reception status information, and the other information can be obtained from header fields of the received packet, upper protocol headers of the packet, or the like.") Uchida at [0104] ("Hereinafter, the description will be made, assuming that a packet forwarding apparatus C1 according to the present invention relays TCP communication between a computer (client) D10 and a server D20 that use network configurations illustrated in FIG. 7. In the example of FIG. 7, the computer D10 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 (controller) D1 is connected to the packet forwarding apparatus C1 via a dedicated channel and manages connection between the two networks. In the following description, the control apparatus (controller) D1 controls the packet forwarding apparatus C1 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 C17 monitors a protocol stack, including: fields in which the TCP is determined; and the FIN flag in the TCP header.")

No.	'111 Patent Claim 24	Uchida

No.	'111 Patent Claim 27	Uchida
27	The method according to claim 1, wherein the network node comprises a router, a switch, or a bridge.	Uchida discloses the method according to claim 1, wherein the network node comprises a router, a switch, or a bridge. For example, Uchida discloses a packet forwarding apparatus such as a HUB or a switch. *See supra* at Claim 1. Uchida at [0003] ("In recent years, a technique referred to as OpenFlow is proposed (see Non-Patent Document 1). In OpenFlow, communication is deemed as an end-to-end flow, and routing control, failure recovery, load distribution, and optimization are executed for each flow. FIG. 18 is a conceptual diagram of an OpenFlow switch. The OpenFlow switch, which serves as a forwarding node, includes a secure channel for communi-cation with an OpenFlow controller serving as a control appa-ratus. The OpenFlow switch operates in accordance with a flow table appropriately added or rewritten by the OpenFlow controller. In the flow table, a group of: matching rules (header fields) matched against packet headers; actions defin-ing process contents; and flow statistics information (counters) is defined for each flow (see FIGS. 19 and 20).") Uchida at [0005] ("For example, upon receiving a packet, the Open-Flow switch searches the flow table for an entry having matching rules (see header fields in FIGS. 19 and 20) that match header information of the received packet. As a result of the search, if an entry matching the received packet is found, the OpenFlow switch executes process contents described in action fields of the entry on the received packet. If, as a result of the search, no entry matching the received packet is found, the OpenFlow switch forwards the received packet to the OpenFlow controller via the secure channel to request the OpenFlow controller to determine a packet route based on the source/destination of the received packet. Upon receiving a flow entry realizing the route, the OpenFlow switch updates the flow table. In this way, the OpenFlow switch uses an entry stored in the flow table as a process rule to forward a packet.") Uchida at [0064] ("As the information to be extracted to determine the process contents to be applied to the received packet by the packet process information extraction unit C12,

No.	'111 Patent Claim 27	Uchida
		information identical to that used in an OpenFlow switch in Non-Patent Documents 1 and 2 can be used. For example, the following information can be used: an ingress port (receiving port ID information); a source address, a destination address, and an upper protocol type of Ethernet (registered trademark); a VID (VLAN-ID) and a PCP (Priority Code Point) in TCI (Tag Control Information) of an IEEE802.1Q-compliant VLAN (Virtual LAN); a source address, a destination address, and an upper protocol type of the IP (Internet Protocol); and a source port and a destination port of the TCP (Transmission Control Protocol) or the UDP (User Datagram Protocol). Among the above information, the ingress port (receiving port ID information) of the packet forwarding apparatus can be obtained from the reception status information, and the other information can be obtained from header fields of the received packet, upper protocol headers of the packet, or the like.") Uchida at [0168] ("The present invention is applicable to a packet forwarding apparatus such as a HUB or a switch having a storage region for storing process rules.")

No.	'111 Patent Claim 28	Uchida
28	The method according to claim 1, wherein the packet network is an Internet Protocol (IP) network, and the packet is an IP packet.	Uchida discloses the method according to claim 1, wherein the packet network is an Internet Protocol (IP) network, and the packet is an IP packet. For example, Uchida discloses packets that have IP addresses and are sent via a TCP/IP communication. *See supra* at Claim 1. Uchida at [0059] ("Next, a first exemplary embodiment of the present invention will be described in detail with reference to the drawings. First, terms will be defined. A "flow" signifies communication from a certain source to destination. Namely, a "flow" signifies a group of forwarded packets or the for-warding itself. Each of the "source" and "destination" does not necessarily signify a single physical or logical address. Namely, communication from a certain subnet to a different subnet may be collectively deemed as a single flow. Thus, a flow may be a collection of more detailed flows, one example of which corresponds to a flow entry in which a plurality of flows are unified by describing part of destination addresses.

No.	'111 Patent Claim 28	Uchida
		source addresses, or the like with wildcards, as in the case of OpenFlow in Non-Patent Documents 1 and 2.") Uchida at [0064] ("As the information to be extracted to determine the process contents to be applied to the received packet by the packet process information extraction unit C12, information identical to that used in an OpenFlow switch in Non-Patent Documents 1 and 2 can be used. For example, the following information can be used: an ingress port (receiving port ID information); a source address, a destination address, and an upper protocol type of Ethernet (registered trademark); a VID (VLAN-ID) and a PCP (Priority Code Point) in TCI (Tag Control Information) of an IEEE802.1Q-compliant VLAN (Virtual LAN); a source address, a destination address, and an upper protocol type of the IP (Internet Protocol); and a source port and a destination port of the TCP (Transmission Control Protocol) or the UDP (User Datagram Protocol). Among the above information, the ingress port (receiving port ID information) of the packet forwarding apparatus can be obtained from the reception status information, and the other information can be obtained from header fields of the received packet, upper protocol headers of the packet, or the like.") Uchida at [0066] ("For example, the header fields inFIG.19 include: an ingress port of the packet forwarding apparatus; a source address, a destination address, and an upper protocol type of Ethernet (registered trademark); a YID and a priority (PCP; Priority Code Point) of an IEEE802.lQ-compliant YLAN; a source address, a destination address, and an upper protocol type of the IP; and a source port and a destination port of the TCP or the UDP. When a packet matching the contents of these header fields is received, process contents defined in the action field are executed. However, wildcards can be used in arbitrary information bits in the header fields.") Uchida at [0109] ("Next, a process rule corresponding to the received packet is searched for. The packet process information extrac-tion unit C12 transmits the extracted information (represent-ing that the ingress port is 1; the source address is 192.168. 0.10; the destination address is 192.168.1.10; and the protocol is TCP (the type is 0x0006)) to a process rule search unit C14. The process rule search unit C14 refers to the process rule storage unit C13. Since no process rule matching the packet exists, the process rule search unit C14 queries the control apparatus (controller) Dl (steps S104 and S106 in FIG. 3). The control apparatus (controller) Dl creates a pro-cess rule including process contents representing modifica-tion of the source

No.	'111 Patent Claim 28	Uchida
		IP and forward from a specified transport port and transmits the process rule to the packet forwarding apparatus Cl. The process rule search unit C14 registers the process rule received from the control apparatus (controller) Dl as a new entry in the process rule storage unit C13 (step S107 in FIG. 3). At this stage, as illustrated in FIG. 9, a new process rule is added to the process rule storage unit C13.") Uchida at [0072] ("The packet process unit C15 processes a received packet stored in the packet storage unit C11 in accordance with the actions (process contents) outputted from the process rule search unit C14. For example, upon receiving a Forward action for forwarding a packet, the packet process unit C15 requests the packet forwarding unit C16 to transmit the packet. In addition to the actions (process contents) outputted from the process rule search unit C14, the packet process unit C15 appropriately executes basic processes and the like relating to communication. For example, in the case of the first process rule in FIG. 20, the packet process unit C15 first changes the source IP address to 192.168.1.1 and next transmits the packet, as the actions (process contents). Simultaneously, as basic processes relating to communication, the packet process unit C15 also executes basic processes relating to the IP and the like, such as rewriting of the TTL (Time To Live) field, the checksum field, and the like in the IP headers, as needed. The basic processes relating to communication include determining the end of packet forwarding when the TTL value reaches 0 or the like.") 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.")

No.	'111 Patent Claim 29	Uchida
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.	Uchida 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. For example, Uchida discloses TCP (Transmission Control Protocol) packets that are sent in a TCP communication. *See supra* at Claim 28. 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.") 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.") 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.") 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.")

No.	'111 Patent Claim 29	Uchida
		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.") 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.") 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

No.	'111 Patent Claim 29	Uchida
		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.") 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).") 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.") 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).") 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

No.	'111 Patent Claim 29	Uchida
		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).") 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.") 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.") 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).") 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).")

No.	'111 Patent Claim 29	Uchida
		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.") 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.") 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.") 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.") 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.") 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).")

No.	'111 Patent Claim 29	Uchida
		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).") 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.") 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.") 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.") 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.")

No.	'111 Patent Claim 29	Uchida
		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).") 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).") 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 30	Uchida
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;	Uchida 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. For example, Uchida discloses a flow of multiple forwarded packets received at the OpenFlow switch. *See supra at* Claim 1, 1[c]. Uchida at [0059] ("Next, a first exemplary embodiment of the present invention will be described in detail with reference to the drawings. First, terms will be defined. A "flow"

No.	'111 Patent Claim 30	Uchida
		signifies communication from a certain source to destination. Namely, a "flow" signifies a group of forwarded packets or the for-warding itself. Each of the "source" and "destination" does not necessarily signify a single physical or logical address. Namely, communication from a certain subnet to a different subnet may be collectively deemed as a single flow. Thus, a flow may be a collection of more detailed flows, one example of which corresponds to a flow entry in which a plurality of flows are unified by describing part of destination addresses, source addresses, or the like with wildcards, as in the case of OpenFlow in Non-Patent Documents 1 and 2.") Uchida at [0061] ("The packet reception unit C10 receives a packet from another node and temporarily stores the received packet in the packet storage unit C11. In addition, the packet reception unit C10 can store reception status information representing an ingress port (receiving port ID information) and the like simultaneously. After storing the received packet, 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.")
30[b]	checking, by the network node, if any one of the one or more additional packets satisfies the criterion;	Uchida discloses checking, by the network node, if any one of the one or more additional packets satisfies the criterion. *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	Uchida 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. *See supra at* Claim 1[e], 30[a].

No.	'111 Patent Claim 30	Uchida
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.	Uchida 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. *See supra at* Claim 1[f], 30[a].

No.	'111 Patent Claim 31	Uchida
31[a]	The method according to claim 1, wherein the packet network is a Software Defined Network (SDN),	Uchida discloses the method according to claim 1, wherein the packet network is a Software Defined Network (SDN). For example, Uchida discloses communication between a switch and controller using the SDN OpenFlow protocol. *See supra* at Claim 1. Uchida at [0003] ("In recent years, a technique referred to as OpenFlow is proposed (see Non-Patent Document 1). In OpenFlow, communication is deemed as an end-to-end flow, and routing control, failure recovery, load distribution, and optimization are executed for each flow. FIG. 18 is a conceptual diagram of an OpenFlow switch. The OpenFlow switch, which serves as a forwarding node, includes a secure channel for communi-cation with an OpenFlow controller serving as a control appa-ratus. The OpenFlow switch operates in accordance with a flow table appropriately added or rewritten by the OpenFlow controller. In the flow table, a group of: matching rules (header fields) matched against packet headers; actions defin-ing process contents; and flow statistics information (counters) is defined for each flow (see FIGS. 19 and 20).") Uchida at [0005] ("For example, upon receiving a packet, the Open-Flow switch searches the flow table for an entry having matching rules (see header fields in FIGS. 19 and 20) that match header information of the received packet. As a result of the search, if an entry matching the received packet is found, the OpenFlow switch executes process contents described in action fields of the entry on the received packet. If, as a result of the search, no entry matching the

No.	'111 Patent Claim 31	Uchida
		received packet is found, the OpenFlow switch forwards the received packet to the OpenFlow controller via the secure channel to request the OpenFlow controller to determine a packet route based on the source/destination of the received packet. Upon receiving a flow entry realizing the route, the OpenFlow switch updates the flow table. In this way, the OpenFlow switch uses an entry stored in the flow table as a process rule to forward a packet.") Uchida at [0064] ("As the information to be extracted to determine the process contents to be applied to the received packet by the packet process information extraction unit C12, information identical to that used in an OpenFlow switch in Non-Patent Documents 1 and 2 can be used. For example, the following information can be used: an ingress port (receiving port ID information); a source address, a destination address, and an upper protocol type of Ethernet (registered trademark); a VID (VLAN-ID) and a PCP (Priority Code Point) in TCI (Tag Control Information) of an IEEE802.1Q-compliant VLAN (Virtual LAN); a source address, a destination address, and an upper protocol type of the IP (Internet Protocol); and a source port and a destination port of the TCP (Transmission Control Protocol) or the UDP (User Datagram Protocol). Among the above information, the ingress port (receiving port ID information) of the packet forwarding apparatus can be obtained from the reception status information, and the other information can be obtained from header fields of the received packet, upper protocol headers of the packet, or the like.") Uchida at [0104] ("Hereinafter, the description will be made, assuming that a packet forwarding apparatus C1 according to the present invention relays TCP communication between a computer (client) D10 and a server D20 that use network configurations illustrated in FIG. 7. In the example of FIG. 7, the computer D10 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 (controller) D1 is connected to the packet forwarding apparatus C1 via a dedicated channel and manages connection between the two networks. In the following description, the control apparatus (controller) D1 controls the packet forwarding apparatus C1 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 C17 monitors a

No.	'111 Patent Claim 31	Uchida
		protocol stack, including: fields in which the TCP is determined; and the FIN flag in the TCP header.")
31[b]	the packet is routed as part of a data plane and	Uchida discloses the packet is routed as part of a data plane. For example, Uchida discloses forwarding packets between devices via the OpenFlow switch. *See supra* at Claim 1. Uchida at [0003] ("In recent years, a technique referred to as OpenFlow is proposed (see Non-Patent Document 1). In OpenFlow, communication is deemed as an end-to-end flow, and routing control, failure recovery, load distribution, and optimization are executed for each flow. FIG. 18 is a conceptual diagram of an OpenFlow switch. The OpenFlow switch, which serves as a forwarding node, includes a secure channel for communi-cation with an OpenFlow controller serving as a control appa-ratus. The OpenFlow switch operates in accordance with a flow table appropriately added or rewritten by the OpenFlow controller. In the flow table, a group of: matching rules (header fields) matched against packet headers; actions defin-ing process contents; and flow statistics information (counters) is defined for each flow (see FIGS. 19 and 20).") Uchida at [0005] ("For example, upon receiving a packet, the Open-Flow switch searches the flow table for an entry having matching rules (see header fields in FIGS. 19 and 20) that match header information of the received packet. As a result of the search, if an entry matching the received packet is found, the OpenFlow switch executes process contents described in action fields of the entry on the received packet. If, as a result of the search, no entry matching the received packet is found, the OpenFlow switch forwards the received packet to the OpenFlow controller via the secure channel to request the OpenFlow controller to determine a packet route based on the source/destination of the received packet. Upon receiving a flow entry realizing the route, the OpenFlow switch updates the flow table. In this way, the OpenFlow switch uses an entry stored in the flow table as a process rule to forward a packet.")

No.	'111 Patent Claim 31	Uchida
		Uchida at [0064] ("As the information to be extracted to determine the process contents to be applied to the received packet by the packet process information extraction unit C12, information identical to that used in an OpenFlow switch in Non-Patent Documents 1 and 2 can be used. For example, the following information can be used: an ingress port (receiving port ID information); a source address, a destination address, and an upper protocol type of Ethernet (registered trademark); a VID (VLAN-ID) and a PCP (Priority Code Point) in TCI (Tag Control Information) of an IEEE802.1Q-compliant VLAN (Virtual LAN); a source address, a destination address, and an upper protocol type of the IP (Internet Protocol); and a source port and a destination port of the TCP (Transmission Control Protocol) or the UDP (User Datagram Protocol). Among the above information, the ingress port (receiving port ID information) of the packet forwarding apparatus can be obtained from the reception status information, and the other information can be obtained from header fields of the received packet, upper protocol headers of the packet, or the like.") Uchida at [0104] ("Hereinafter, the description will be made, assuming that a packet forwarding apparatus C1 according to the present invention relays TCP communication between a computer (client) D10 and a server D20 that use network configurations illustrated in FIG. 7. In the example of FIG. 7, the computer D10 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 (controller) D1 is connected to the packet forwarding apparatus C1 via a dedicated channel and manages connection between the two networks. In the following description, the control apparatus (controller) D1 controls the packet forwarding apparatus C1 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 C17 monitors a protocol stack, including: fields in which the TCP is determined; and the FIN flag in the TCP header.")

No.	'111 Patent Claim 31	Uchida
31[c]	the network node communication with the controller serves as a control plane.	Uchida discloses the network node communication with the controller serves as a control plane. For example, Uchida discloses communication between the OpenFlow switch and OpenFlow control apparatus. Uchida at [0003] ("In recent years, a technique referred to as OpenFlow is proposed (see Non-Patent Document 1). In OpenFlow, communication is deemed as an end-to-end flow, and routing control, failure recovery, load distribution, and optimization are executed for each flow. FIG. 18 is a conceptual diagram of an OpenFlow switch. The OpenFlow switch, which serves as a forwarding node, includes a secure channel for communi-cation with an OpenFlow controller serving as a control appa-ratus. The OpenFlow switch operates in accordance with a flow table appropriately added or rewritten by the OpenFlow controller. In the flow table, a group of: matching rules (header fields) matched against packet headers; actions defin-ing process contents; and flow statistics information (counters) is defined for each flow (see FIGS. 19 and 20).") Uchida at [0005] ("For example, upon receiving a packet, the Open-Flow switch searches the flow table for an entry having matching rules (see header fields in FIGS. 19 and 20) that match header information of the received packet. As a result of the search, if an entry matching the received packet is found, the OpenFlow switch executes process contents described in action fields of the entry on the received packet. If, as a result of the search, no entry matching the received packet is found, the OpenFlow switch forwards the received packet to the OpenFlow controller via the secure channel to request the OpenFlow controller to determine a packet route based on the source/destination of the received packet. Upon receiving a flow entry realizing the route, the OpenFlow switch updates the flow table. In this way, the OpenFlow switch uses an entry stored in the flow table as a process rule to forward a packet.") Uchida at [0049] ("The packet forwarding apparatus further comprises a packet storage unit that stores a received packet. In addition, the packet forwarding apparatus can cause the end determination information extraction unit to extract, from the packet storage unit, information for identifying process rules corresponding to a flow to which a received packet belongs and

No.	'111 Patent Claim 31	Uchida
		a reverse-direction flow of the flow, respectively, and can cause the flow end check unit to determine whether to delete the process rules in both directions.") Uchida at [0050] ("The process rule search unit C14 determines actions (process contents) to be applied to the received packet, based on the information transmitted from the packet process information extraction unit C12 and outputs the actions to the packet process unit C15. More specifically, the process rule search unit C14 searches the process rule storage unit C13 for a process rule matching the information extracted from the received packet transmitted from the packet process information extraction unit C12 and outputs the process rule to the packet process unit C15. If a process rule matching the received packet exists, the process rule search unit C14 extracts actions (process contents) defined in the action field of the process rule and outputs the actions to the packet process unit C15. If no process rule matching the received packet is found, the process rule search unit C14 transmits the received packet or information extracted from the received packet to a predetermined control apparatus (controller; see FIG. 18) to request creation and transmission of a process rule. Upon receiving a process rule from the predetermined control apparatus (controller; see FIG. 18), the process rule search unit C14 registers the received process rule in the process rule storage unit C13. Next, the process rule search unit C14 extracts actions (process contents) from the process rule and outputs the actions to the packet process unit C15.") Uchida at [0064] ("As the information to be extracted to determine the process contents to be applied to the received packet by the packet process information extraction unit C12, information identical to that used in an OpenFlow switch in Non-Patent Documents 1 and 2 can be used. For example, the following information can be used: an ingress port (receiving port ID information); a source address, a destination address, and an upper protocol type of Ethernet (registered trademark); a VID (VLAN-ID) and a PCP (Priority Code Point) in TCI (Tag Control Information) of an IEEE802.1Q-compliant VLAN (Virtual LAN); a source address, a destination address, and an upper protocol type of the IP (Internet Protocol); and a source port and a destination port of the TCP (Transmission Control Protocol) or the UDP (User Datagram Protocol). Among the above information, the ingress port (receiving port ID information) of the packet forwarding apparatus can be obtained from the reception status information, and the other information can be obtained from header fields of the received packet, upper protocol headers of the packet, or the like.")

Chart for U.S. Patent 10,652,111 ("the '111 Patent")
VMware NSX Network Virtualization ("VMware NSX System")

The VMware NSX network virtualization product family ("VMware NSX System"), which was publicly disclosed, sold, or on sale no later than April 22, 2014, constitutes prior art to U.S. Patent No. 10,652,111 to Barsheshet et al. 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 the VMware NSX System meet each element of those claims, (2) invalid under AIA-35 U.S.C. § 102 (a) because the references describing the VMware NSX System disclose every limitation of every Asserted Claim, and/or (3) 35 U.S.C. § 103 because the VMware NSX 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 VMware NSX System comprises switches and servers running VMware NSX virtualization solutions from before April 22, 2014, such as VMware's vSphere suite of products including VMware ESXi and vCenter Server. The following quotations and diagrams come from documentation describing the VMware NSX System and its functionalities that were published prior to April 22, 2014.

On information and belief, the VMware NSX System is described at least in the following documents and other materials published before April 22, 2014 cited in this chart:

- https://blogs.vmware.com/networkvirtualization/2013/08/vmware-nsx.html/ ("Introducing VMware NSX")
- https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/downloads/eula/product-guide/2013/vmware-product-guide-13-dec.pdf ("VMware Product Guide")
- https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/products/nsx/vmware-nsx-on-cisco-n7kucs-design-guide.pdf ("VMware NSX Network Virtualization Design Guide")
- https://blogs.vmware.com/networkvirtualization/2014/02/vmware-nsx-pex-2.html/ ("VMware NSX: Helping Make the Software-Defined Data Center Real in 2014")
- https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/products/nsx/vmw-nsx-network-virtualization-design-guide.pdf ("NSX Design Guide")

- https://blogs.vmware.com/networkvirtualization/2013/11/distributed-virtual-and-physical-routing-in-vmware-nsx-for-vsphere.html/ ("Distributed virtual and physical routing in VMware NSX for vSphere")
- https://blog.ipspace.net/2013/11/layer-2-and-layer-3-switching-in-vmware.html ("Switching in VMware NSX")
- https://blog.ipspace.net/2013/08/nicira-nvp-control-plane.html ("Nicira NVP Control Plane")
- VMware documents (VMW_ORCT_CSCO_000001 - VMW_ORCT_CSCO_005234)

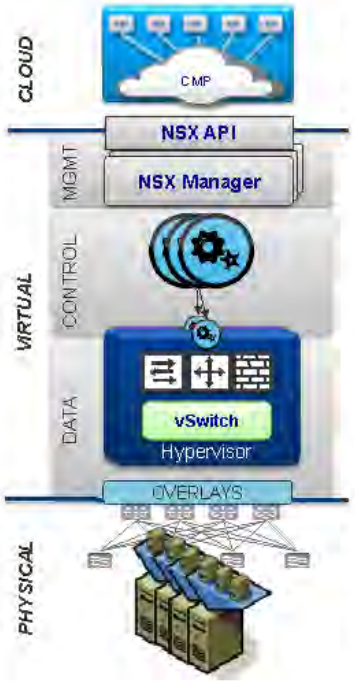
Motivations to combine the disclosures in VMware NSX System with disclosures in other publications known in the art, as explained in this chart, include at least the similarity in subject matter between the references to the extent they concern methods of relating to routing certain network traffic to 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.

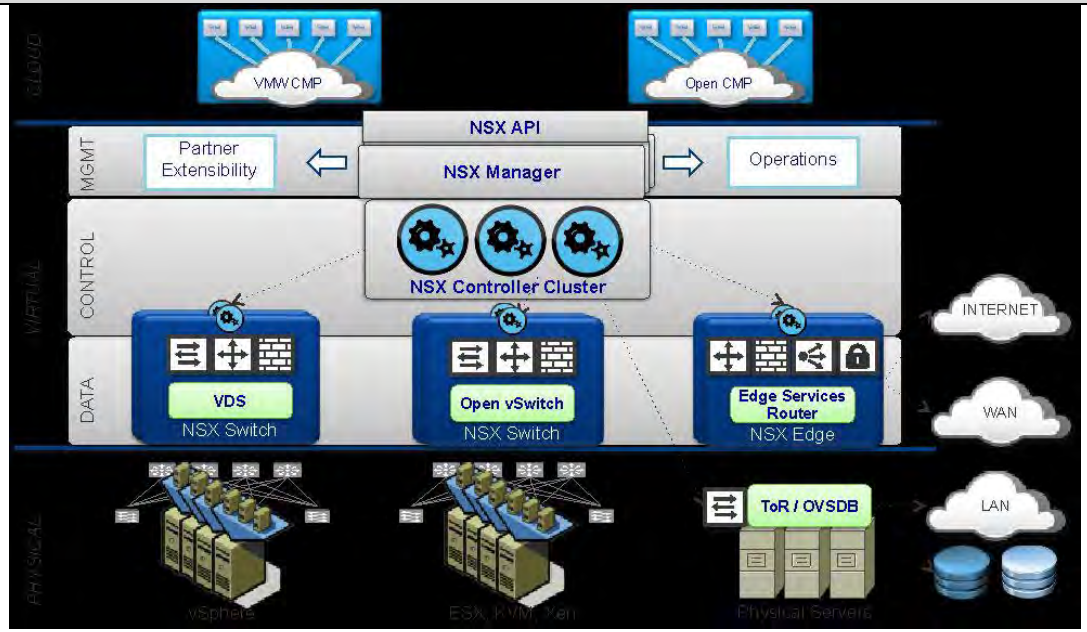
These invalidity contentions are based on Defendant's present understanding of the asserted claims, and Orckit's apparent construction of the claims in its November 3, 2022 Disclosure of Asserted Claims and Infringement Contentions Pursuant to P.R. 3-1, and Orckit's January 19, 2023 First Amended Disclosure 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.

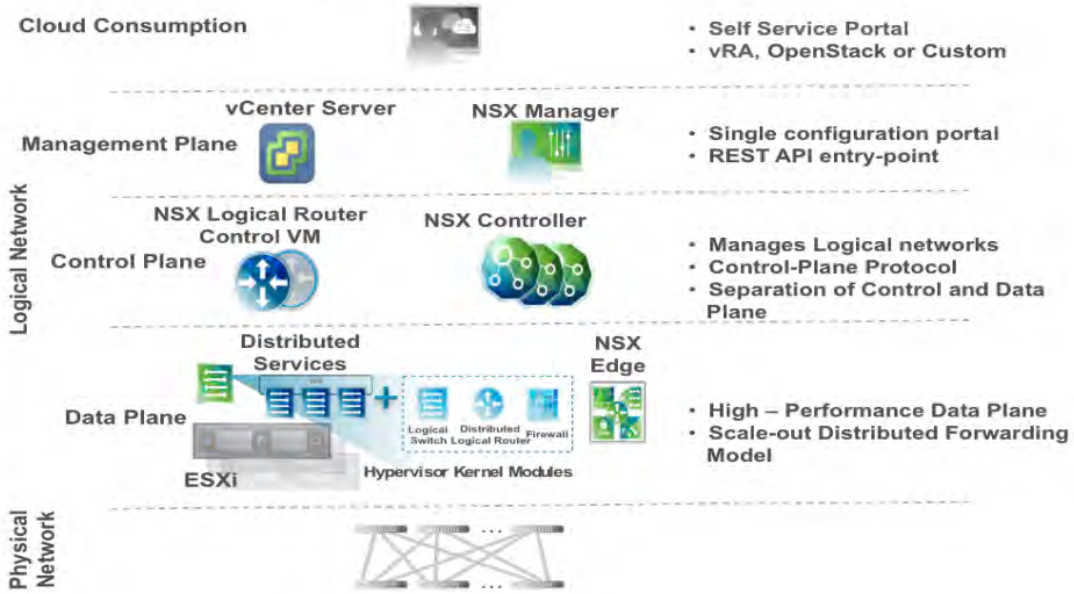
The following contentions are subject to revision and amendment pursuant to Federal Rule of Civil Procedure 26(e), the Local Rules, and the Orders of record in this matter subject to further investigation and discovery regarding the prior art and the Court's construction of the claims at issue.

No.	'111 Patent Claim 1	VMware NSX
1[preamble]	A method for use with a packet	VMware NSX 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.

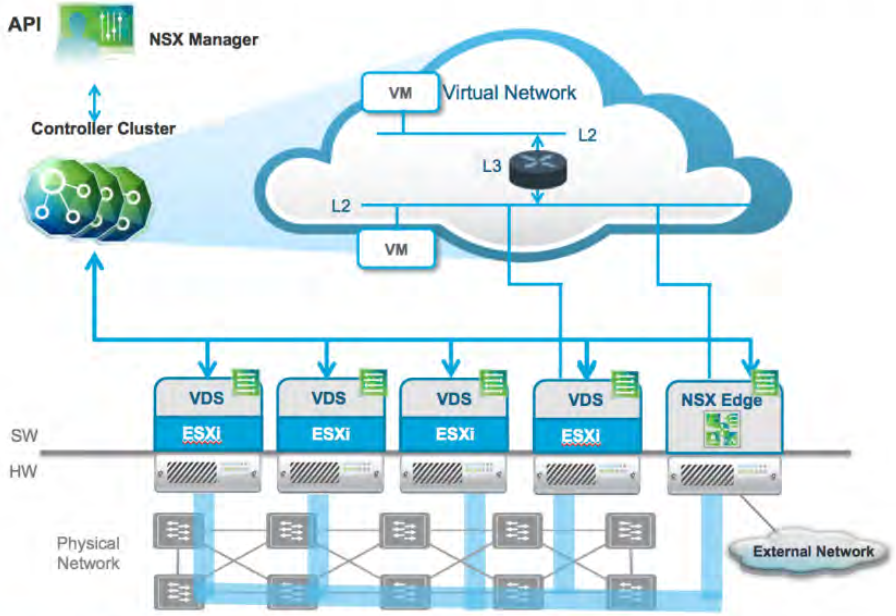
No.	'111 Patent Claim 1	VMware NSX
	network including a network node for transporting packets between first and second entities under control of a controller that is external to the network node, the method comprising:	For example, VMware NSX System discloses a manner in which data is send between ESXi hosts governed by an external NSX controller separate from a virtual switch. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. Introducing VMware NSX at 3-4 **VMware NSX Architecture and Design** As depicted in the diagram below, the VMware NSX solution resides in the virtualization layer, providing L2-L7 network services to the cloud consumption layer above, and mapping these services onto the physical infrastructure below. Independence between the cloud and virtualization layers is achieved by providing a REST API exposing the network services to any upstream provisioning platform. Likewise, vSwitch overlays such as VXLAN provide independence to deploy NSX on any physical IP network infrastructure. This brings us to the network virtualization layer. For each capability, be it switching, routing or firewall, the services are provided via NSX APIs, and realized using a three-tiered design pattern encompassing the management plane, control plane and data plane. The NSX Manager internally maps the APIs onto the control plane. **The controller cluster is the work horse of the system**, handling real time mapping between the system's desired state and the running state, which it communicates to the control plane agent(s) present per hypervisor. The local information is now used to set up the appropriate switching, routing, or firewall tables and contexts. The appropriate data plane function now proceeds with high-performance, in a scale out fashion across the virtual plane.

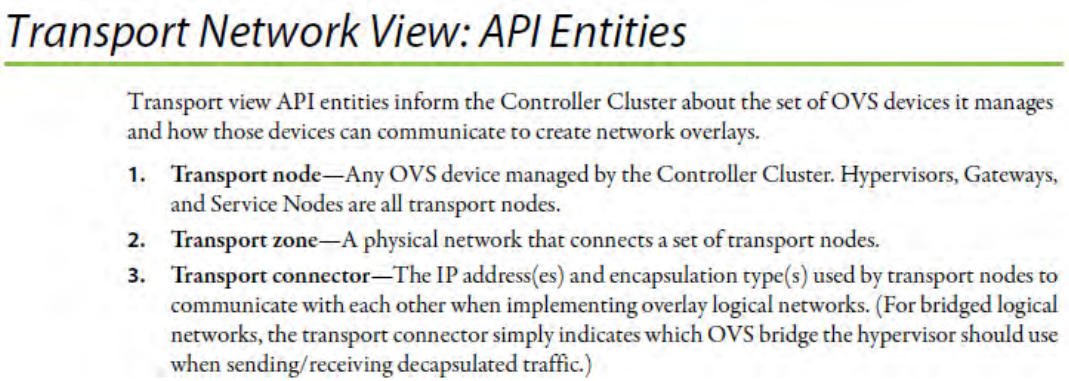
No.	'111 Patent Claim 1	VMware NSX
		 **NSX: architectural context & design pattern**: Cloud Management Platform consumes services The REST **API** abstracts underlying services The **Manager** cluster maps services to the controllers The control plane consists of a master **controller** cluster delegating tasks to control plane agents in each hypervisor (or controlled device) The agent performs the local task of activating the **data plane** action in the hypervisor e.g. switching, routing or firewalls **Overlays** de-couple the virtual plane from the underlying physical network fabric The network fabric provides connectivity to physical servers, hosting multiple VMs connected to a programmable vSwitch. IP connectivity is the only requirement of the physical network fabric

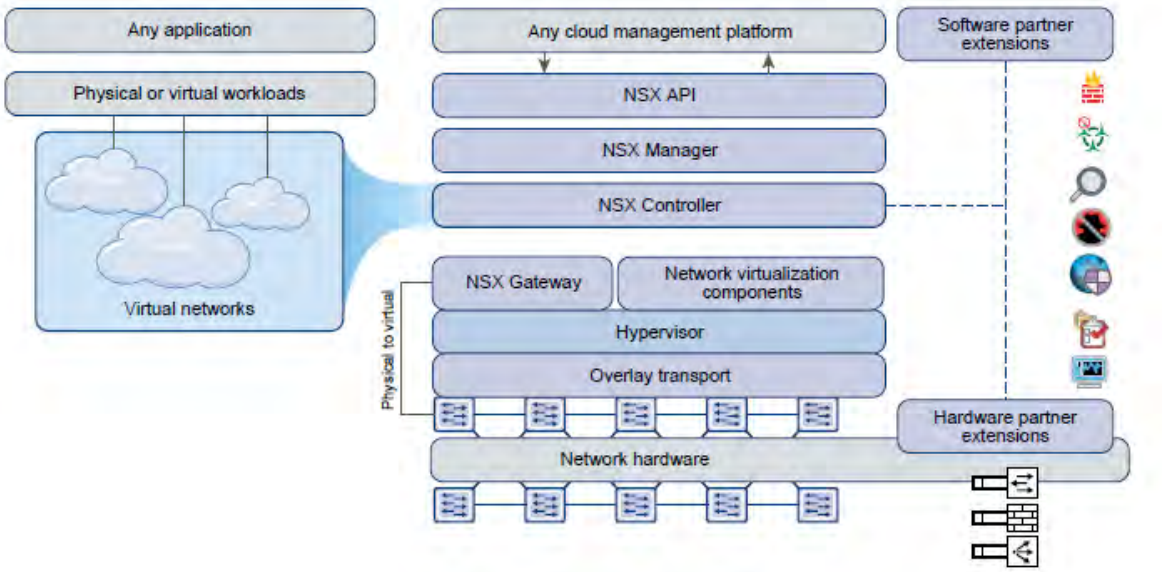
No.	'111 Patent Claim 1	VMware NSX
		 NSX Design Guide at 8

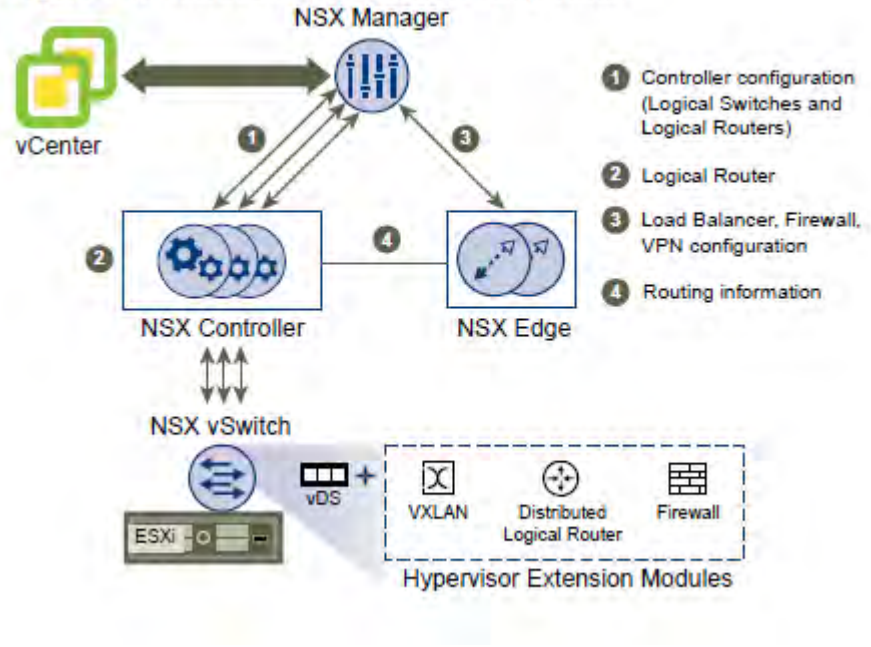
No.	'111 Patent Claim 1	VMware NSX
		2 Introduction to Network Virtualization 2.1 Overview of NSX Network Virtualization Solution An NSX deployment consists of a data plane, control plane, and management plane, as shown in Figure 4.  Figure 4 - NSX Components The NSX architecture has built in separation of data, control, and management layers. The NSX components that maps to each layer and each layer's architectural properties are shown in above Figure 4. This separation allows the architecture to grow and scale without impacting workload. Each layer and its specific overview are described below. NSX Design Guide at 11

No.	'111 Patent Claim 1	VMware NSX
		2.1.2 Control Plane The NSX controller is a key part of the NSX control plane. In a vSphere environment with the vSphere Distributed Switch (VDS), the controller enables multicast free VXLAN and control plane programming of elements such as the Distributed Logical Routing (DLR). Stability and reliability of data transport are central concerns in networking. The NSX controller is a part of the control plane; it is logically separated from all data plane traffic. To further enhance high availability and scalability, NSX controller nodes are deployed in a cluster of odd number instances. In addition to controller, the control VM, provides the routing control plane that allows the local forwarding in ESXi and allows dynamic routing between ESXI and north-south routing provided by Edge VM. It is critical to understand that data plane traffic never traverses the control plane component. NSX Design Guide at 13

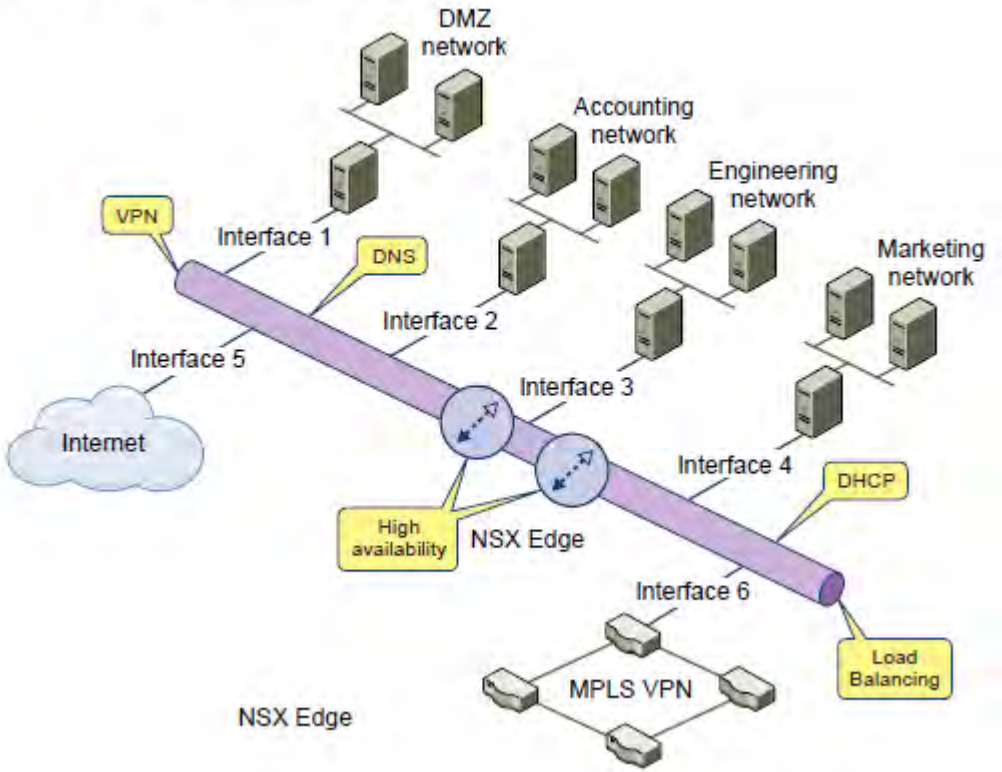
No.	'111 Patent Claim 1	VMware NSX
		3 NSX Functional Components NSX logical networks leverage two types of access layer entities – the hypervisor access layer and the gateway access layer. The hypervisor access layer represents the point of attachment to the logical networks for virtual endpoints (e.g., VM, service-point). The gateway access layer provides L2 and L3 connectivity into the logical space for devices deployed in the physical network infrastructure.  Figure 7 – NSX Functional Components The NSX platform consists of multiple components, responsible for platform management, traffic control, and service delivery. The following sections detail their functional and operational specifics.

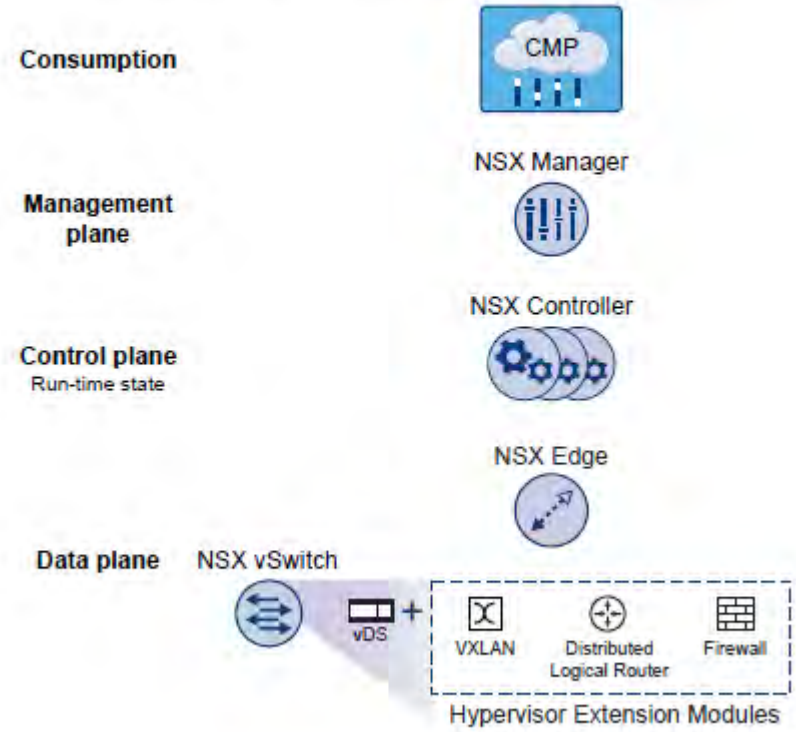
No.	'111 Patent Claim 1	VMware NSX
		VMW_ORCT_CSCO_ 15 "The new HP-VMware networking solution will federate the HP Virtual Application Networks SDN Controller with the VMware NSX™ network virtualization platform to provide customers with an integrated approach to automating their physical and virtual network infrastructure. The networking solution will provide a centralized view, unified automation, visibility and control of the complete data center network, improving agility, monitoring and troubleshooting." VMW_ORCT_CSCO_ 59  *Transport Network View: API Entities* Transport view API entities inform the Controller Cluster about the set of OVS devices it manages and how those devices can communicate to create network overlays. 1. **Transport node**—Any OVS device managed by the Controller Cluster. Hypervisors, Gateways, and Service Nodes are all transport nodes. 2. **Transport zone**—A physical network that connects a set of transport nodes. 3. **Transport connector**—The IP address(es) and encapsulation type(s) used by transport nodes to communicate with each other when implementing overlay logical networks. (For bridged logical networks, the transport connector simply indicates which OVS bridge the hypervisor should use when sending/receiving decapsulated traffic.) VMW_ORCT_CSCO_ 1523

No.	'111 Patent Claim 1	VMware NSX
		 VMW_ORCT_CSCO_1629

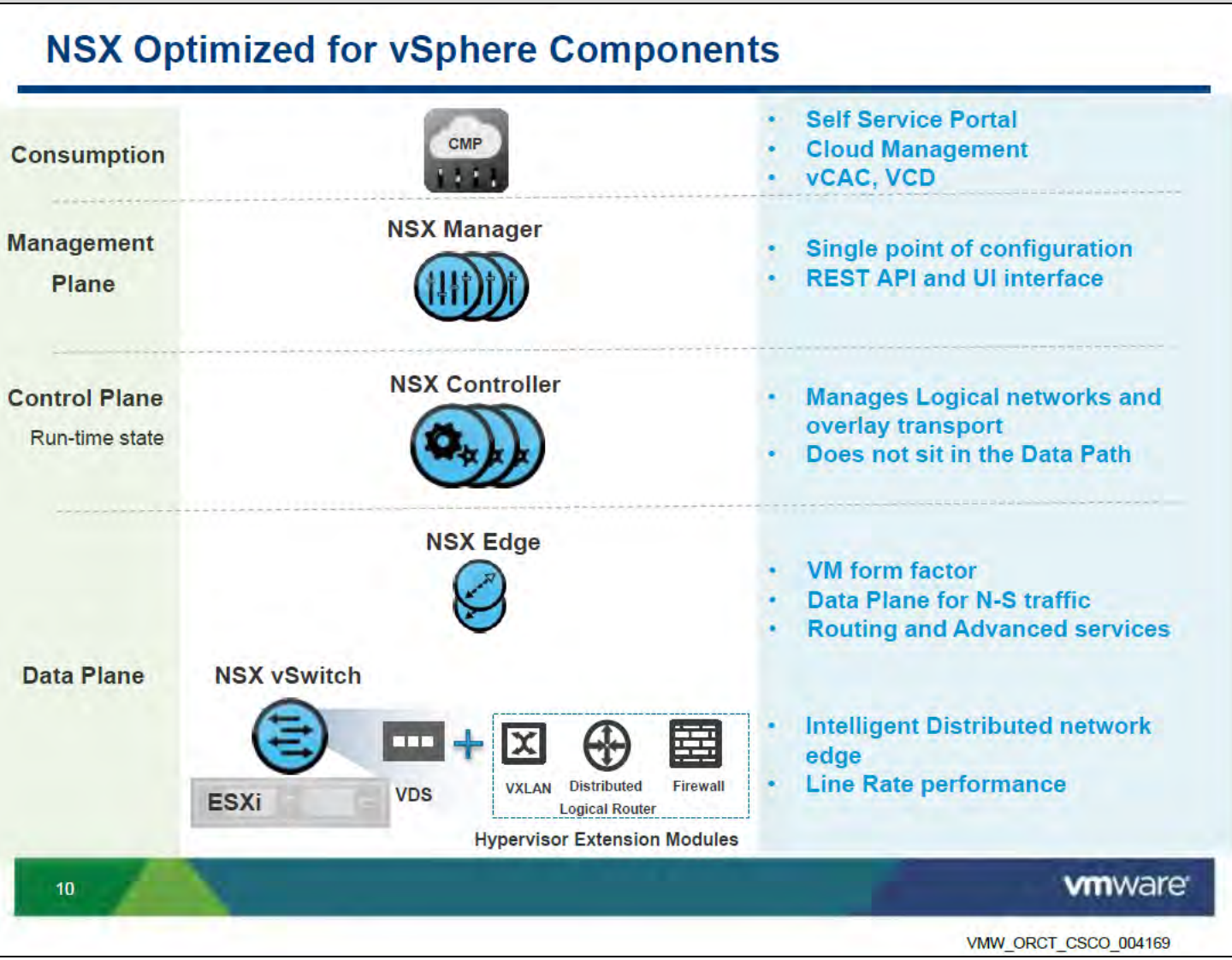
No.	'111 Patent Claim 1	VMware NSX
		Figure 1-1. Interaction between NSX components  VMW_ORCT_CSCO_1630

No.	'111 Patent Claim 1	VMware NSX
		NSX Controller NSX controller is an advanced distributed state management system that controls virtual networks and overlay transport tunnels. NSX controller is the central control point for all logical switches within a network and maintains information of all virtual machines, hosts, logical switches, and VXLANs. The controller supports two new logical switch control plane modes, Unicast and Hybrid. These modes decouple NSX from the physical network. VXLANs no longer require the physical network to support multicast in order to handle the Broadcast, Unknown unicast, and Multicast (BUM) traffic within a logical switch. The unicast mode replicates all the BUM traffic locally on the host and requires no physical network configuration. In the hybrid mode, some of the BUM traffic replication is offloaded to the first hop physical switch to achieve better performance. VMW_ORCT_CSCO_1630 **NSX Edge** NSX Edge provides network edge security and gateway services to isolate a virtualized network. You can install an NSX Edge either as a logical (distributed) router or as a services gateway. The NSX Edge logical (distributed) router provides East-West distributed routing with tenant IP address space and data path isolation. Virtual machines or workloads that reside on the same host on different subnets can communicate with one another without having to traverse a traditional routing interface. The NSX Edge gateway connects isolated, stub networks to shared (uplink) networks by providing common gateway services such as DHCP, VPN, NAT, dynamic routing, and Load Balancing. Common deployments of NSX Edge include in the DMZ, VPN Extranets, and multi-tenant Cloud environments where the NSX Edge creates virtual boundaries for each tenant. VMW_ORCT_CSCO_1631

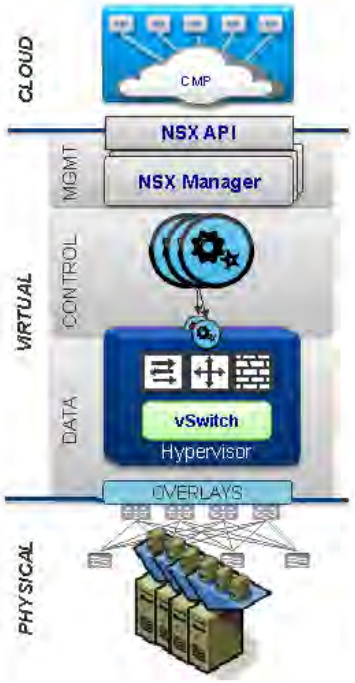
No.	'111 Patent Claim 1	VMware NSX
		Figure 1-2. Multi-Interface Edge  VMW_ORCT_CSCO_2160

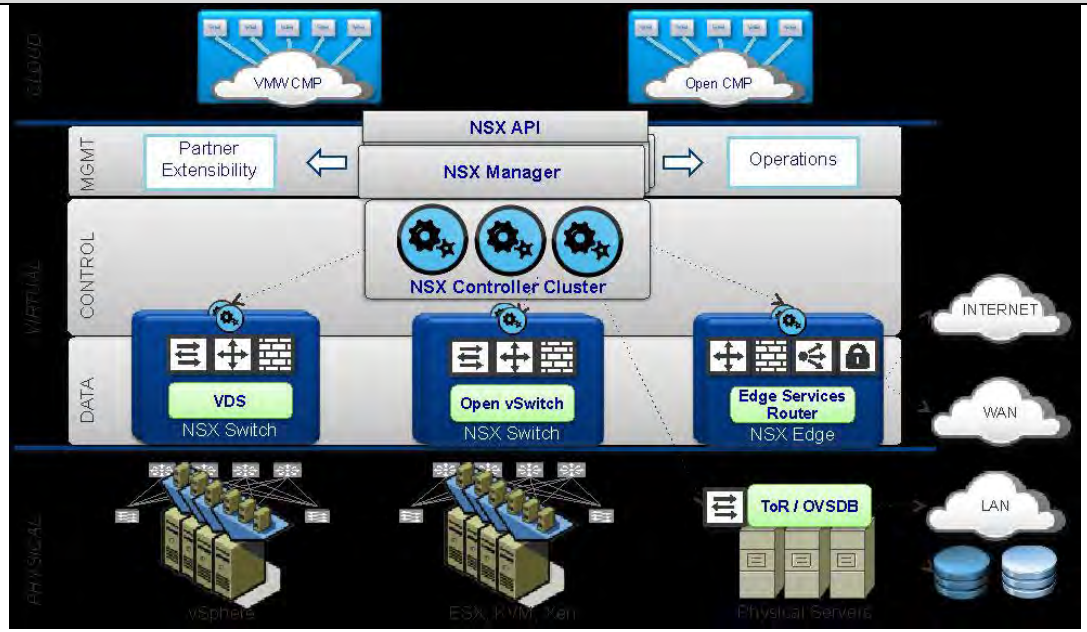
No.	'111 Patent Claim 1	VMware NSX
		NSX Components This section describes the components of the NSX solution.  VMW_ORCT_CSCO_2161

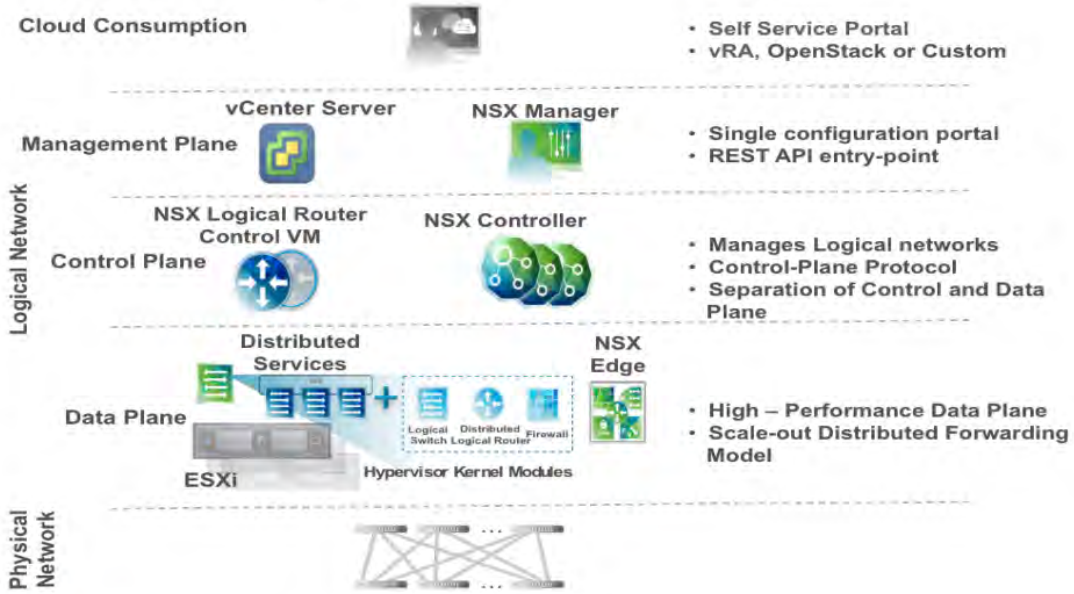
No.	'111 Patent Claim 1	VMware NSX
		Control Plane The NSX control plane runs in the NSX Controller cluster. NSX Controller is an advanced distributed state management system that provides control plane functions for NSX logical switching and routing functions. It is the central control point for all logical switches within a network and maintains information about all hosts, logical switches (VXLANs), and distributed logical routers. The controller cluster is responsible for managing the distributed switching and routing modules in the hypervisors. The controller does not have any dataplane traffic passing through it. Controller nodes are deployed in a cluster of three members to enable high-availability and scale. Any failure of the controller nodes does not impact any data-plane traffic. NSX Controllers work by distributing network information to hosts. To achieve a high level of resiliency the NSX Controller is clustered for scale out and HA. NSX Controllers must be deployed in a three-node cluster. The three virtual appliances provide, maintain, and update the state of all network functioning within the NSX domain. NSX Manager is used to deploy NSX Controller nodes. The three NSX Controller nodes form a control cluster. The controller cluster requires a quorum (also called a majority) in order to avoid a "split-brain scenario." In a split-brain scenario, data inconsistencies originate from the maintenance of two separate data sets that overlap. The inconsistencies can be caused by failure conditions and data synchronization issues. Having three controller nodes ensures data redundancy in case of failure of one NSX Controller node. VMW_ORCT_CSCO_ 4169

No.	'111 Patent Claim 1	VMware NSX
		

No.	'111 Patent Claim 1	VMware NSX
1[a]	sending, by the controller to the network node over the packet network, an instruction and a packet-applicable criterion;	VMware NSX System discloses sending, by the controller to the network node over the packet network, an instruction and a packet-applicable criterion. For example, VMware NSX System discloses programming the vSwitch by sending routing rules and tables with specific packet header information from the NSX controller. Introducing VMware NSX at 3-4 **VMware NSX Architecture and Design** As depicted in the diagram below, the VMware NSX solution resides in the virtualization layer, providing L2-L7 network services to the cloud consumption layer above, and mapping these services onto the physical infrastructure below. Independence between the cloud and virtualization layers is achieved by providing a REST API exposing the network services to any upstream provisioning platform. Likewise, vSwitch overlays such as VXLAN provide independence to deploy NSX on any physical IP network infrastructure. This brings us to the network virtualization layer. For each capability, be it switching, routing or firewall, the services are provided via NSX APIs, and realized using a three-tiered design pattern encompassing the management plane, control plane and data plane. The NSX Manager internally maps the APIs onto the control plane. **The controller cluster is the work horse of the system**, handling real time mapping between the system's desired state and the running state, which it communicates to the control plane agent(s) present per hypervisor. The local information is now used to set up the appropriate switching, routing, or firewall tables and contexts. The appropriate data plane function now proceeds with high-performance, in a scale out fashion across the virtual plane.

No.	'111 Patent Claim 1	VMware NSX
		 **NSX: architectural context & design pattern**: Cloud Management Platform consumes services The REST **API** abstracts underlying services The **Manager** cluster maps services to the controllers The control plane consists of a master **controller** cluster delegating tasks to control plane agents in each hypervisor (or controlled device) The agent performs the local task of activating the **data plane** action in the hypervisor e.g. switching, routing or firewalls **Overlays** de-couple the virtual plane from the underlying physical network fabric The network fabric provides connectivity to physical servers, hosting multiple VMs connected to a programmable vSwitch. IP connectivity is the only requirement of the physical network fabric

No.	'111 Patent Claim 1	VMware NSX
		 NSX Design Guide at 8

No.	'111 Patent Claim 1	VMware NSX
		2 Introduction to Network Virtualization 2.1 Overview of NSX Network Virtualization Solution An NSX deployment consists of a data plane, control plane, and management plane, as shown in Figure 4.  Figure 4 - NSX Components The NSX architecture has built in separation of data, control, and management layers. The NSX components that maps to each layer and each layer's architectural properties are shown in above Figure 4. This separation allows the architecture to grow and scale without impacting workload. Each layer and its specific overview are described below.

No.	'111 Patent Claim 1	VMware NSX
		NSX Design Guide at 11 **2.1.2 Control Plane** The NSX controller is a key part of the NSX control plane. In a vSphere environment with the vSphere Distributed Switch (VDS), the controller enables multicast free VXLAN and control plane programming of elements such as the Distributed Logical Routing (DLR). Stability and reliability of data transport are central concerns in networking. The NSX controller is a part of the control plane; it is logically separated from all data plane traffic. To further enhance high availability and scalability, NSX controller nodes are deployed in a cluster of odd number instances. In addition to controller, the control VM, provides the routing control plane that allows the local forwarding in ESXi and allows dynamic routing between ESXI and north-south routing provided by Edge VM. It is critical to understand that data plane traffic never traverses the control plane component. NSX Design Guide at 15 **3.1.2 Controller Cluster** The controller cluster in the NSX platform is the control plane component responsible for managing the hypervisor switching and routing modules. The controller cluster consists of controller nodes that manage specific logical switches. The use of controller cluster in managing VXLAN based logical switches eliminates the need for multicast configuration at the physical layer for VXLAN overlay. NSX Design Guide at 25

No.	'111 Patent Claim 1	VMware NSX
		3.1.7 NSX Distributed Firewall (DFW) The NSX DFW provides L2-L4 stateful firewall services to any workload in the NSX environment. DFW runs in the kernel space and provides near line rate network traffic protection. DFW performance and throughput scale linearly through addition of new ESXi hosts. DFW is activated as soon as the host preparation process is completed. If a VM does not require DFW service, it can be added in the exclusion list functionality. By default, NSX Manager, NSX Controllers, and Edge services gateways are automatically excluded from DFW function. One DFW instance is created per VM vNIC; for example, if a new VM with 3 vNICs is created, 3 instances of DFW will be allocated to this VM. Configuration of these DFW instances can be identical or different based on the "apply to" setting. When a DFW rule is created, the user can select a Point of Enforcement (PEP) for this rule, with options varying from Logical Switch to vNIC. By default, "apply to" is not selected, so DFW rules are propagated down to all the ESXi hosts which are part of the NSX domain that have been prepared for enforcement and applied to all the connected virtual machines. DFW policy rules can be written in 2 ways, using L2 rules (Ethernet) or L3/L4 rules. • L2 rules are mapped to L2 in the OSI model; only MAC addresses can be used in the source and destination fields and only L2 protocols can be used in the service fields (e.g., ARP). • L3/L4 rules are mapped to L3/L4 in the OSI model; policy rules can be written using IP addresses and TCP/UDP ports. It is important to remember that L2 rules are always enforced before L3/L4 rules. If the L2 default policy rule is modified to 'block', and then all L3/L4 traffic will be blocked as well by DFW (e.g., pings would fail).

No.	'111 Patent Claim 1	VMware NSX
		NSX Design Guide at 26-27

No.	'111 Patent Claim 1	VMware NSX
		The DFW system architecture is based on 3 distinct entities, each with a clearly defined role. • **vCenter Server:** vCenter Server is the management plane of the solution. DFW policy rules are created in the vSphere Web client. Any vCenter container can be used in the source/destination field of the policy rule: cluster, VDS port-group, Logical Switch, VM, vNIC, Resource Pool, etc. • **NSX Manager:** NSX manager is the control plane of the solution. It receives rules from the vCenter Server and stores them in its central database. NSX manager then pushes DFW rules down to all ESXi hosts that have been prepared for enforcement. Backup of DFW security policy rule table is performed each time the table is modified and published. NSX 26 [redacted] manager can also receive DFW rules directly from REST API calls in deployments where a cloud management system is used for security automation. • **ESXi Host:** ESXi host is the data plane of the solution. DFW rules are received from the NSX manager and translated into kernel space for real-time execution. VM network traffic is inspected and enforced per ESXi host. As an example, VM1 is located on ESXi host 1 and sends packets to VM2 that is located on ESXi host 2. Policy enforcement is done on ESXi host 1 for egress traffic when packets leave VM1 and then on ESXi host 2 for ingress traffic destined to VM2.

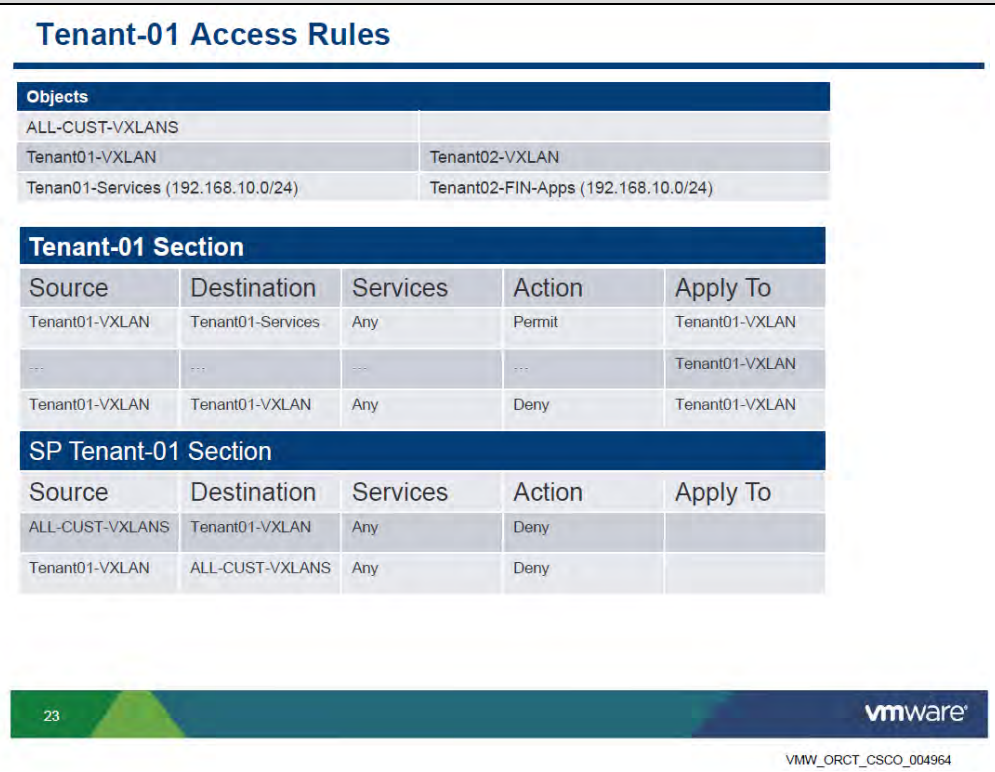
No.	'111 Patent Claim 1	VMware NSX
		NSX Design Guide at 29 Traffic redirection to third party vendor provides the capability to steer a particular type of traffic to a selected partner services VM. For instance, web traffic from the Internet to a set of Apache or Tomcat servers can be redirected to an L4-L7 deep packet inspection firewall for advanced protection. Traffic redirection is defined under Service Composer/Security Policy for NSX version 6.0 or under Partner Security Services tab of the DFW menu in NSX version 6.1. The Partner Security Services tab provides a powerful and easy to use user interface to define what type of traffic needs to be redirected to which partner services. It follows the same policy definition construct as DFW, providing the same options for source field, destination field and services field. The only difference is in the action field; instead of Block/Allow/Reject, a user can select between redirect/no redirect followed by a partner list. Any partner that has been registered with NSX can be successfully deployed on the platform. Additionally, log options can be enabled for this traffic redirection rule. The DFW instance on an ESXi host contains 2 separate tables. The rule table is used to store all policy rules, while the connection tracker table caches flow entries for rules with permit actions. One DFW instance is permitted per VM vNIC. A specific flow is identified by the 5-tuple information consisting source IP address, destination IP address, protocols, L4 source port, and L4 destination port fields. By default, DFW does not perform a lookup on L4 source port, but it can be configured to do so by defining a specific policy rule. NSX Design Guide at 33

No.	'111 Patent Claim 1	VMware NSX
		4 NSX Functional Services In the context of the NSX architecture, logical networking functionality (e.g., switching, routing, security) can be viewed as having a packet forwarding pipeline that implements multiple features, including L2/L3 forwarding, security filtering, and packet queuing. Unlike a physical switch or router, NSX does not rely on a single device to implement the entire forwarding pipeline. Instead, the NSX controller creates packet processing rules on all relevant ESXi hosts to mimic how a single logical device would handle data. Those hypervisors can be thought of as line cards that implement a portion of the logical network function pipeline. The physical network connecting the ESXi hosts acts as the backplane that carries packets from one line card to the other. NSX Design Guide at 41 **4.2.2 Populating the Controller Tables** Controller tables handle information essential for L2 unicast communication. Control plane communication between ESXi hosts and the controller cluster is used to populate the VTEP, MAC, and ARP tables on controller nodes. This process is detailed in Figure 32. NSX Design Guide at 52

No.	'111 Patent Claim 1	VMware NSX
		Distributed routing is provided by a logical element called Distributed Logical Router (DLR). The DLR is essentially a router with directly connected interfaces to all hosts where VM connectivity is required. The supervisory function (i.e., control plane) to control the forwarding is imported from a control VM. The DLR consists of two primary components: • **Control Plane:** The control plane is provided by the DLR Control VM and NSX controller. The control plane supports dynamic routing protocols (e.g., BGP, OSPF), exchanges routing updates with the next layer 3 hop device (e.g., NSX Edge), and communicates with the NSX manager and the controller cluster. High availability for the DLR control VM is supported through an active/standby configuration. • **Data Plane:** DLR kernel modules (e.g., VIBs) are installed on the ESXi hosts part of the NSX domain. The kernel modules are similar to the line cards in a modular chassis supporting layer 3 routing. These kernel modules have a routing information base (RIB) that is pushed through the controller cluster. The traditional data plane functionality of route and ARP lookups is performed by the kernel modules. The kernel modules are equipped with logical interfaces (LIFs) connecting to the different logical switches. Each LIF has an IP address representing the default IP gateway for its logical L2 segment as well as a vMAC address. The IP address is unique per LIF and remains same where the logical switch exists. The vMAC associated with each LIF remains the consistent in each hypervisor as well and thus during the vMotion, the default gateway and MAC remains the same. Switching at 1

No.	'111 Patent Claim 1	VMware NSX
		Layer-2 forwarding VMware NSX supports traditional layer-2 segments with proper flooding of BUM (Broadcast, Unknown unicast, Multicast) frames. NSX controller downloads forwarding entries to individual virtual switches, either through OpenFlow (NSX for multiple hypervisors) or a proprietary protocol (NSX for vSphere). The forwarding entries map destination VM MAC addresses into destination hypevisor (or gateway) IP addresses. On top of static forwarding entries downloaded from the controller, virtual switches perform dynamic MAC learning for MAC addresses reachable through layer-2 gateways. VMW_ORCT_CSCO_ 94 **flow and flow-based networking** RFC 3697 defines a network flow as "a sequence of packets sent from a particular source to a particular unicast, anycast, or multicast destination that the source desires to label as a flow." A flow is any unique stream of network traffic (as identified by its packet type, source and destination) that can be categorized based on inspection of a packet's fields. Flow-based networking creates a flow table that pairs each flow with an action that determines whether and how packets in that flow will be forwarded. VMW_ORCT_CSCO_ 140 *Static routes on logical routers* Network administrators can define static network routes on logical routers. Static routes are resolved as they would be in a traditional hardware router. NSX evaluates the routing rules in the routing table in the following order: The longest (i.e., most exact) prefix matches first. If two or more rules match, then the connected route for a given logical router port takes precedence over all other matched prefixes. You can manage static routes using NSX Manager or the NVP API, as explained below. VMW_ORCT_CSCO_ 2317

No.	'111 Patent Claim 1	VMware NSX
		Create an IP Address Group You can create an IP address group and then add this group as the source or destination in a firewall rule. Such a rule can help protect physical machines from virtual machines or vice versa. VMW_ORCT_CSCO_ 2320 **Working with Security Groups** A security group is a collection of assets or grouping objects from your vSphere inventory. VMW_ORCT_CSCO_ 4964-4965

No.	'111 Patent Claim 1	VMware NSX
		

No.	'111 Patent Claim 1	VMware NSX

Tenant-01 Access Rules

Objects	
ALL-CUST-VXLANS	
Tenant01-VXLAN	Tenant02-VXLAN
Tenan01-Services (192.168.10.0/24)	Tenant02-FIN-Apps (192.168.10.0/24)

Tenant-01 Section

Source	Destination	Services	Action	Apply To
Tenant01-VXLAN	Tenant01-Services	Any	Permit	Tenant01-VXLAN
...	...	...	...	Tenant01-VXLAN
Tenant01-VXLAN	Tenant01-VXLAN	Any	Deny	Tenant01-VXLAN

SP Tenant-01 Section

Source	Destination	Services	Action	Apply To
ALL-CUST-VXLANS	Tenant01-VXLAN	Any	Deny	
Tenant01-VXLAN	ALL-CUST-VXLANS	Any	Deny	

vmware

VMW_ORCT_CSCO_004964

No.	'111 Patent Claim 1	VMware NSX
1[b]	receiving, by the network node from the controller, the instruction and the criterion;	VMware NSX System discloses receiving, by the network node from the controller, the 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;	VMware NSX System discloses receiving, by the network node from the first entity over the packet network, a packet addressed to the second entity. For example, VMware NSX System discloses receiving packets at the vSwitch that are transmitting between ESXi hosts or endpoints. NSX Design Guide at 20

No.	'111 Patent Claim 1	VMware NSX
		Figure 12 describes the high level steps required to establish L2 communication between virtual machines connected to different ESXi hosts leveraging the VXLAN overlay functionalities. • VM1 originates a frame destined for the VM2 part of the same L2 logical segment/IP subnet. • The source ESXi host identifies the ESXi host (VTEP) where the VM2 is connected and encapsulates the frame before sending it into the transport network. • The transport network is only required to enable IP communication between the source and destination VTEPs. • The destination ESXi host receives the VXLAN frame, decapsulates it, and identifies the L2 segment it belongs to, leveraging the VNI value inserted in the VXLAN header by the source ESXi host. • The frame is delivered to VM2. NSX Design Guide at 22

3.1.5 NSX Edge Services Gateway

The NSX Edge is a multi-function, multi-use VM appliance for network virtualization.

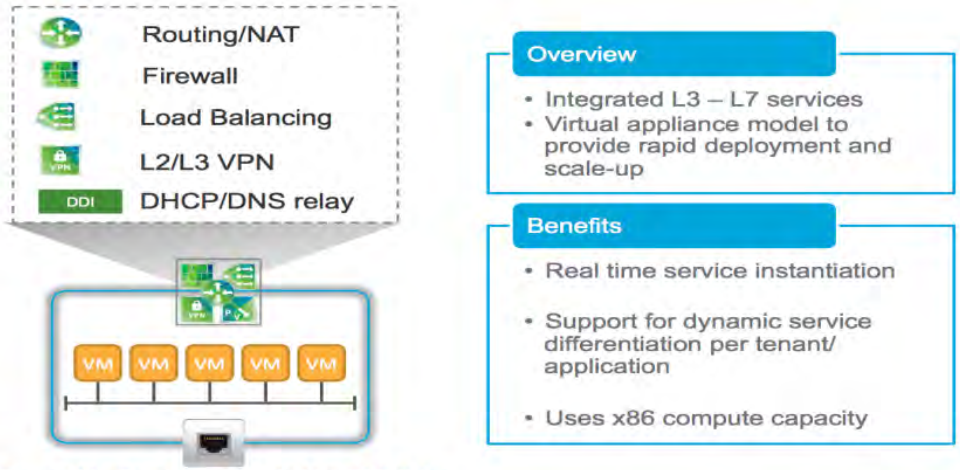


Figure 14 - Services Provided by NSX Edge

Figure 14 highlights the logical services provided by the NSX Edge. Its deployment varies based on its use, places in the topology, elastic performance requirements, and stateful services such as load balancer, firewall, VPN, and SSL. Edge VM supports two distinct modes of operation. The first one is active-standby in which all services are available. The second mode is ECMP mode, which provides high bandwidth (up to eight Edge VM supporting up to 80 GB traffic per DLR) and faster convergence. In ECMP mode, only routing service is available. Stateful services cannot be supported due to asymmetric routing inherent in ECMP-based forwarding. Each service is described briefly below, with additional detailed design choices discussed in "Edge Design and Deployment Considerations" section.

Routing: The NSX Edge provides centralized on-ramp/off-ramp routing between the logical networks deployed in the NSX domain and the external physical network infrastructure. The NSX Edge supports various dynamic routing protocols (e.g., OSPF, iBGP, eBGP) and can also leveraging static routing. The routing capability supports two models, active-standby stateful services and ECMP. The routing functionality is covered in greater detail in the "Logical Routing" section.

No.	'111 Patent Claim 1	VMware NSX
		NSX Design Guide at 45

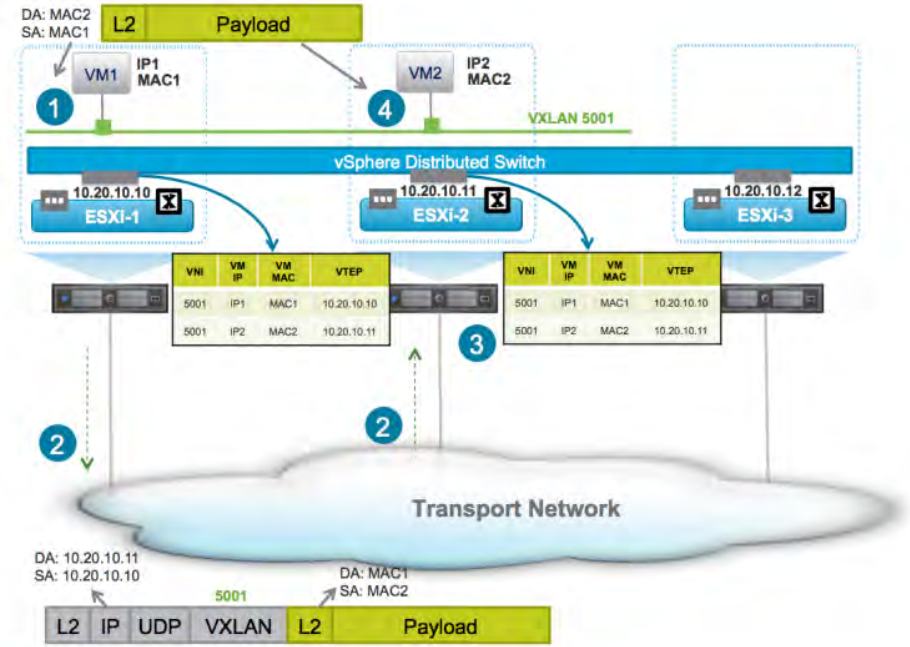


Figure 36 - Intra Logical Switch Unicast Communication

1. VM1 generates a data packet directed to VM2.
2. ESXi-1 learned about the location of VM2 from the ARP report received from the controller through control plane learning. It encapsulates the packet originated by VM1 in a VXLAN packet destined to the VTEP of ESXi2 – 10.20.10.11.
3. ESXi-2 receives the packet and leverages the information in the external IP header to learn about the location of VM1, associating VM1 MAC and IP addresses to the VTEP of ESXi-1.
4. The frame is delivered to VM2.

No.	'111 Patent Claim 1	VMware NSX
		NSX Design Guide at 52 Distributed routing is provided by a logical element called Distributed Logical Router (DLR). The DLR is essentially a router with directly connected interfaces to all hosts where VM connectivity is required. The supervisory function (i.e., control plane) to control the forwarding is imported from a control VM. The DLR consists of two primary components: • **Control Plane:** The control plane is provided by the DLR Control VM and NSX controller. The control plane supports dynamic routing protocols (e.g., BGP, OSPF), exchanges routing updates with the next layer 3 hop device (e.g., NSX Edge), and communicates with the NSX manager and the controller cluster. High availability for the DLR control VM is supported through an active/standby configuration. • **Data Plane:** DLR kernel modules (e.g., VIBs) are installed on the ESXi hosts part of the NSX domain. The kernel modules are similar to the line cards in a modular chassis supporting layer 3 routing. These kernel modules have a routing information base (RIB) that is pushed through the controller cluster. The traditional data plane functionality of route and ARP lookups is performed by the kernel modules. The kernel modules are equipped with logical interfaces (LIFs) connecting to the different logical switches. Each LIF has an IP address representing the default IP gateway for its logical L2 segment as well as a vMAC address. The IP address is unique per LIF and remains same where the logical switch exists. The vMAC associated with each LIF remains the consistent in each hypervisor as well and thus during the vMotion, the default gateway and MAC remains the same. NSX Design Guide at 54

No.	'111 Patent Claim 1	VMware NSX
		The required steps to establish routed communication between two virtual machines connected to separate logical segments are shown in Figure 46.

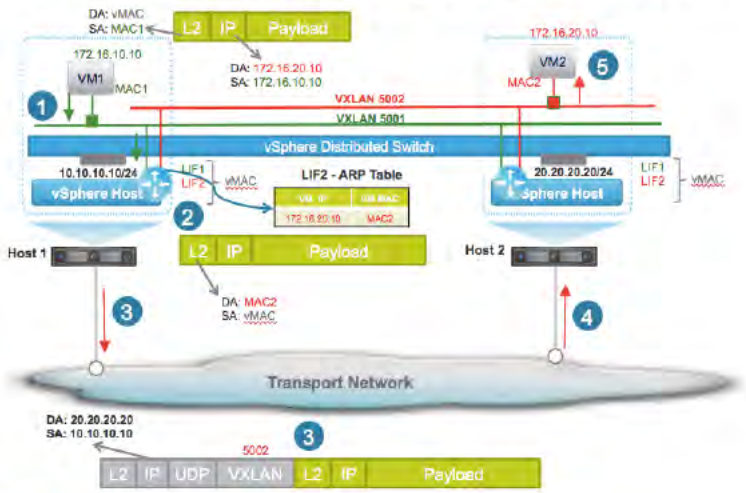


Figure 46 - Routed Communication between Virtual Machines

1. VM1 sends a packet to VM2 connected to a different VXLAN segment. The packet is sent to the VM1 default gateway interfaces located on the local DLR (172.16.10.1).
2. A routing lookup is performed on the local DLR, which determines that the destination subnet is directly connected to DLR LIF2. A lookup is performed in the LIF2 ARP table to determine the MAC address associated with the VM2 IP address. If the ARP information is not available, the DLR will generate an ARP request on VXLAN 5002 to determine the required mapping information.
3. An L2 lookup is performed in the local MAC table to determine how to reach VM2. The original packet is VXLAN encapsulated and sent to the VTEP of ESXi2 (20.20.20.20).
4. ESXi-2 decapsulates the packet and performs an L2 lookup in the local MAC table associated to VXLAN 5002 segment.
5. The packet is delivered to the destination VM2.

No.	'111 Patent Claim 1	VMware NSX
		VMW_ORCT_CSCO_.169 5. In the **Destination IP Address** and **Destination Port** fields, specify the values will be applied as the *destination address* and *port* in packets that match this rule. In the typical use case, this is a private IP address of a VM on your NSX network. That is, it's the private address of the VM that you want to receive all inbound traffic destined for a given known public IP address. VMW_ORCT_CSCO_2647 **Manage OSPF Routes for NSX Edge** NSX Edge supports OSPF, an interior gateway protocol that routes IP packets only within a single routing domain. It gathers link state information from available routers and constructs a topology map of the network. The topology determines the routing table presented to the Internet Layer, which makes routing decisions based on the destination IP address found in IP packets. OSPF routing policies provide a dynamic process of traffic load balancing between routes of equal cost. An OSPF network is divided into routing areas to optimize traffic. An area is a logical collection of OSPF networks, routers, and links that have the same area identification. Areas are identified by an Area ID.
1[d]	checking, by the network node, if the packet satisfies the criterion;	VMware NSX System discloses checking, by the network node, if the packet satisfies the criterion. For example, VMware NSX System discloses checking each packet against the rules in the rule table to determine if the specific packet header matches the rule. NSX Design Guide at 22

3.1.5 NSX Edge Services Gateway

The NSX Edge is a multi-function, multi-use VM appliance for network virtualization.

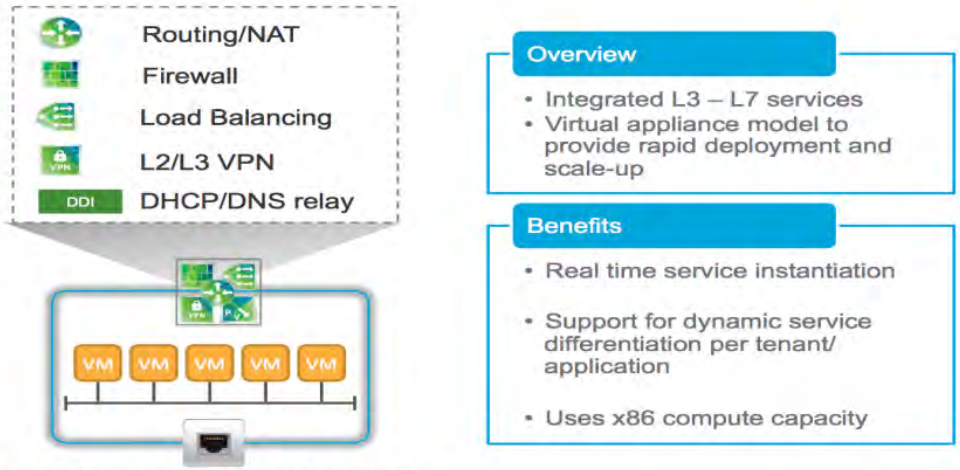


Figure 14 - Services Provided by NSX Edge

Figure 14 highlights the logical services provided by the NSX Edge. Its deployment varies based on its use, places in the topology, elastic performance requirements, and stateful services such as load balancer, firewall, VPN, and SSL. Edge VM supports two distinct modes of operation. The first one is active-standby in which all services are available. The second mode is ECMP mode, which provides high bandwidth (up to eight Edge VM supporting up to 80 GB traffic per DLR) and faster convergence. In ECMP mode, only routing service is available. Stateful services cannot be supported due to asymmetric routing inherent in ECMP-based forwarding. Each service is described briefly below, with additional detailed design choices discussed in "Edge Design and Deployment Considerations" section.

Routing: The NSX Edge provides centralized on-ramp/off-ramp routing between the logical networks deployed in the NSX domain and the external physical network infrastructure. The NSX Edge supports various dynamic routing protocols (e.g., OSPF, iBGP, eBGP) and can also leveraging static routing. The routing capability supports two models, active-standby stateful services and ECMP. The routing functionality is covered in greater detail in the "Logical Routing" section.

No.	'111 Patent Claim 1	VMware NSX
		NSX Design Guide at 29 Traffic redirection to third party vendor provides the capability to steer a particular type of traffic to a selected partner services VM. For instance, web traffic from the Internet to a set of Apache or Tomcat servers can be redirected to an L4-L7 deep packet inspection firewall for advanced protection. Traffic redirection is defined under Service Composer/Security Policy for NSX version 6.0 or under Partner Security Services tab of the DFW menu in NSX version 6.1. The Partner Security Services tab provides a powerful and easy to use user interface to define what type of traffic needs to be redirected to which partner services. It follows the same policy definition construct as DFW, providing the same options for source field, destination field and services field. The only difference is in the action field; instead of Block/Allow/Reject, a user can select between redirect/no redirect followed by a partner list. Any partner that has been registered with NSX can be successfully deployed on the platform. Additionally, log options can be enabled for this traffic redirection rule. The DFW instance on an ESXi host contains 2 separate tables. The rule table is used to store all policy rules, while the connection tracker table caches flow entries for rules with permit actions. One DFW instance is permitted per VM vNIC. A specific flow is identified by the 5-tuple information consisting source IP address, destination IP address, protocols, L4 source port, and L4 destination port fields. By default, DFW does not perform a lookup on L4 source port, but it can be configured to do so by defining a specific policy rule. NSX Design Guide at 30

DFW rules are enforced as follows:

- Rules are processed in top-to-bottom ordering.
- Each packet is checked against the top rule in the rule table before moving down the subsequent rules in the table.
- The first rule in the table that matches the traffic parameters is enforced. No subsequent rules can be enforced as the search is then terminated for that packet.

Because of this behavior, it is always recommended to put the most granular policies at the top of the rule table. This will ensure they will be enforced before more specific rules.

The DFW default policy rule, located at the bottom of the rule table, is a "catch-all" rule; packets not matching any other rules will be enforced by the default rule. After the host preparation operation, the DFW default rule is set to 'allow' action. This ensures that VM-to-VM communication is not broken during staging or migration phases. It is a best practice to then change this default rule to 'block' action and enforce access control through a positive control model (i.e., only traffic defined in the firewall policy is allowed onto the network).

Figure 22 steps through policy rule lookup and packet flow:

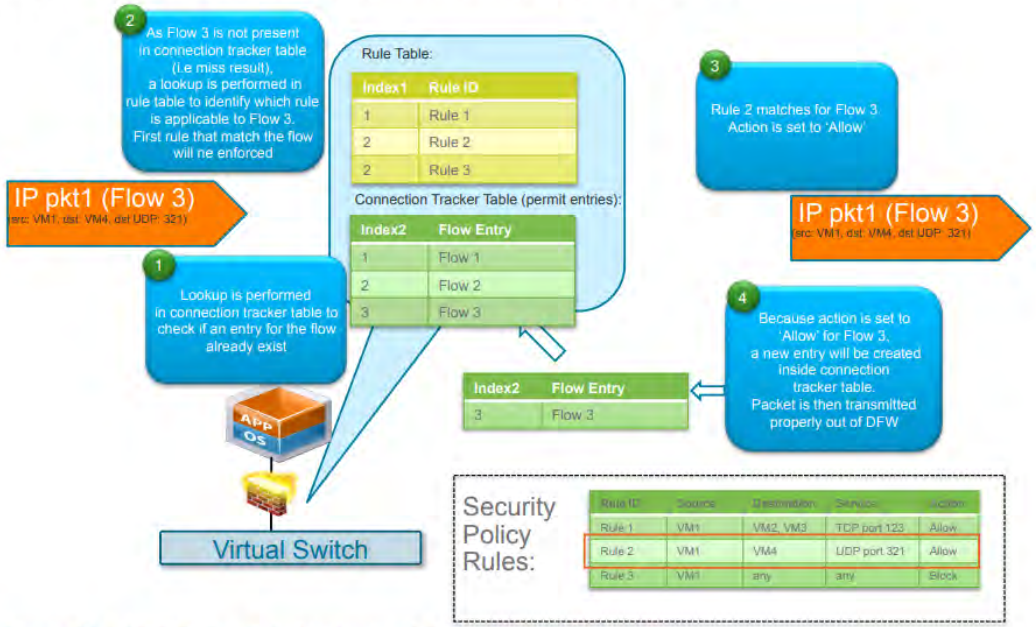
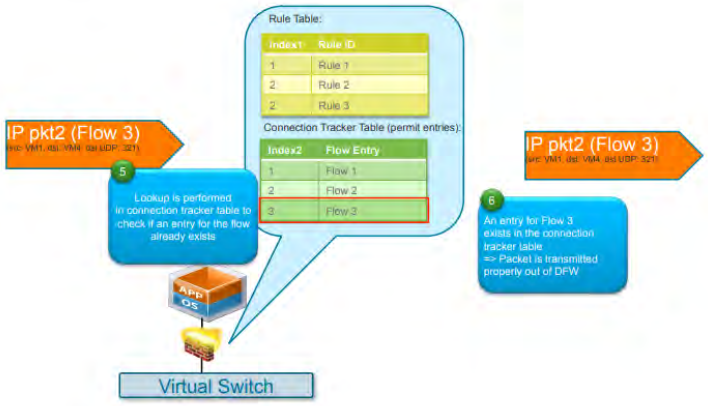
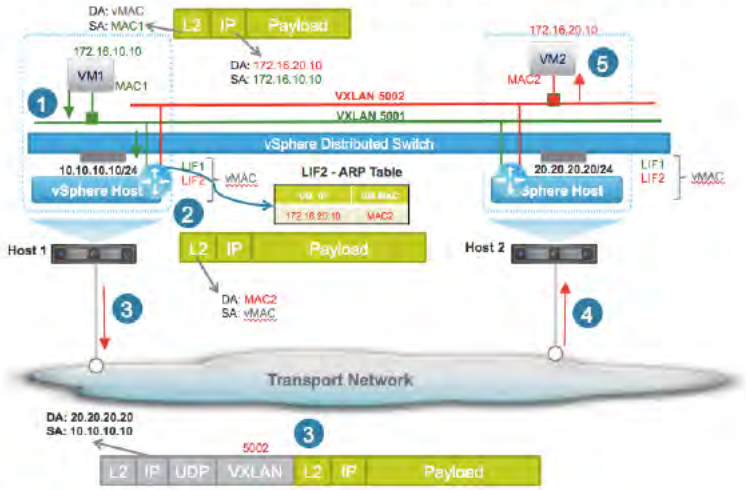


Figure 22 – DFW Policy Rule Lookup and Packet – First Packet

No.	'111 Patent Claim 1	VMware NSX
		NSX Design Guide at 30-31

No.	'111 Patent Claim 1	VMware NSX
		An IP packet identified as pkt1 that matches rule number 2. The order of operation is the following: 30 1. A lookup is performed in the connection tracker table to determine if an entry for the flow already exists. 2. As flow 3 is not present in the connection tracker table, a lookup is performed in the rule table to identify which rule is applicable to flow 3. The first rule that matches the flow will be enforced. 3. Rule 2 matches for flow 3. The action is set to 'Allow'. 4. Because the action is set to 'Allow' for flow 3, a new entry will be created inside the connection tracker table. The packet is then transmitted out of DFW.  Figure 23 – DFW Policy Rule Lookup and Packet – Subsequent Packets.

No.	'111 Patent Claim 1	VMware NSX
		NSX Design Guide at 54 The required steps to establish routed communication between two virtual machines connected to separate logical segments are shown in Figure 46.  Figure 46 - Routed Communication between Virtual Machines 1. VM1 sends a packet to VM2 connected to a different VXLAN segment. The packet is sent to the VM1 default gateway interfaces located on the local DLR (172.16.10.1). 2. A routing lookup is performed on the local DLR, which determines that the destination subnet is directly connected to DLR LIF2. A lookup is performed in the LIF2 ARP table to determine the MAC address associated with the VM2 IP address. If the ARP information is not available, the DLR will generate an ARP request on VXLAN 5002 to determine the required mapping information. 3. An L2 lookup is performed in the local MAC table to determine how to reach VM2. The original packet is VXLAN encapsulated and sent to the VTEP of ESXi2 (20.20.20.20). 4. ESXi-2 decapsulates the packet and performs an L2 lookup in the local MAC table associated to VXLAN 5002 segment. 5. The packet is delivered to the destination VM2.

No.	'111 Patent Claim 1	VMware NSX
		Switching at 1 Layer-2 forwarding VMware NSX supports traditional layer-2 segments with proper flooding of BUM (Broadcast, Unknown unicast, Multicast) frames. NSX controller downloads forwarding entries to individual virtual switches, either through OpenFlow (NSX for multiple hypervisors) or a proprietary protocol (NSX for vSphere). The forwarding entries map destination VM MAC addresses into destination hypevisor (or gateway) IP addresses. On top of static forwarding entries downloaded from the controller, virtual switches perform dynamic MAC learning for MAC addresses reachable through layer-2 gateways. VMW_ORCT_CSCO_ 85

Logical Layer Packet Filtering

NSX provides several capabilities in the logical pipeline that perform packet filtering. Port Security and Port Isolation provide simple filtering models, while Security Profiles provide a richer L3/L4 filtering model. This section details how this filtering is performed, in particular, the details of how NSX automatically matches and permits "reply" traffic for advanced filtering mechanisms like Security Profiles.

For more details on the use cases and configuration details of these features, please see Logical Network Features: Security (page 120).

Transport Layer Implementation

For all logical layer security features, ingress packet filtering is performed by the OVS instance on the hypervisor running the transmitting VM and egress filtering is implemented at the hypervisor running the VM about to receive the traffic.

In the case of Security Profiles, assigning a profile to a logical port requires that the logical port is using the NSX Port Security feature. This is so that the NSX Controller is aware of all valid IP addresses that can be used by a VM, in order to implement filter rules where the remote hosts are identified using a Security Profile UUID (e.g., a rule like "allow tcp 3306 from Webservers").

Interactions between Security Filtering Features

In general, Port Isolation, Port Security, and Security Profiles should be thought of as different stages in the logical pipeline, meaning that traffic must pass all filters in order to be permitted. Thus, even if a connection is allowed by the Security Profile, it will be dropped if it does not match the allowed address pairs for a logical port. For example, even if a Security Profile was configured to allow IP multicast to reach a VM, the multicast traffic would be dropped by Port Security because the destination MAC address did not match an allowed address pair.

One exception to this rule is that Port Security automatically allows non-spoofed ARP (IPv4) and ICMP Neighbor Discovery (IPv6) packets to allow VMs to discover the MAC address associated with an IP address. No Security Profile rules are required to permit such traffic, and it is not possible for Security Profiles to drop such traffic.

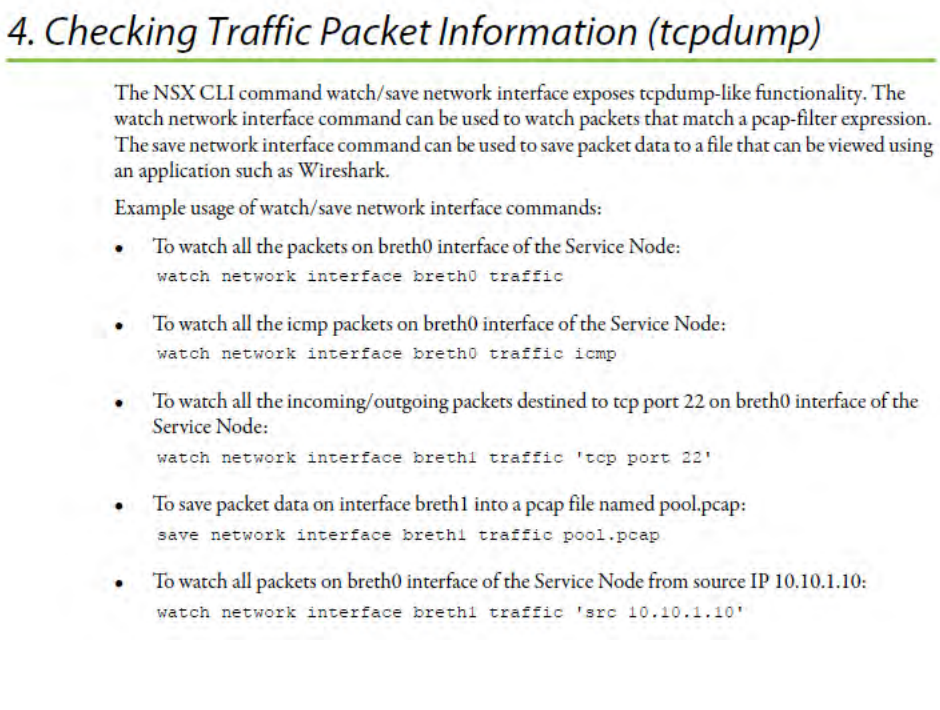
Automatically Allows "Reply" Traffic for Existing Connections

With Security Profiles, replies to already permitted traffic are automatically allowed. More precisely:

- Egress traffic that matches an existing connection already allowed by ingress filtering bypasses egress filtering and is automatically allowed.
- Ingress traffic that matches an existing connection allowed by egress filtering bypasses ingress filtering and is automatically allowed.

For egress reply traffic (i.e., replies to an outbound connection initiated by the VM), this is implemented using "connection tracking", which dynamically sets up a flow entry to match return traffic by reversing the TCP or UDP 5-tuple of (IP protocol, source IP address, source TCP/UDP port, destination IP address, destination TCP/UDP port). Egress ICMP ping reply, ICMP TTL-exceeded, and ICMP destination unreachable messages are always allowed on logical port egress. There is no support for "application layer gateway" (ALG) functionality that parses higher level protocols like FTP to open related ports.

No.	'111 Patent Claim 1	VMware NSX
		VMW_ORCT_CSCO_. 140 *Static routes on logical routers* Network administrators can define static network routes on logical routers. Static routes are resolved as they would be in a traditional hardware router. NSX evaluates the routing rules in the routing table in the following order: The longest (i.e., most exact) prefix matches first. If two or more rules match, then the connected route for a given logical router port takes precedence over all other matched prefixes. You can manage static routes using NSX Manager or the NVP API, as explained below. VMW_ORCT_CSCO_. 169 5. In the **Destination IP Address** and **Destination Port** fields, specify the values will be applied as the *destination address* and *port* in packets that match this rule. In the typical use case, this is a private IP address of a VM on your NSX network. That is, it's the private address of the VM that you want to receive all inbound traffic destined for a given known public IP address. VMW_ORCT_CSCO_. 379

No.	'111 Patent Claim 1	VMware NSX
		

4. Checking Traffic Packet Information (tcpdump)

The NSX CLI command watch/save network interface exposes tcpdump-like functionality. The watch network interface command can be used to watch packets that match a pcap-filter expression. The save network interface command can be used to save packet data to a file that can be viewed using an application such as Wireshark.

Example usage of watch/save network interface commands:

- To watch all the packets on breth0 interface of the Service Node:

```
watch network interface breth0 traffic
```

- To watch all the icmp packets on breth0 interface of the Service Node:

```
watch network interface breth0 traffic icmp
```

- To watch all the incoming/outgoing packets destined to tcp port 22 on breth0 interface of the Service Node:

```
watch network interface breth1 traffic 'tcp port 22'
```

- To save packet data on interface breth1 into a pcap file named pool.pcap:

```
save network interface breth1 traffic pool.pcap
```

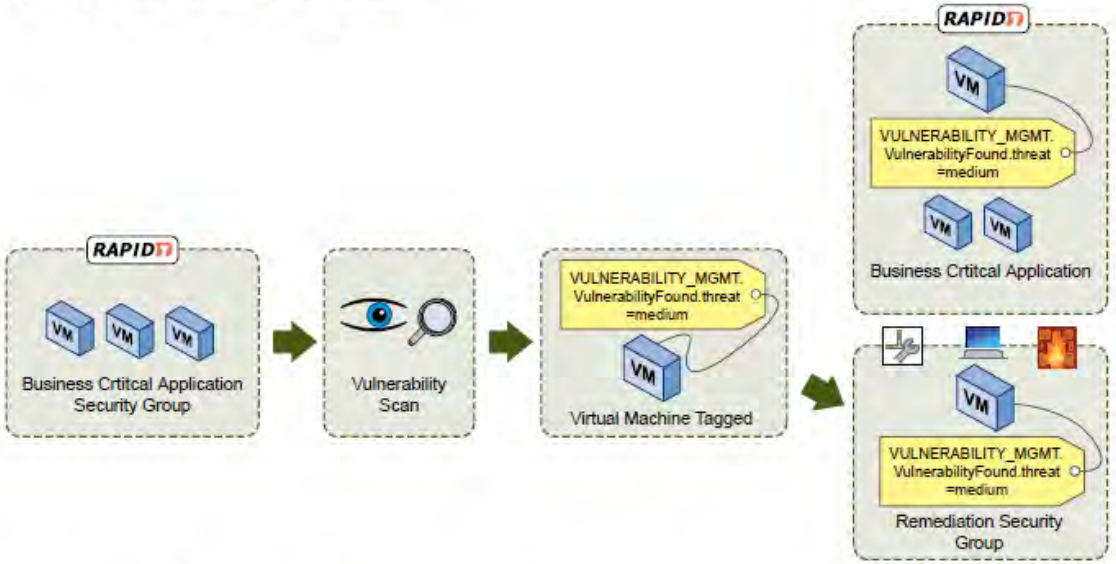
- To watch all packets on breth0 interface of the Service Node from source IP 10.10.1.10:

```
watch network interface breth1 traffic 'src 10.10.1.10'
```

VMW_ORCT_CSCO_1461

No.	'111 Patent Claim 1	VMware NSX

Figure 11-2. Service Composer in action



This topic will now take you through the steps required to consume the security services offered by Service Composer.

1 Create a Security Group in Service Composer on page 125

You create a security group at the NSX Manager level.

2 Create a Security Policy on page 127

A security policy is a set of Endpoint, firewall, and network introspection services that can be applied to a security group. The order in which security policies are displayed is determined by the weight associated with the policy. By default, a new policy is assigned the highest weight so that it is at the top of the table. However, you can modify the default suggested weight to change the order assigned to the new policy.

3 Map a Security Policy to a Security Group on page 129

You can apply a security policy to a security group to secure your virtual desktops, business critical applications, and the connections between them. You can also view a list of the services that were not applied and the reason they failed to apply.

No.	'111 Patent Claim 1	VMware NSX
		VMW_ORCT_CSCO_2610 **Manage BGP Routes for NSX Edge** Border Gateway Protocol (BGP) makes core routing decisions. It includes a table of IP networks or prefixes which designate network reachability among autonomous systems. An underlying connection between two BGP speakers is established before any routing information is exchanged. Keep alive messages are sent out by the BGP speakers in order to keep this relationship alive. Once the connection is established, the BGP speakers exchange routes and synchronize their tables. VMW_ORCT_CSCO_3404

No.	'111 Patent Claim 1	VMware NSX
		Edge Firewall Edge Firewall provides perimeter security functionality including firewall, Network Address Translation (NAT) as well as Site to site IPSec and SSL VPN functionality. This solution is available in the virtual machine form factor and can be deployed in a High Availability mode. **Working with Firewall Rules** Distributed Firewall rules and Edge Firewall rules can be managed in a centralized manner on the Firewall tab. In a multi-tenant environment, providers can define high-level traffic flow rules on the centralized Firewall user interface. Rules defined on the centralized level are referred to as pre rules. Tenants can then add rules at an individual NSX Edge level, which are referred to as local rules. Each traffic session is checked against the top rule in the Firewall table before moving down the subsequent rules in the table. The first rule in the table that matches the traffic parameters is enforced. Rules are displayed (and enforced) in the following order: 1 User defined pre rules have the highest priority, and are enforced in top-to-bottom ordering with a per-virtual NIC level precedence. 2 Auto plumbed rules. 3 Local rules defined at an NSX Edge level. 4 Service Composer rules - a separate section for each policy. You cannot edit these rules in the Firewall table, but you can add rules at the top of a security policy firewall rules section. If you do so, you must re-synchronize the rules in Service Composer. For more information, see Chapter 11, "Service Composer," on page 121. 5 Default Distributed Firewall rule Note that firewall rules are enforced only on clusters on which you have enabled firewall. For information on preparing clusters, see the *NSX Installation and Upgrade Guide*. VMW_ORCT_CSCO_ 4033

Distributed Firewall is a hypervisor kernel-embedded firewall that provides visibility and control for virtualized workloads and networks. You can create access control policies based on VMware vCenter objects like datacenters and clusters, virtual machine names and tags, network constructs like IP/VLAN/VXLAN addresses, as well as user group identity from Active Directory. Firewall rules are enforced at the vNIC level of each virtual machine to provide consistent access control even when the virtual machine gets vMotioned. The hypervisor-embedded nature of the firewall delivers close to line rate throughput to enable higher workload consolidation on physical servers. The distributed nature of the firewall provides a scale-out architecture that automatically extends firewall capacity when additional hosts are added to a prepared cluster.

Distributed Firewall offers multiple sets of configurable rules: Layer 3 (L3) rules (General tab) and Layer 2 (L2) rules (Ethernet tab). Layer 2 firewall rules are processed before Layer 3 rules. The default firewall rule allows all L3 and L2 traffic to pass through all clusters in your infrastructure. The default rule is always at the bottom of the rules table and cannot be deleted or added to. However, you can change the Action element of the rule from Allow to Block, add comments for the rule, and indicate whether traffic for that rule should be logged.

Firewall rules are created at the global scope, but you can then narrow the scope (datacenter, cluster, distributed virtual port group, network, virtual machine, vNIC, or virtual wire) at which you want to apply the rule by using the AppliedTo keyword.

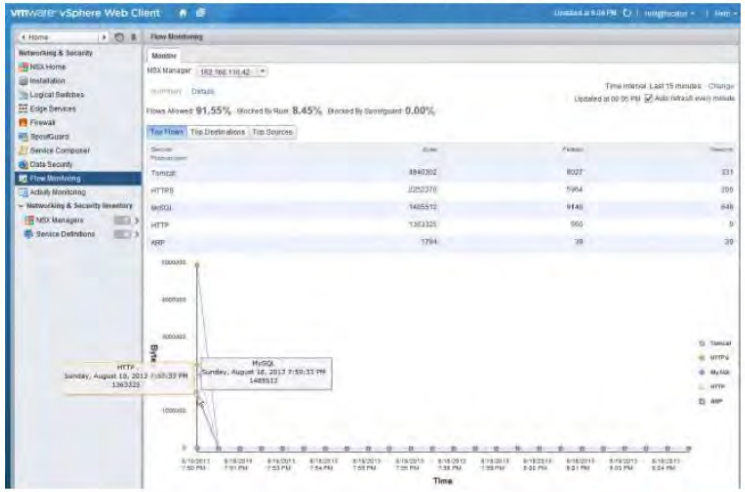
User defined firewall rules are enforced in top-to-bottom ordering, with a per-virtual NIC level precedence. Each traffic session is checked against the top rule in the Firewall table before moving down the subsequent rules in the table. The first rule in the table that matches the traffic parameters is enforced.

The followng table lists the elements that can be used in firewall rules.

Table 9-1. Firewall rule elements

Element	Keyword for API	Used In
application	Application	service
application group	ApplicationGroup	service
cluster compute resource	ClusterComputeResource	AppliedTo
datacenter	Datacenter	source/destination AppliedTo
distributed virtual port group	DistributedVirtualPortgroup	source/destination AppliedTo
global root	GlobalRoot	source/destination
IPv4 addresses	Ipv4Address	source/destination
IPv6 addresses	Ipv6Address	source/destination
logical switch	VirtualWire	source/destination AppliedTo
network	Network	AppliedTo

No.	'111 Patent Claim 1	VMware NSX
		VMW_ORCT_CSCO_. 2350 **Flow Monitoring** Flow Monitoring is a traffic analysis tool that provides a detailed view of the traffic to and from protected virtual machines. When flow monitoring is enabled, its output defines which machines are exchanging data and over which application. This data includes the number of sessions and packets transmitted per session. Session details include sources, destinations, applications, and ports being used. Session details can be used to create firewall allow or block rules. You can view TCP and UDP connections to and from a selected vNIC. You can also exclude flows by specifying filters. Flow Monitoring can thus be used as a forensic tool to detect rogue services and examine outbound sessions. **Configure Flow Monitoring Data Collection** After you have viewed and filtered the flow monitoring data that you want to collect, you can configure data collection. You can filter the data being displayed by specifying exclusion criterion. For example, you may want to exclude a proxy server to avoid seeing duplicate flows. Or if you are running a Nessus scan on the virtual machines in your inventory, you may not want to exclude the scan flows from being collected. You can configure IPFix so that information for specific flows are exported directly from a firewall to a flow collector. The flow monitoring graphs do not include the IPFix flows. These are displayed on the IPFix collector's interface. **Procedure** 1 Log in to the vSphere Web Client. 2 Select **Networking & Security** from the left navigation pane and then select **Flow Monitoring**. 3 Select the **Configuration** tab. 4 Ensure that **Global Flow Collection Status** is **Enabled**. All firewall related flows are collected across your inventory except for the objects specified in **Exclusion Settings**. VMW_ORCT_CSCO_ 4519

No.	'111 Patent Claim 1	VMware NSX
		Operational Tools – Flow Monitoring ▪ Flow monitoring provides vNIC level visibility of VM traffic flows ▪ Reporting on Top Flows, Destinations and Sources ▪ Detailed Flow Data for both Allowed and Blocked Flows  37 vmware VMW_ORCT_CSCO_004519
1[e]	responsive to the packet not satisfying the criterion, sending, by the network node over	VMware NSX 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. For example, VMware NSX System discloses forwarding the packet to the destination endpoint based on a default catch all rule that is applied when the packet does not match any of the other rules. NSX Design Guide at 11

No.	'111 Patent Claim 1	VMware NSX
	the packet network, the packet to the second entity; and	**2.1.2 Control Plane** The NSX controller is a key part of the NSX control plane. In a vSphere environment with the vSphere Distributed Switch (VDS), the controller enables multicast free VXLAN and control plane programming of elements such as the Distributed Logical Routing (DLR). Stability and reliability of data transport are central concerns in networking. The NSX controller is a part of the control plane; it is logically separated from all data plane traffic. To further enhance high availability and scalability, NSX controller nodes are deployed in a cluster of odd number instances. In addition to controller, the control VM, provides the routing control plane that allows the local forwarding in ESXi and allows dynamic routing between ESXI and north-south routing provided by Edge VM. It is critical to understand that data plane traffic never traverses the control plane component. NSX Design Guide at 22

3.1.5 NSX Edge Services Gateway

The NSX Edge is a multi-function, multi-use VM appliance for network virtualization.

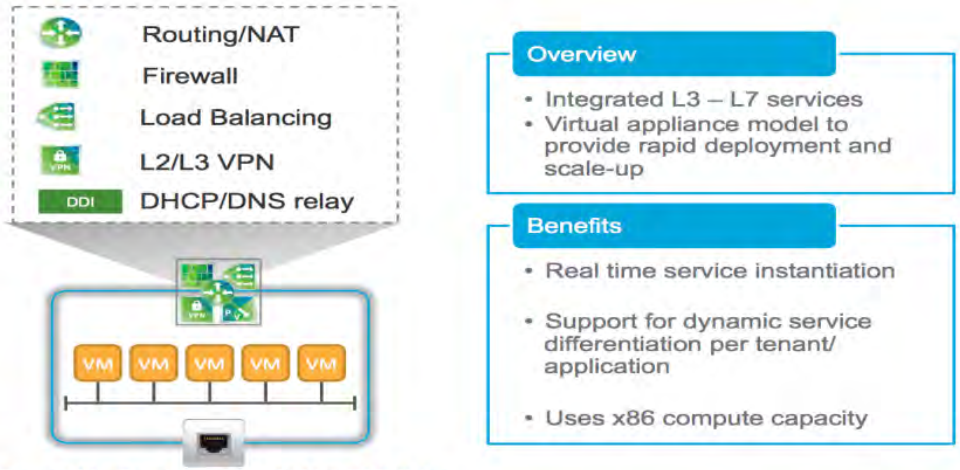
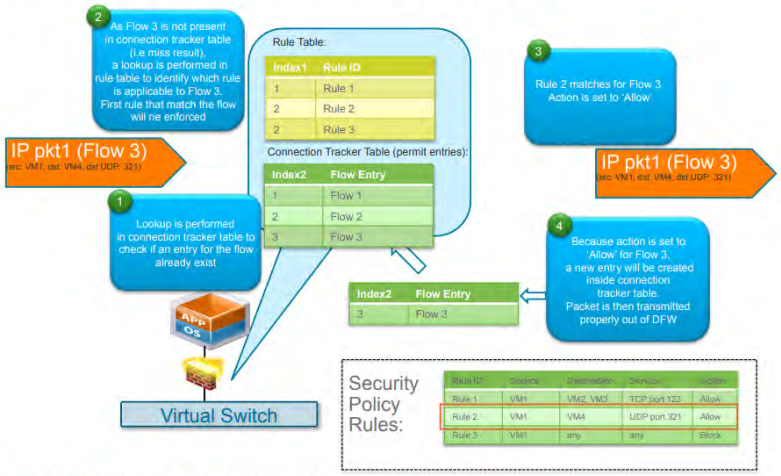


Figure 14 - Services Provided by NSX Edge

Figure 14 highlights the logical services provided by the NSX Edge. Its deployment varies based on its use, places in the topology, elastic performance requirements, and stateful services such as load balancer, firewall, VPN, and SSL. Edge VM supports two distinct modes of operation. The first one is active-standby in which all services are available. The second mode is ECMP mode, which provides high bandwidth (up to eight Edge VM supporting up to 80 GB traffic per DLR) and faster convergence. In ECMP mode, only routing service is available. Stateful services cannot be supported due to asymmetric routing inherent in ECMP-based forwarding. Each service is described briefly below, with additional detailed design choices discussed in "Edge Design and Deployment Considerations" section.

Routing: The NSX Edge provides centralized on-ramp/off-ramp routing between the logical networks deployed in the NSX domain and the external physical network infrastructure. The NSX Edge supports various dynamic routing protocols (e.g., OSPF, iBGP, eBGP) and can also leveraging static routing. The routing capability supports two models, active-standby stateful services and ECMP. The routing functionality is covered in greater detail in the "Logical Routing" section.

No.	'111 Patent Claim 1	VMware NSX
		NSX Design Guide at 30 DFW rules are enforced as follows: • Rules are processed in top-to-bottom ordering. • Each packet is checked against the top rule in the rule table before moving down the subsequent rules in the table. • The first rule in the table that matches the traffic parameters is enforced. No subsequent rules can be enforced as the search is then terminated for that packet. Because of this behavior, it is always recommended to put the most granular policies at the top of the rule table. This will ensure they will be enforced before more specific rules. The DFW default policy rule, located at the bottom of the rule table, is a "catch-all" rule; packets not matching any other rules will be enforced by the default rule. After the host preparation operation, the DFW default rule is set to 'allow' action. This ensures that VM-to-VM communication is not broken during staging or migration phases. It is a best practice to then change this default rule to 'block' action and enforce access control through a positive control model (i.e., only traffic defined in the firewall policy is allowed onto the network). Figure 22 steps through policy rule lookup and packet flow:  Figure 22 – DFW Policy Rule Lookup and Packet – First Packet

No.	'111 Patent Claim 1	VMware NSX
		NSX Design Guide at 30-31

An IP packet identified as pkt1 that matches rule number 2. The order of operation is the following:

30

1. A lookup is performed in the connection tracker table to determine if an entry for the flow already exists.
2. As flow 3 is not present in the connection tracker table, a lookup is performed in the rule table to identify which rule is applicable to flow 3. The first rule that matches the flow will be enforced.
3. Rule 2 matches for flow 3. The action is set to 'Allow'.
4. Because the action is set to 'Allow' for flow 3, a new entry will be created inside the connection tracker table. The packet is then transmitted out of DFW.

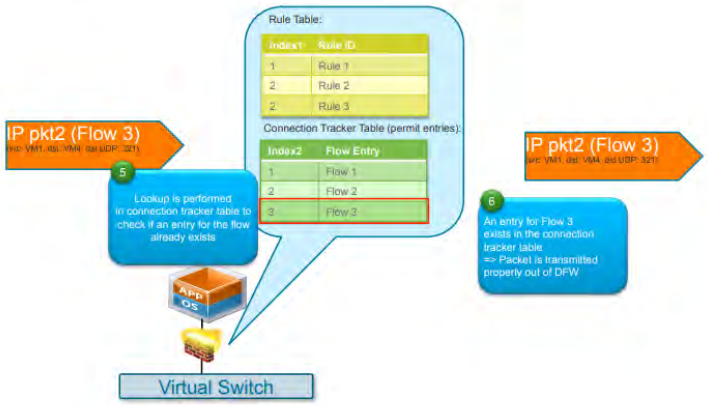


Figure 23 – DFW Policy Rule Lookup and Packet – Subsequent Packets.

No.	'111 Patent Claim 1	VMware NSX
		NSX Design Guide at 54

No.	'111 Patent Claim 1	VMware NSX
		The required steps to establish routed communication between two virtual machines connected to separate logical segments are shown in Figure 46.

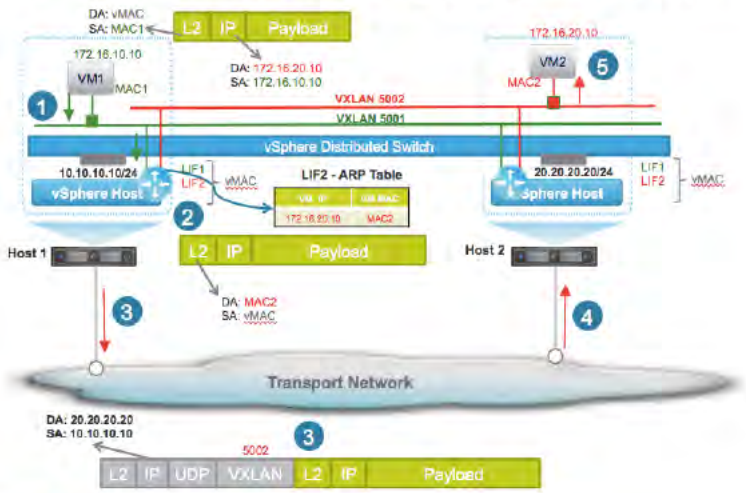


Figure 46 - Routed Communication between Virtual Machines

1. VM1 sends a packet to VM2 connected to a different VXLAN segment. The packet is sent to the VM1 default gateway interfaces located on the local DLR (172.16.10.1).
2. A routing lookup is performed on the local DLR, which determines that the destination subnet is directly connected to DLR LIF2. A lookup is performed in the LIF2 ARP table to determine the MAC address associated with the VM2 IP address. If the ARP information is not available, the DLR will generate an ARP request on VXLAN 5002 to determine the required mapping information.
3. An L2 lookup is performed in the local MAC table to determine how to reach VM2. The original packet is VXLAN encapsulated and sent to the VTEP of ESXi2 (20.20.20.20).
4. ESXi-2 decapsulates the packet and performs an L2 lookup in the local MAC table associated to VXLAN 5002 segment.
5. The packet is delivered to the destination VM2.

No.	'111 Patent Claim 1	VMware NSX
		VMW_ORCT_CSCO_. 85

Logical Layer Packet Filtering

NSX provides several capabilities in the logical pipeline that perform packet filtering. Port Security and Port Isolation provide simple filtering models, while Security Profiles provide a richer L3/L4 filtering model. This section details how this filtering is performed, in particular, the details of how NSX automatically matches and permits "reply" traffic for advanced filtering mechanisms like Security Profiles.

For more details on the use cases and configuration details of these features, please see Logical Network Features: Security (page 120).

Transport Layer Implementation

For all logical layer security features, ingress packet filtering is performed by the OVS instance on the hypervisor running the transmitting VM and egress filtering is implemented at the hypervisor running the VM about to receive the traffic.

In the case of Security Profiles, assigning a profile to a logical port requires that the logical port is using the NSX Port Security feature. This is so that the NSX Controller is aware of all valid IP addresses that can be used by a VM, in order to implement filter rules where the remote hosts are identified using a Security Profile UUID (e.g., a rule like "allow tcp 3306 from Webservers").

Interactions between Security Filtering Features

In general, Port Isolation, Port Security, and Security Profiles should be thought of as different stages in the logical pipeline, meaning that traffic must pass all filters in order to be permitted. Thus, even if a connection is allowed by the Security Profile, it will be dropped if it does not match the allowed address pairs for a logical port. For example, even if a Security Profile was configured to allow IP multicast to reach a VM, the multicast traffic would be dropped by Port Security because the destination MAC address did not match an allowed address pair.

One exception to this rule is that Port Security automatically allows non-spoofed ARP (IPv4) and ICMP Neighbor Discovery (IPv6) packets to allow VMs to discover the MAC address associated with an IP address. No Security Profile rules are required to permit such traffic, and it is not possible for Security Profiles to drop such traffic.

Automatically Allows "Reply" Traffic for Existing Connections

With Security Profiles, replies to already permitted traffic are automatically allowed. More precisely:

- Egress traffic that matches an existing connection already allowed by ingress filtering bypasses egress filtering and is automatically allowed.
- Ingress traffic that matches an existing connection allowed by egress filtering bypasses ingress filtering and is automatically allowed.

For egress reply traffic (i.e., replies to an outbound connection initiated by the VM), this is implemented using "connection tracking", which dynamically sets up a flow entry to match return traffic by reversing the TCP or UDP 5-tuple of (IP protocol, source IP address, source TCP/UDP port, destination IP address, destination TCP/UDP port). Egress ICMP ping reply, ICMP TTL-exceeded, and ICMP destination unreachable messages are always allowed on logical port egress. There is no support for "application layer gateway" (ALG) functionality that parses higher level protocols like FTP to open related ports.

No.	'111 Patent Claim 1	VMware NSX
		VMW_ORCT_CSCO_4033

Distributed Firewall is a hypervisor kernel-embedded firewall that provides visibility and control for virtualized workloads and networks. You can create access control policies based on VMware vCenter objects like datacenters and clusters, virtual machine names and tags, network constructs like IP/VLAN/VXLAN addresses, as well as user group identity from Active Directory. Firewall rules are enforced at the vNIC level of each virtual machine to provide consistent access control even when the virtual machine gets vMotioned. The hypervisor-embedded nature of the firewall delivers close to line rate throughput to enable higher workload consolidation on physical servers. The distributed nature of the firewall provides a scale-out architecture that automatically extends firewall capacity when additional hosts are added to a prepared cluster.

Distributed Firewall offers multiple sets of configurable rules: Layer 3 (L3) rules (General tab) and Layer 2 (L2) rules (Ethernet tab). Layer 2 firewall rules are processed before Layer 3 rules. The default firewall rule allows all L3 and L2 traffic to pass through all clusters in your infrastructure. The default rule is always at the bottom of the rules table and cannot be deleted or added to. However, you can change the Action element of the rule from Allow to Block, add comments for the rule, and indicate whether traffic for that rule should be logged.

Firewall rules are created at the global scope, but you can then narrow the scope (datacenter, cluster, distributed virtual port group, network, virtual machine, vNIC, or virtual wire) at which you want to apply the rule by using the AppliedTo keyword.

User defined firewall rules are enforced in top-to-bottom ordering, with a per-virtual NIC level precedence. Each traffic session is checked against the top rule in the Firewall table before moving down the subsequent rules in the table. The first rule in the table that matches the traffic parameters is enforced.

The followng table lists the elements that can be used in firewall rules.

Table 9-1. Firewall rule elements

Element	Keyword for API	Used In
application	Application	service
application group	ApplicationGroup	service
cluster compute resource	ClusterComputeResource	AppliedTo
datacenter	Datacenter	source/destination AppliedTo
distributed virtual port group	DistributedVirtualPortgroup	source/destination AppliedTo
global root	GlobalRoot	source/destination
IPv4 addresses	Ipv4Address	source/destination
IPv6 addresses	Ipv6Address	source/destination
logical switch	VirtualWire	source/destination AppliedTo
network	Network	AppliedTo

No.	'111 Patent Claim 1	VMware NSX
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.	VMware NSX 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. For example, VMware NSX System discloses sending the packet over the virtualized network to a service device or other entity in response to the packet matching the rule. NSX Design Guide at 9-10

ii. NSX Services Platform

NSX is a services platform that enables both security services and distributed firewall.

9

These services are either a built-in part of NSX or available from 3rd party vendors. Existing physical devices – examples include physical load balancers, firewalls, syslog collectors, and monitoring devices – can also be integrated with NSX. All services can be orchestrated via the NSX consumption model (explained in section 2.1.3 below).

2.1.1.1 Networking and Edge Services

NSX provides built-in networking and edge services including logical load balancing, L2/L3 VPN services, edge firewalling, and DHCP/NAT.

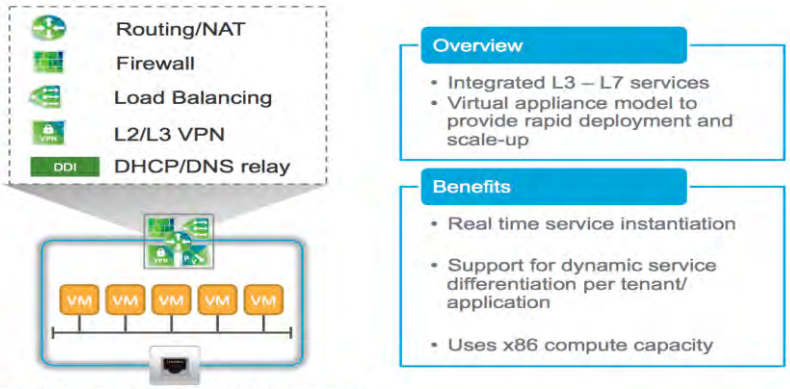


Figure 6 - NSX Logical Network Services

Edge Firewall: Edge firewall services are part of the NSX Edge Services Gateway (ESG). The Edge firewall provides essential perimeter firewall protection which can be used in addition to a physical perimeter firewall. The ESG-based firewall is useful in developing PCI zones, multi-tenant environments, or dev-ops style connectivity without forcing the inter-tenant or inter-zone traffic onto the physical network.

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1822 of 2137

No.	'111 Patent Claim 1	VMware NSX
		NSX Design Guide at 22

No.	'111 Patent Claim 1	VMware NSX
		3.1.5 NSX Edge Services Gateway The NSX Edge is a multi-function, multi-use VM appliance for network virtualization.  Figure 14 - Services Provided by NSX Edge Figure 14 highlights the logical services provided by the NSX Edge. Its deployment varies based on its use, places in the topology, elastic performance requirements, and stateful services such as load balancer, firewall, VPN, and SSL. Edge VM supports two distinct modes of operation. The first one is active-standby in which all services are available. The second mode is ECMP mode, which provides high bandwidth (up to eight Edge VM supporting up to 80 GB traffic per DLR) and faster convergence. In ECMP mode, only routing service is available. Stateful services cannot be supported due to asymmetric routing inherent in ECMP-based forwarding. Each service is described briefly below, with additional detailed design choices discussed in "Edge Design and Deployment Considerations" section. **Routing:** The NSX Edge provides centralized on-ramp/off-ramp routing between the logical networks deployed in the NSX domain and the external physical network infrastructure. The NSX Edge supports various dynamic routing protocols (e.g., OSPF, iBGP, eBGP) and can also leveraging static routing. The routing capability supports two models, active-standby stateful services and ECMP. The routing functionality is covered in greater detail in the "Logical Routing" section.

No.	'111 Patent Claim 1	VMware NSX
		NSX Design Guide at 29 Traffic redirection to third party vendor provides the capability to steer a particular type of traffic to a selected partner services VM. For instance, web traffic from the Internet to a set of Apache or Tomcat servers can be redirected to an L4-L7 deep packet inspection firewall for advanced protection. Traffic redirection is defined under Service Composer/Security Policy for NSX version 6.0 or under Partner Security Services tab of the DFW menu in NSX version 6.1. The Partner Security Services tab provides a powerful and easy to use user interface to define what type of traffic needs to be redirected to which partner services. It follows the same policy definition construct as DFW, providing the same options for source field, destination field and services field. The only difference is in the action field; instead of Block/Allow/Reject, a user can select between redirect/no redirect followed by a partner list. Any partner that has been registered with NSX can be successfully deployed on the platform. Additionally, log options can be enabled for this traffic redirection rule. The DFW instance on an ESXi host contains 2 separate tables. The rule table is used to store all policy rules, while the connection tracker table caches flow entries for rules with permit actions. One DFW instance is permitted per VM vNIC. A specific flow is identified by the 5-tuple information consisting source IP address, destination IP address, protocols, L4 source port, and L4 destination port fields. By default, DFW does not perform a lookup on L4 source port, but it can be configured to do so by defining a specific policy rule. NSX Design Guide at 30

DFW rules are enforced as follows:

- Rules are processed in top-to-bottom ordering.
- Each packet is checked against the top rule in the rule table before moving down the subsequent rules in the table.
- The first rule in the table that matches the traffic parameters is enforced. No subsequent rules can be enforced as the search is then terminated for that packet.

Because of this behavior, it is always recommended to put the most granular policies at the top of the rule table. This will ensure they will be enforced before more specific rules.

The DFW default policy rule, located at the bottom of the rule table, is a "catch-all" rule; packets not matching any other rules will be enforced by the default rule. After the host preparation operation, the DFW default rule is set to 'allow' action. This ensures that VM-to-VM communication is not broken during staging or migration phases. It is a best practice to then change this default rule to 'block' action and enforce access control through a positive control model (i.e., only traffic defined in the firewall policy is allowed onto the network).

Figure 22 steps through policy rule lookup and packet flow:

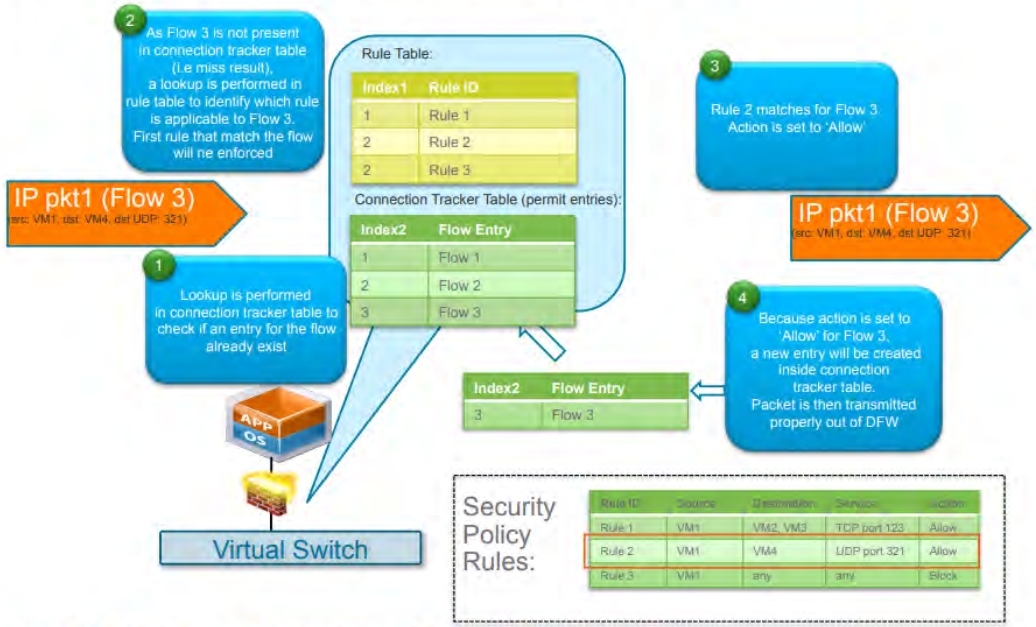
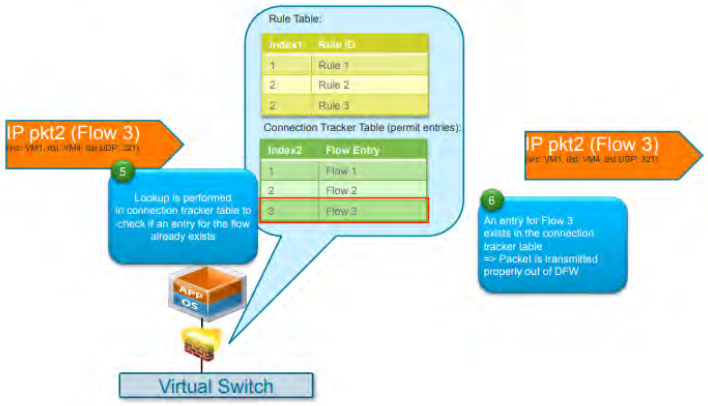
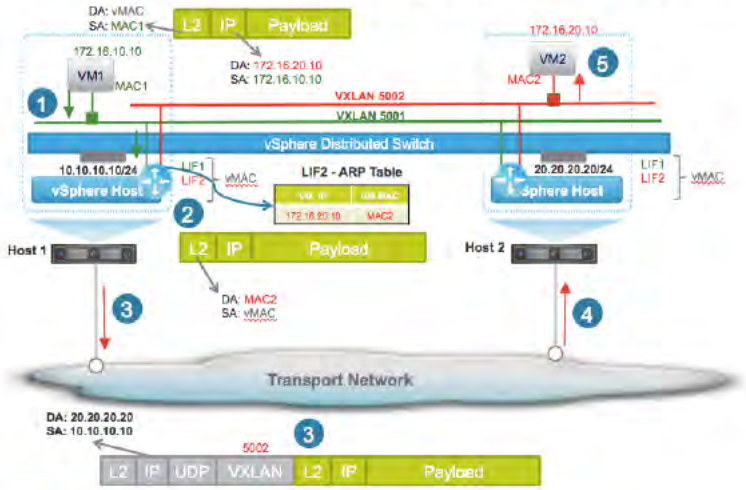


Figure 22 – DFW Policy Rule Lookup and Packet – First Packet

No.	'111 Patent Claim 1	VMware NSX
		NSX Design Guide at 30-31

No.	'111 Patent Claim 1	VMware NSX
		An IP packet identified as pkt1 that matches rule number 2. The order of operation is the following: 30 1. A lookup is performed in the connection tracker table to determine if an entry for the flow already exists. 2. As flow 3 is not present in the connection tracker table, a lookup is performed in the rule table to identify which rule is applicable to flow 3. The first rule that matches the flow will be enforced. 3. Rule 2 matches for flow 3. The action is set to 'Allow'. 4. Because the action is set to 'Allow' for flow 3, a new entry will be created inside the connection tracker table. The packet is then transmitted out of DFW.  Figure 23 – DFW Policy Rule Lookup and Packet – Subsequent Packets.

No.	'111 Patent Claim 1	VMware NSX
		NSX Design Guide at 54 The required steps to establish routed communication between two virtual machines connected to separate logical segments are shown in Figure 46.  Figure 46 - Routed Communication between Virtual Machines 1. VM1 sends a packet to VM2 connected to a different VXLAN segment. The packet is sent to the VM1 default gateway interfaces located on the local DLR (172.16.10.1). 2. A routing lookup is performed on the local DLR, which determines that the destination subnet is directly connected to DLR LIF2. A lookup is performed in the LIF2 ARP table to determine the MAC address associated with the VM2 IP address. If the ARP information is not available, the DLR will generate an ARP request on VXLAN 5002 to determine the required mapping information. 3. An L2 lookup is performed in the local MAC table to determine how to reach VM2. The original packet is VXLAN encapsulated and sent to the VTEP of ESXi2 (20.20.20.20). 4. ESXi-2 decapsulates the packet and performs an L2 lookup in the local MAC table associated to VXLAN 5002 segment. 5. The packet is delivered to the destination VM2.

No.	'111 Patent Claim 1	VMware NSX
		VMW_ORCT_CSCO_. 85

Logical Layer Packet Filtering

NSX provides several capabilities in the logical pipeline that perform packet filtering. Port Security and Port Isolation provide simple filtering models, while Security Profiles provide a richer L3/L4 filtering model. This section details how this filtering is performed, in particular, the details of how NSX automatically matches and permits "reply" traffic for advanced filtering mechanisms like Security Profiles.

For more details on the use cases and configuration details of these features, please see Logical Network Features: Security (page 120).

Transport Layer Implementation

For all logical layer security features, ingress packet filtering is performed by the OVS instance on the hypervisor running the transmitting VM and egress filtering is implemented at the hypervisor running the VM about to receive the traffic.

In the case of Security Profiles, assigning a profile to a logical port requires that the logical port is using the NSX Port Security feature. This is so that the NSX Controller is aware of all valid IP addresses that can be used by a VM, in order to implement filter rules where the remote hosts are identified using a Security Profile UUID (e.g., a rule like "allow tcp 3306 from Webservers").

Interactions between Security Filtering Features

In general, Port Isolation, Port Security, and Security Profiles should be thought of as different stages in the logical pipeline, meaning that traffic must pass all filters in order to be permitted. Thus, even if a connection is allowed by the Security Profile, it will be dropped if it does not match the allowed address pairs for a logical port. For example, even if a Security Profile was configured to allow IP multicast to reach a VM, the multicast traffic would be dropped by Port Security because the destination MAC address did not match an allowed address pair.

One exception to this rule is that Port Security automatically allows non-spoofed ARP (IPv4) and ICMP Neighbor Discovery (IPv6) packets to allow VMs to discover the MAC address associated with an IP address. No Security Profile rules are required to permit such traffic, and it is not possible for Security Profiles to drop such traffic.

Automatically Allows "Reply" Traffic for Existing Connections

With Security Profiles, replies to already permitted traffic are automatically allowed. More precisely:

- Egress traffic that matches an existing connection already allowed by ingress filtering bypasses egress filtering and is automatically allowed.
- Ingress traffic that matches an existing connection allowed by egress filtering bypasses ingress filtering and is automatically allowed.

For egress reply traffic (i.e., replies to an outbound connection initiated by the VM), this is implemented using "connection tracking", which dynamically sets up a flow entry to match return traffic by reversing the TCP or UDP 5-tuple of (IP protocol, source IP address, source TCP/UDP port, destination IP address, destination TCP/UDP port). Egress ICMP ping reply, ICMP TTL-exceeded, and ICMP destination unreachable messages are always allowed on logical port egress. There is no support for "application layer gateway" (ALG) functionality that parses higher level protocols like FTP to open related ports.

No.	'111 Patent Claim 1	VMware NSX
		VMW_ORCT_CSCO_2207 **Edit the Default Distributed Firewall Rule** Default firewall settings apply to traffic that does not match any of the user-defined firewall rules. The Distributed Firewall default rule is displayed on the centralized firewall user interface, and the default rule for each NSX Edge is displayed at the NSX Edge level. The default Distributed Firewall rule allows all L3 and L2 traffic to pass through all prepared clusters in your infrastructure. The default rule is always at the bottom of the rules table and cannot be deleted or added to. However, you can change the Action element of the rule from Allow to Block or Reject, add comments for the rule, and indicate whether traffic for that rule should be logged.

No.	'111 Patent Claim 2	VMware NSX
2[a]	The method according to claim 1, wherein the instruction is 'probe', 'mirror', or 'terminate' instruction, and	VMware NSX System discloses the method according to claim 1, wherein the instruction is 'probe', 'mirror', or 'terminate' instruction. For example, VMware NSX System discloses redirecting, replicating, and blocking rules send by the NSX controller to the vSwitch. NSX Design Guide at 11

No.	'111 Patent Claim 2	VMware NSX
		2.1.2 Control Plane The NSX controller is a key part of the NSX control plane. In a vSphere environment with the vSphere Distributed Switch (VDS), the controller enables multicast free VXLAN and control plane programming of elements such as the Distributed Logical Routing (DLR). Stability and reliability of data transport are central concerns in networking. The NSX controller is a part of the control plane; it is logically separated from all data plane traffic. To further enhance high availability and scalability, NSX controller nodes are deployed in a cluster of odd number instances. In addition to controller, the control VM, provides the routing control plane that allows the local forwarding in ESXi and allows dynamic routing between ESXI and north-south routing provided by Edge VM. It is critical to understand that data plane traffic never traverses the control plane component. NSX Design Guide at 22

No.	'111 Patent Claim 2	VMware NSX
		3.1.7 NSX Distributed Firewall (DFW) The NSX DFW provides L2-L4 stateful firewall services to any workload in the NSX environment. DFW runs in the kernel space and provides near line rate network traffic protection. DFW performance and throughput scale linearly through addition of new ESXi hosts. DFW is activated as soon as the host preparation process is completed. If a VM does not require DFW service, it can be added in the exclusion list functionality. By default, NSX Manager, NSX Controllers, and Edge services gateways are automatically excluded from DFW function. One DFW instance is created per VM vNIC; for example, if a new VM with 3 vNICs is created, 3 instances of DFW will be allocated to this VM. Configuration of these DFW instances can be identical or different based on the "apply to" setting. When a DFW rule is created, the user can select a Point of Enforcement (PEP) for this rule, with options varying from Logical Switch to vNIC. By default, "apply to" is not selected, so DFW rules are propagated down to all the ESXi hosts which are part of the NSX domain that have been prepared for enforcement and applied to all the connected virtual machines. DFW policy rules can be written in 2 ways, using L2 rules (Ethernet) or L3/L4 rules. • L2 rules are mapped to L2 in the OSI model; only MAC addresses can be used in the source and destination fields and only L2 protocols can be used in the service fields (e.g., ARP). • L3/L4 rules are mapped to L3/L4 in the OSI model; policy rules can be written using IP addresses and TCP/UDP ports. It is important to remember that L2 rules are always enforced before L3/L4 rules. If the L2 default policy rule is modified to 'block', and then all L3/L4 traffic will be blocked as well by DFW (e.g., pings would fail).

No.	'111 Patent Claim 2	VMware NSX
		NSX Design Guide at 29 Traffic redirection to third party vendor provides the capability to steer a particular type of traffic to a selected partner services VM. For instance, web traffic from the Internet to a set of Apache or Tomcat servers can be redirected to an L4-L7 deep packet inspection firewall for advanced protection. Traffic redirection is defined under Service Composer/Security Policy for NSX version 6.0 or under Partner Security Services tab of the DFW menu in NSX version 6.1. The Partner Security Services tab provides a powerful and easy to use user interface to define what type of traffic needs to be redirected to which partner services. It follows the same policy definition construct as DFW, providing the same options for source field, destination field and services field. The only difference is in the action field; instead of Block/Allow/Reject, a user can select between redirect/no redirect followed by a partner list. Any partner that has been registered with NSX can be successfully deployed on the platform. Additionally, log options can be enabled for this traffic redirection rule. The DFW instance on an ESXi host contains 2 separate tables. The rule table is used to store all policy rules, while the connection tracker table caches flow entries for rules with permit actions. One DFW instance is permitted per VM vNIC. A specific flow is identified by the 5-tuple information consisting source IP address, destination IP address, protocols, L4 source port, and L4 destination port fields. By default, DFW does not perform a lookup on L4 source port, but it can be configured to do so by defining a specific policy rule. NSX Design Guide at 30

DFW rules are enforced as follows:

- Rules are processed in top-to-bottom ordering.
- Each packet is checked against the top rule in the rule table before moving down the subsequent rules in the table.
- The first rule in the table that matches the traffic parameters is enforced. No subsequent rules can be enforced as the search is then terminated for that packet.

Because of this behavior, it is always recommended to put the most granular policies at the top of the rule table. This will ensure they will be enforced before more specific rules.

The DFW default policy rule, located at the bottom of the rule table, is a "catch-all" rule; packets not matching any other rules will be enforced by the default rule. After the host preparation operation, the DFW default rule is set to 'allow' action. This ensures that VM-to-VM communication is not broken during staging or migration phases. It is a best practice to then change this default rule to 'block' action and enforce access control through a positive control model (i.e., only traffic defined in the firewall policy is allowed onto the network).

Figure 22 steps through policy rule lookup and packet flow:

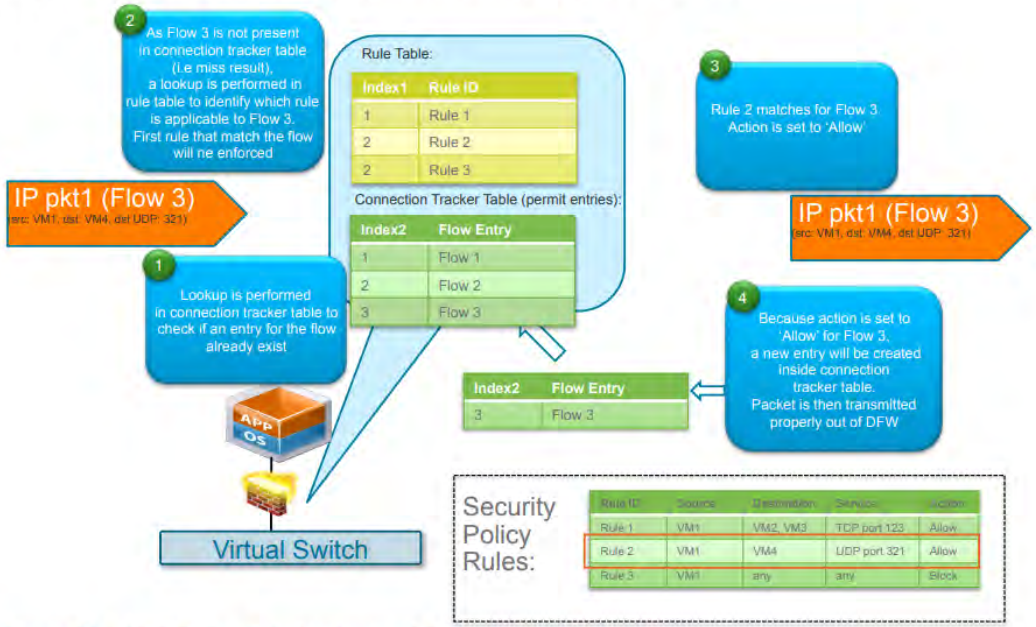


Figure 22 – DFW Policy Rule Lookup and Packet – First Packet

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1836 of 2137

No.	'111 Patent Claim 2	VMware NSX
		NSX Design Guide at 33 4 NSX Functional Services In the context of the NSX architecture, logical networking functionality (e.g., switching, routing, security) can be viewed as having a packet forwarding pipeline that implements multiple features, including L2/L3 forwarding, security filtering, and packet queuing. Unlike a physical switch or router, NSX does not rely on a single device to implement the entire forwarding pipeline. Instead, the NSX controller creates packet processing rules on all relevant ESXi hosts to mimic how a single logical device would handle data. Those hypervisors can be thought of as line cards that implement a portion of the logical network function pipeline. The physical network connecting the ESXi hosts acts as the backplane that carries packets from one line card to the other. VMW_ORCT_CSCO_ 2488

No.	'111 Patent Claim 2	VMware NSX
		Data Plane The NSX Data plane consists of the NSX vSwitch, which is based on the vSphere Distributed Switch (vDS) with additional components to enable services. Kernel modules (VIBs) run within the hypervisor kernel to provide services such as distributed routing and logical firewall and to enable VXLAN bridging capabilities. The NSX vSwitch (vDS-based) abstracts the physical network and provides access-level switching in the hypervisor. It is central to network virtualization because it enables logical networks that are independent of physical constructs such as VLANs. Some of the benefits of the vSwitch are: ■ Support for overlay networking with protocols (such as VXLAN) and centralized network configuration. Overlay networking enables the following capabilities: ■ Creation of a flexible logical Layer 2 (L2) overlay over existing IP networks on existing physical infrastructure without the need to re-architect any of the data center networks ■ Provision of communication (east.west and north.south), while maintaining isolation between tenants ■ Application workloads and virtual machines that are agnostic of the overlay network and operate as if they were connected to a physical L2 network ■ Facilitates massive scale of hypervisors ■ Multiple features.such as Port Mirroring, NetFlow/IPFIX, Configuration Backup and Restore, Network Health Check, QoS, and LACP.provide a comprehensive toolkit for traffic management, monitoring, and troubleshooting within a virtual network Additionally, the data plane consists of gateway devices that can provide L2 bridging from the logical networking space (VXLAN) to the physical network (VLAN). The gateway device is typically an NSX Edge virtual appliance. NSX Edge offers L2, L3, perimeter firewall, load balancing, and other services such as SSL VPN and DHCP. VMW_ORCT_CSCO_76

No.	'111 Patent Claim 2	VMware NSX
		• Port Mirroring—Allows you to duplicate all traffic coming into or out of a logical port. The typical reasons you'll use port mirroring are for troubleshooting and for compliance and monitoring. Port mirroring allows you to forward all monitored traffic to a network appliance for analysis and remediation. The mirrored traffic is sent encapsulated within a GRE Tunnel to a collector so that all the original information of the packets is preserved while traversing the network to a remote location. VMW_ORCT_CSCO_166

No.	'111 Patent Claim 2	VMware NSX

Logical Port Mirroring

Logical port mirroring allows you to duplicate (or "mirror") all traffic coming into or out of a logical switch port that is attached to a VM VIF or to an L2 Gateway port.

The typical reasons you'll use port mirroring are:

- Troubleshooting: Analyze, debug, and diagnose errors on a network by inspecting the traffic.
- Compliance and monitoring: Forward all monitored traffic to a network appliance for analysis and remediation.

The mirrored traffic is sent encapsulated within a GRE Tunnel to a collector so that all the original information of the packets is preserved while traversing the network to a remote location.



Note: Logical port mirroring can only be applied to a logical switch port that is attached to a VM VIF or to an L2 Gateway port. You *cannot* perform mirroring on a logical router port or a patch port.

Why use logical port mirroring in NSX?

In a virtualized environment, NSX logical port mirroring avoids pitfalls often encountered with physical port mirroring:

- Logical port mirroring ensures all VM network traffic is captured. By contrast, if you implement port mirroring only in the physical network, some traffic will fail to be mirrored. This happens because virtual machines frequently communicate with other virtual machines residing on the same hypervisor host. This intra-hypervisor host traffic never enters the physical network and thus never gets mirrored.
- Logical port mirroring allows you to continue mirroring a VM's traffic when that VM is migrated to another hypervisor host.
- NSX allows you to mirror ports of virtual machines in the NSX domain and ports of physical applications in the same manner.

NSX allows you to forward the traffic captured by a workload connected to a logical network and mirror that traffic to a collector with an IP address directly reachable from the IP address of the hypervisor guest on which the virtual machine is actually hosted. This is also true of physical applications connected to Gateway nodes.

No.	'111 Patent Claim 2	VMware NSX
		VMW_ORCT_CSCO_1400 **Using SpoofGuard** After synchronizing with the vCenter Server, NSX Manager collects the IP addresses of all vCenter guest virtual machines from VMware Tools on each virtual machine. If a virtual machine has been compromised, the IP address can be spoofed and malicious transmissions can bypass firewall policies. You create a SpoofGuard policy for specific networks that allows you to authorize the IP addresses reported by VMware Tools and alter them if necessary to prevent spoofing. SpoofGuard inherently trusts the MAC addresses of virtual machines collected from the VMX files and vSphere SDK. Operating separately from Firewall rules, you can use SpoofGuard to block traffic determined to be spoofed. SpoofGuard supports both IPv4 and IPv6 addresses. When using IPv4, the SpoofGuard policy supports a single IP address assigned to a vNIC. IPv6 supports multiple IP addresses assigned to a vNIC. The SpoofGuard policy monitors and manages the IP addresses reported by your virtual machines in one of the following modes. **Automatically Trust IP Assignments On Their First Use** — This mode allows all traffic from your virtual machines to pass while building a table of vNIC-to-IP address assignments. You can review this table at your convenience and make IP address changes. This mode automatically approves all ipv4 and ipv6 address on a vNIC. **Manually Inspect and Approve All IP Assignments Before Use** — This mode blocks all traffic until you approve each vNIC-to-IP address assignment. NOTE SpoofGuard inherently allows DHCP requests regardless of enabled mode. However, if in manual inspection mode, traffic does not pass until the DHCP-assigned IP address has been approved. SpoofGuard includes a system-generated default policy that applies to port groups and logical networks not covered by the other SpoofGuard policies. A newly added network is automatically added to the default policy until you add the network to an existing policy or create a new policy for it. VMW_ORCT_CSCO_1400-01

No.	'111 Patent Claim 2	VMware NSX
		Create a SpoofGuard Policy You can create a SpoofGuard policy to specify the operation mode for specific networks. The system generated policy applies to port groups and logical switches not covered by existing SpoofGuard policies. **Procedure** 1 Log in to the vSphere Web Client. 2 Click **Networking & Security** and then click **SpoofGuard**. 3 Click the **Add** icon. 4 Type a name for the policy. 5 Select **Enabled** or **Disabled** to indicate whether the policy is enabled. 6 For **Operation Mode**, select one of the following: **Option** / **Description** **Automatically Trust IP Assignments on Their First Use** — Select this option to trust all IP assignments upon initial registration with the NSX Manager. **Manually Inspect and Approve All IP Assignments Before Use** — Select this option to require manual approval of all IP addresses. All traffic to and from unapproved IP addresses is blocked. 7 Click **Allow local address as valid address in this namespace** to allow local IP addresses in your setup. When you power on a virtual machine but it is unable to connect to the DHCP server, a local IP address is assigned to it. This local IP address is considered valid only if the SpoofGuard mode is set to **Allow local address as valid address in this namespace**. Otherwise, the local IP address is ignored. 8 Click **Next**. 9 To specify the scope for the policy, click **Add** and select the networks, distributed port groups, or logical switches that this policy should apply to. A port group or logical switch can belong to only one SpoofGuard policy. 10 Click **OK** and then click **Finish**. VMW_ORCT_CSCO_1504-05

No.	'111 Patent Claim 2	VMware NSX
		Configure Flow Monitoring Flow collection must be enabled for you to view traffic information. You can filter the data being displayed by specifying exclusion criterion. For example, you may want to exclude a proxy server to avoid seeing duplicate flows. Or if you are running a Nessus scan on the virtual machines in your inventory, you may not want to exclude the scan flows from being collected. b Specify the required information. **If you selected** / **Specify the following information** **Collect Blocked Flows**: Select No to exclude blocked flows. **Collect Layer2 Flows**: Select No to exclude Layer2 flows. **Source**: Flows are not collected for the specified sources. 1 Click the **Add** icon. 2 In View, select the appropriate container. 3 Select the objects to exclude. **Destination**: Flows are not collected for the specified destinations. 1 Click the **Add** icon. 2 In View, select the appropriate container. 3 Select the objects to exclude. **Destination ports**: Excludes flows to the specified ports. Type the port numbers to exclude. **Service**: Excludes flows for the specified services and service groups. 1 Click the **Add** icon. 2 Select the appropriate services and/or service groups. VMW_ORCT_CSCO_ 2054 **Configuring Fail-Safe Mode for Distributed Firewall** By default, failure or unavailability of the vShield App appliance results in traffic being blocked (fail close). You can change this to allow traffic (fail open).

No.	'111 Patent Claim 2	VMware NSX
		VMW_ORCT_CSCO_ 2063 **Flow Exclusion** Firewalling is done by a kernel module present on each host. This kernel module on each host generates flow records for network activity happening on protected on VMs. These flow records generated on each host are sent to NSX Manager, which consumes the records from all hosts and displays aggregated meaningful information. Due to the vast amount of flow records which can be generated on a host, capability has been provided to exclude generation of flow records by the kernel module as per criteria chosen by administrator. Following knobs are provided to control flow exclusion. All exclusion parameters are applied globally on all hosts. ■ Disable Flows completely at a global level ■ Ignore allowed flows ■ Ignore blocked flows ■ Ignore layer 2 flows ■ Source IPs to ignore. Ex: 10.112.3.14, 10.112.3.15-10.112.3.18,192.168.1.1\24 ■ Source containers to ignore. Container can contain Vm, vNic, IP Set, MAC Set ■ Destination IPs to ignore. ■ Destination containers to ignore. Container can contain Vm, vNic, IP Set, MAC Set ■ Destination ports ■ Service containers to ignore. Container can contain Application or Application group Flow exclusion happens at the source of generation of flow records i.e. host itself. The following flows are discarded by default: ■ Broadcast IP (255.255.255.255) ■ Local multicast group (224.0.0.0/24) ■ Broadcast MAC address (FF:FF:FF:FF:FF:FF) **Exclude Flows** Excludes specified flows. VMW_ORCT_CSCO_ 88

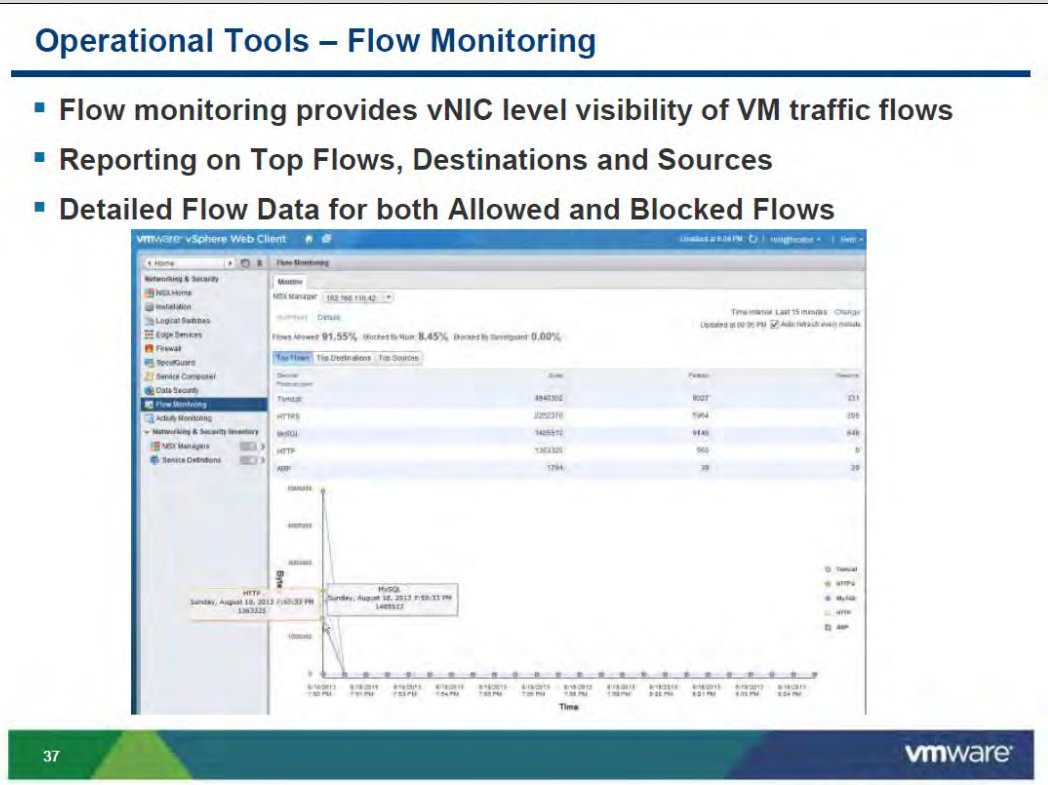
No.	'111 Patent Claim 2	VMware NSX
		Per-flow hashing is used to spread MDI data traffic across all available Service Nodes, for both load sharing and high-availability with respect to Service Node failures. This hashing takes place at two points in the MDI forwarding process: first when the source OVS edge switch (hypervisor or Gateway) chooses a Service Node in the source domain to handle a flow, then again when the Service Node in the source domain chooses a Service Node in the remote domain as the tunnel destination. In both cases, Service Node failures are detected using the standard OVS tunnel monitoring probes, allowing fast-failover in the case of a hardware event on a Service Node. VMW_ORCT_CSCO_ 202 **Manage Auto Configuration Settings** Auto configuration default setting is enabled by default and the priority is high. If you disable auto configuration settings, you must add the required NAT, firewall, routing rules to enable control-channel traffic for other services such as load balancing, VPN, etc. If you change the priority of the auto configuration settings to low, the internal/auto configured rules are placed in lower precedence than the rules you create. With this, you can again control special allow/deny rules for these services too. For example, you can block specific IP addresses from accessing the VPN services. VMW_ORCT_CSCO_ 203

No.	'111 Patent Claim 2	VMware NSX
		Distributed Firewall Management 9 Distributed Firewall is a hypervisor kernel-embedded firewall that provides visibility and control for virtualized workloads and networks. You can create access control policies based on VMware vCenter objects like datacenters and clusters, virtual machine names and tags, network constructs like IP/VLAN/VXLAN addresses, as well as user group identity from Active Directory. Firewall rules are enforced at the vNIC level of each virtual machine to provide consistent access control even when the virtual machine gets vMotioned. The hypervisor-embedded nature of the firewall delivers close to line rate throughput to enable higher workload consolidation on physical servers. The distributed nature of the firewall provides a scale-out architecture that automatically extends firewall capacity when additional hosts are added to a prepared cluster. Distributed Firewall offers multiple sets of configurable rules: Layer 3 (L3) rules (General tab) and Layer 2 (L2) rules (Ethernet tab). Layer 2 firewall rules are processed before Layer 3 rules. The default firewall rule allows all L3 and L2 traffic to pass through all clusters in your infrastructure. The default rule is always at the bottom of the rules table and cannot be deleted or added to. However, you can change the Action element of the rule from Allow to Block, add comments for the rule, and indicate whether traffic for that rule should be logged. Firewall rules are created at the global scope, but you can then narrow the scope (datacenter, cluster, distributed virtual port group, network, virtual machine, vNIC, or virtual wire) at which you want to apply the rule by using the AppliedTo keyword. User defined firewall rules are enforced in top-to-bottom ordering, with a per-virtual NIC level precedence. Each traffic session is checked against the top rule in the Firewall table before moving down the subsequent rules in the table. The first rule in the table that matches the traffic parameters is enforced. VMW_ORCT_CSCO_120

No.	'111 Patent Claim 2	VMware NSX
		Field / *Description* **Management Rendezvous Server** — (Optional) A Service Node designated as Management Rendezvous server will proxy management traffic between the NSX Controller Cluster and remotely deployed Gateways. See Configure an NSX Controller Cluster or Rendezvous Server Management IP Address (page 308). **Admin Status Enabled** — Un-check this checkbox to disable this transport node. **Tunnel Keep-alive Spray** — To more accurately test the health of a transport node's tunnels, turn on the **Tunnel Keep-alive Spray** option. Randomizes TCP source port for STT tunnel keep-alives for better CFM packet spray across active network paths. With this option turned *on*, OVS sends CFM probes over randomly chosen VLANs in the tunnel, thereby providing a more thorough test. With this option turned *off* (the default), OVS always sends the CFMs over the same VLAN (that is, over the same physical network path). For transport nodes that run as VMs, and in all situations in which you wish to ensure that CFM probes always travel the same physical network path, you must turn *off* the **Tunnel Keep-alive Spray** option. See 3. Tunnel Troubleshooting (page 342) for instructions on monitoring tunnel health. VMW_ORCT_CSCO_128

No.	'111 Patent Claim 2	VMware NSX
		Gateways in the L2 Gateway Service are responsible for choosing the active Gateway to forward traffic. The leader election process is designed to quickly fail over to a backup Gateway when the active fails, while preventing multiple Gateways from becoming active concurrently (potentially causing loops). In addition, it is responsible for detecting merges of multiple broadcast domains, or partitions of existing ones. In an L2 Gateway Service, Gateways use a modified version of 802.1ag CFM as leader election probes. Leader election probes are sent on an L2 multicast address so they reach every Gateway in the relevant broadcast domain. The Gateway with the highest priority (lowest numerical CFM ID value) is chosen as the leader and designated the active Gateway for the L2 Gateway Service. Gateways other than the active Gateway do not forward data packets to or from logical networks. Failure detection with the CFM probes occurs within 1-2 seconds. No interaction with the NSX Controller Cluster is required, meaning that failover can occur even when the Gateways cannot reach the NSX Controller. See **L2 Gateway Commands (page 221)** for information on setting failover options for a high availability L2 Gateway Service. VMW_ORCT_CSCO_376 *Verifying Tunnel Status* When implementing Overlay Logical Networks, NSX uses L2-in-L3 tunnels between transport nodes to create logical network topologies that are decoupled from the physical network topology. Problems with tunnel connectivity negatively impact the ability of NSX to create logical networks. It can be compared to having a faulty backplane in a modular switch. To highlight problems with tunnel connectivity, the NSX Controller Cluster monitors the status of all tunnels using active probes. OVS and NSX assess network tunnel health by sending CFM probe packets over the network tunnels that connect transport nodes. Based on the packet loss observed, OVS gives the tunnel a health rating between 0 (unhealthy) and 100 (healthy). You can view tunnel health assessments in NSX Manager (open the transport node whose tunnels you wish to check, open the Tunnels section, and check the Health Status column for each tunnel) and from the OVS command line (use "ovs-appctl show").

No.	'111 Patent Claim 2	VMware NSX
		VMW_ORCT_CSCO_ 2212 10 Point to the **Action** cell of the new rule and click ⊞. Make appropriate selections as described in the table below and click **OK**. **Action** / **Results in** **Allow** — Allows traffic from or to the specified source(s), destination(s), and service(s). **Block** — Blocks traffic from or to the specified source(s), destination(s), and service(s). **Reject** — Sends reject message for unaccepted packets. RST packets are sent for TCP connections. ICMP messages with administratively prohibited code are sent for UDP, ICMP, and other IP connections. **Log** — Logs all sessions matching this rule. Enabling logging can affect performance. **Do not log** — Does not log sessions. VMW_ORCT_CSCO_ 4519

No.	'111 Patent Claim 2	VMware NSX
		 VMW_ORCT_CSCO_004519
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	VMware NSX 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. For example, VMware NSX System discloses a blocking rule, which, when enforced, blocks data traffic from being sent to the destination endpoint, the controller, or any other device. NSX Design Guide at 22

No.	'111 Patent Claim 2	VMware NSX
	entity and to the controller.	**3.1.7 NSX Distributed Firewall (DFW)** The NSX DFW provides L2-L4 stateful firewall services to any workload in the NSX environment. DFW runs in the kernel space and provides near line rate network traffic protection. DFW performance and throughput scale linearly through addition of new ESXi hosts. DFW is activated as soon as the host preparation process is completed. If a VM does not require DFW service, it can be added in the exclusion list functionality. By default, NSX Manager, NSX Controllers, and Edge services gateways are automatically excluded from DFW function. One DFW instance is created per VM vNIC; for example, if a new VM with 3 vNICs is created, 3 instances of DFW will be allocated to this VM. Configuration of these DFW instances can be identical or different based on the "apply to" setting. When a DFW rule is created, the user can select a Point of Enforcement (PEP) for this rule, with options varying from Logical Switch to vNIC. By default, "apply to" is not selected, so DFW rules are propagated down to all the ESXi hosts which are part of the NSX domain that have been prepared for enforcement and applied to all the connected virtual machines. DFW policy rules can be written in 2 ways, using L2 rules (Ethernet) or L3/L4 rules. • L2 rules are mapped to L2 in the OSI model; only MAC addresses can be used in the source and destination fields and only L2 protocols can be used in the service fields (e.g., ARP). • L3/L4 rules are mapped to L3/L4 in the OSI model; policy rules can be written using IP addresses and TCP/UDP ports. It is important to remember that L2 rules are always enforced before L3/L4 rules. If the L2 default policy rule is modified to 'block', and then all L3/L4 traffic will be blocked as well by DFW (e.g., pings would fail).

No.	'111 Patent Claim 2	VMware NSX
		NSX Design Guide at 29 Traffic redirection to third party vendor provides the capability to steer a particular type of traffic to a selected partner services VM. For instance, web traffic from the Internet to a set of Apache or Tomcat servers can be redirected to an L4-L7 deep packet inspection firewall for advanced protection. Traffic redirection is defined under Service Composer/Security Policy for NSX version 6.0 or under Partner Security Services tab of the DFW menu in NSX version 6.1. The Partner Security Services tab provides a powerful and easy to use user interface to define what type of traffic needs to be redirected to which partner services. It follows the same policy definition construct as DFW, providing the same options for source field, destination field and services field. The only difference is in the action field; instead of Block/Allow/Reject, a user can select between redirect/no redirect followed by a partner list. Any partner that has been registered with NSX can be successfully deployed on the platform. Additionally, log options can be enabled for this traffic redirection rule. The DFW instance on an ESXi host contains 2 separate tables. The rule table is used to store all policy rules, while the connection tracker table caches flow entries for rules with permit actions. One DFW instance is permitted per VM vNIC. A specific flow is identified by the 5-tuple information consisting source IP address, destination IP address, protocols, L4 source port, and L4 destination port fields. By default, DFW does not perform a lookup on L4 source port, but it can be configured to do so by defining a specific policy rule. VMW_ORCT_CSCO_ 140

No.	'111 Patent Claim 2	VMware NSX
		Edit the Default Edge Firewall Rule Default firewall settings apply to traffic that does not match any of the user-defined firewall rules. The default Edge firewall policy blocks all incoming traffic. You can change the default action and logging settings. **Procedure** 1 Log in to the vSphere Web Client. 2 Click **Networking & Security** and then click **NSX Edges**. 3 Double-click an NSX Edge. 4 Click the **Manage** tab and then click **Firewall**. 5 Select the **Default Rule**, which is the last rule in the firewall table. 6 Point to the **Action** cell of the new rule and click  . a Click **Accept** to allow traffic from or to the specified source and destination. b Click **Log** to log all sessions matching this rule. Enabling logging can affect performance. c Type comments if required. d Click **OK**. VMW_ORCT_CSCO_ 202 **Manage Auto Configuration Settings** Auto configuration default setting is enabled by default and the priority is high. If you disable auto configuration settings, you must add the required NAT, firewall, routing rules to enable control-channel traffic for other services such as load balancing, VPN, etc. If you change the priority of the auto configuration settings to low, the internal/auto configured rules are placed in lower precedence than the rules you create. With this, you can again control special allow/deny rules for these services too. For example, you can block specific IP addresses from accessing the VPN services. VMW_ORCT_CSCO_ 1400

No.	'111 Patent Claim 2	VMware NSX
		Using SpoofGuard After synchronizing with the vCenter Server, NSX Manager collects the IP addresses of all vCenter guest virtual machines from VMware Tools on each virtual machine. If a virtual machine has been compromised, the IP address can be spoofed and malicious transmissions can bypass firewall policies. You create a SpoofGuard policy for specific networks that allows you to authorize the IP addresses reported by VMware Tools and alter them if necessary to prevent spoofing. SpoofGuard inherently trusts the MAC addresses of virtual machines collected from the VMX files and vSphere SDK. Operating separately from Firewall rules, you can use SpoofGuard to block traffic determined to be spoofed. SpoofGuard supports both IPv4 and IPv6 addresses. When using IPv4, the SpoofGuard policy supports a single IP address assigned to a vNIC. IPv6 supports multiple IP addresses assigned to a vNIC. The SpoofGuard policy monitors and manages the IP addresses reported by your virtual machines in one of the following modes. **Automatically Trust IP Assignments On Their First Use** — This mode allows all traffic from your virtual machines to pass while building a table of vNIC-to-IP address assignments. You can review this table at your convenience and make IP address changes. This mode automatically approves all ipv4 and ipv6 address on a vNIC. **Manually Inspect and Approve All IP Assignments Before Use** — This mode blocks all traffic until you approve each vNIC-to-IP address assignment. NOTE SpoofGuard inherently allows DHCP requests regardless of enabled mode. However, if in manual inspection mode, traffic does not pass until the DHCP-assigned IP address has been approved. SpoofGuard includes a system-generated default policy that applies to port groups and logical networks not covered by the other SpoofGuard policies. A newly added network is automatically added to the default policy until you add the network to an existing policy or create a new policy for it. VMW_ORCT_CSCO_1400-01

No.	'111 Patent Claim 2	VMware NSX
		Create a SpoofGuard Policy You can create a SpoofGuard policy to specify the operation mode for specific networks. The system generated policy applies to port groups and logical switches not covered by existing SpoofGuard policies. **Procedure** 1 Log in to the vSphere Web Client. 2 Click **Networking & Security** and then click **SpoofGuard**. 3 Click the **Add** icon. 4 Type a name for the policy. 5 Select **Enabled** or **Disabled** to indicate whether the policy is enabled. 6 For **Operation Mode**, select one of the following: **Option** / **Description** **Automatically Trust IP Assignments on Their First Use** — Select this option to trust all IP assignments upon initial registration with the NSX Manager. **Manually Inspect and Approve All IP Assignments Before Use** — Select this option to require manual approval of all IP addresses. All traffic to and from unapproved IP addresses is blocked. 7 Click **Allow local address as valid address in this namespace** to allow local IP addresses in your setup. When you power on a virtual machine but it is unable to connect to the DHCP server, a local IP address is assigned to it. This local IP address is considered valid only if the SpoofGuard mode is set to **Allow local address as valid address in this namespace**. Otherwise, the local IP address is ignored. 8 Click **Next**. 9 To specify the scope for the policy, click **Add** and select the networks, distributed port groups, or logical switches that this policy should apply to. A port group or logical switch can belong to only one SpoofGuard policy. 10 Click **OK** and then click **Finish**. VMW_ORCT_CSCO_1504-05

No.	'111 Patent Claim 2	VMware NSX
		Configure Flow Monitoring Flow collection must be enabled for you to view traffic information. You can filter the data being displayed by specifying exclusion criterion. For example, you may want to exclude a proxy server to avoid seeing duplicate flows. Or if you are running a Nessus scan on the virtual machines in your inventory, you may not want to exclude the scan flows from being collected. b Specify the required information.

If you selected	Specify the following information
Collect Blocked Flows	Select No to exclude blocked flows.
Collect Layer2 Flows	Select No to exclude Layer2 flows.
Source	Flows are not collected for the specified sources. 1 Click the **Add** icon. 2 In View, select the appropriate container. 3 Select the objects to exclude.
Destination	Flows are not collected for the specified destinations. 1 Click the **Add** icon. 2 In View, select the appropriate container. 3 Select the objects to exclude.
Destination ports	Excludes flows to the specified ports. Type the port numbers to exclude.
Service	Excludes flows for the specified services and service groups. 1 Click the **Add** icon. 2 Select the appropriate services and/or service groups.

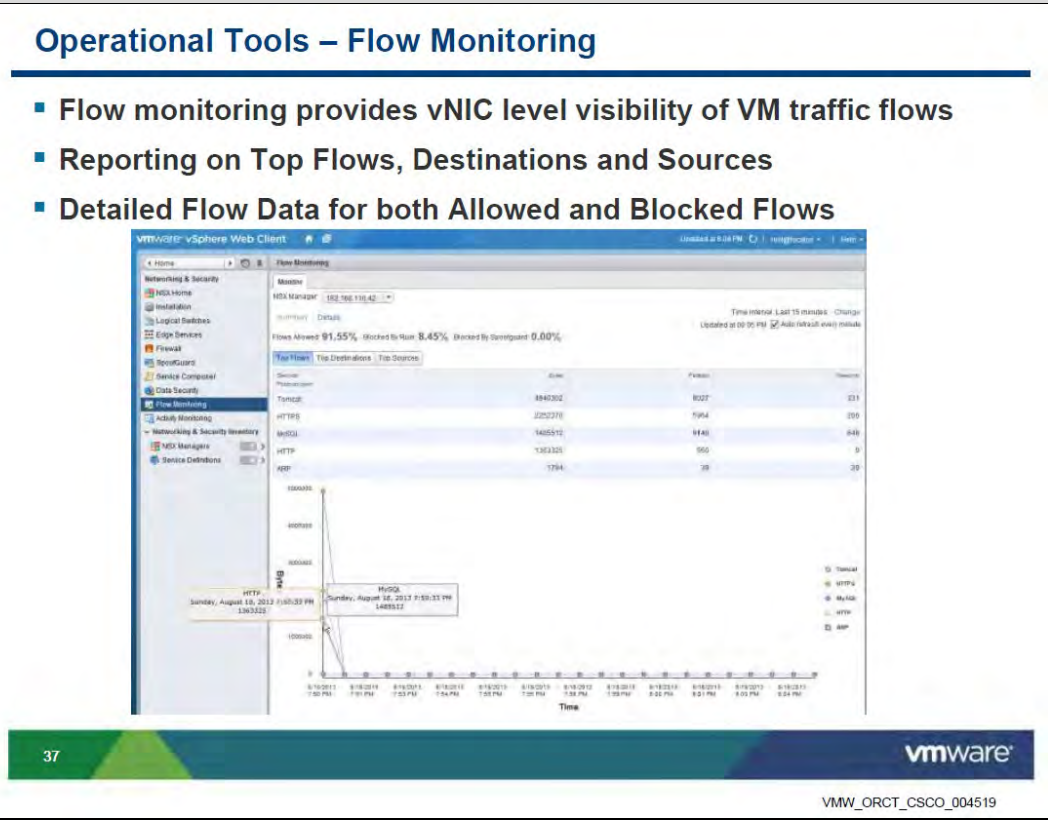
VMW_ORCT_CSCO_ 2054

Configuring Fail-Safe Mode for Distributed Firewall

By default, failure or unavailability of the vShield App appliance results in traffic being blocked (fail close). You can change this to allow traffic (fail open).

No.	'111 Patent Claim 2	VMware NSX
		VMW_ORCT_CSCO_ 2063 **Flow Exclusion** Firewalling is done by a kernel module present on each host. This kernel module on each host generates flow records for network activity happening on protected on VMs. These flow records generated on each host are sent to NSX Manager, which consumes the records from all hosts and displays aggregated meaningful information. Due to the vast amount of flow records which can be generated on a host, capability has been provided to exclude generation of flow records by the kernel module as per criteria chosen by administrator. Following knobs are provided to control flow exclusion. All exclusion parameters are applied globally on all hosts. ■ Disable Flows completely at a global level ■ Ignore allowed flows ■ Ignore blocked flows ■ Ignore layer 2 flows ■ Source IPs to ignore. Ex: 10.112.3.14, 10.112.3.15-10.112.3.18,192.168.1.1\24 ■ Source containers to ignore. Container can contain Vm, vNic, IP Set, MAC Set ■ Destination IPs to ignore. ■ Destination containers to ignore. Container can contain Vm, vNic, IP Set, MAC Set ■ Destination ports ■ Service containers to ignore. Container can contain Application or Application group Flow exclusion happens at the source of generation of flow records i.e. host itself. The following flows are discarded by default: ■ Broadcast IP (255.255.255.255) ■ Local multicast group (224.0.0.0/24) ■ Broadcast MAC address (FF:FF:FF:FF:FF:FF) **Exclude Flows** Excludes specified flows. VMW_ORCT_CSCO_ 2212

No.	'111 Patent Claim 2	VMware NSX
		10 Point to the **Action** cell of the new rule and click ⊞. Make appropriate selections as described in the table below and click **OK**. **Action** — **Results in** **Allow** — Allows traffic from or to the specified source(s), destination(s), and service(s). **Block** — Blocks traffic from or to the specified source(s), destination(s), and service(s). **Reject** — Sends reject message for unaccepted packets. RST packets are sent for TCP connections. ICMP messages with administratively prohibited code are sent for UDP, ICMP, and other IP connections. **Log** — Logs all sessions matching this rule. Enabling logging can affect performance. **Do not log** — Does not log sessions. VMW_ORCT_CSCO_4519

No.	'111 Patent Claim 2	VMware NSX
		

No.	'111 Patent Claim 3	VMware NSX
3[a]	The method according to claim 1, wherein the instruction is a 'probe', a 'mirror', or a 'terminate' instruction, and	VMware NSX System discloses the method according to claim 1, wherein the instruction is a 'probe', a 'mirror', or a 'terminate' instruction. *See supra* at 2(a).

No.	'111 Patent Claim 3	VMware NSX
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.	VMware NSX System 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. For example, VMware NSX System discloses a replicating mechanism in which traffic originating from one virtual machine is replicated to be sent to multiple destinations, which may include the destination endpoint and the NSX controller. NSX Design Guide at 29

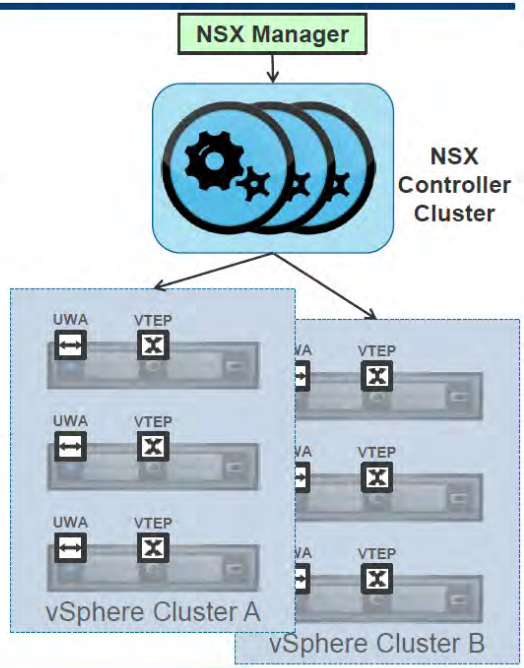
No.	'111 Patent Claim 3	VMware NSX
		Traffic redirection to third party vendor provides the capability to steer a particular type of traffic to a selected partner services VM. For instance, web traffic from the Internet to a set of Apache or Tomcat servers can be redirected to an L4-L7 deep packet inspection firewall for advanced protection. Traffic redirection is defined under Service Composer/Security Policy for NSX version 6.0 or under Partner Security Services tab of the DFW menu in NSX version 6.1. The Partner Security Services tab provides a powerful and easy to use user interface to define what type of traffic needs to be redirected to which partner services. It follows the same policy definition construct as DFW, providing the same options for source field, destination field and services field. The only difference is in the action field; instead of Block/Allow/Reject, a user can select between redirect/no redirect followed by a partner list. Any partner that has been registered with NSX can be successfully deployed on the platform. Additionally, log options can be enabled for this traffic redirection rule. The DFW instance on an ESXi host contains 2 separate tables. The rule table is used to store all policy rules, while the connection tracker table caches flow entries for rules with permit actions. One DFW instance is permitted per VM vNIC. A specific flow is identified by the 5-tuple information consisting source IP address, destination IP address, protocols, L4 source port, and L4 destination port fields. By default, DFW does not perform a lookup on L4 source port, but it can be configured to do so by defining a specific policy rule. NSX Design Guide at 35

No.	'111 Patent Claim 3	VMware NSX
		4.2.1 Replication Modes for Multi-Destination Traffic When two VMs connected to different ESXi hosts need to communicate directly, unicast VXLAN-encapsulated traffic is exchanged between the VTEP IP addresses associated with their associated hypervisors. Traffic originated by a VM may also need to be sent to all the other VMs belonging to the same logical switch. Specific instances of this type of L2 traffic include broadcast, unknown unicast, and multicast. These multi-destination traffic types are collectively referred to using the acronym BUM (Broadcast, Unknown Unicast, Multicast). In each of these scenarios, traffic originated by a given ESXi host must be replicated to multiple remote. NSX supports three different replications modes to enable multi-destination communication on VXLAN backed Logical Switches: multicast, unicast and hybrid. By default, a Logical Switch inherits its replication mode from the Transport Zone, though this behavior can be overridden at the Logical Switch level. VMW_ORCT_CSCO_2488

No.	'111 Patent Claim 3	VMware NSX
		Data Plane The NSX Data plane consists of the NSX vSwitch, which is based on the vSphere Distributed Switch (vDS) with additional components to enable services. Kernel modules (VIBs) run within the hypervisor kernel to provide services such as distributed routing and logical firewall and to enable VXLAN bridging capabilities. The NSX vSwitch (vDS-based) abstracts the physical network and provides access-level switching in the hypervisor. It is central to network virtualization because it enables logical networks that are independent of physical constructs such as VLANs. Some of the benefits of the vSwitch are: ■ Support for overlay networking with protocols (such as VXLAN) and centralized network configuration. Overlay networking enables the following capabilities: ■ Creation of a flexible logical Layer 2 (L2) overlay over existing IP networks on existing physical infrastructure without the need to re-architect any of the data center networks ■ Provision of communication (east.west and north.south), while maintaining isolation between tenants ■ Application workloads and virtual machines that are agnostic of the overlay network and operate as if they were connected to a physical L2 network ■ Facilitates massive scale of hypervisors ■ Multiple features.such as Port Mirroring, NetFlow/IPFIX, Configuration Backup and Restore, Network Health Check, QoS, and LACP.provide a comprehensive toolkit for traffic management, monitoring, and troubleshooting within a virtual network Additionally, the data plane consists of gateway devices that can provide L2 bridging from the logical networking space (VXLAN) to the physical network (VLAN). The gateway device is typically an NSX Edge virtual appliance. NSX Edge offers L2, L3, perimeter firewall, load balancing, and other services such as SSL VPN and DHCP. VMW_ORCT_CSCO_76

No.	'111 Patent Claim 3	VMware NSX
		• Port Mirroring—Allows you to duplicate all traffic coming into or out of a logical port. The typical reasons you'll use port mirroring are for troubleshooting and for compliance and monitoring. Port mirroring allows you to forward all monitored traffic to a network appliance for analysis and remediation. The mirrored traffic is sent encapsulated within a GRE Tunnel to a collector so that all the original information of the packets is preserved while traversing the network to a remote location. VMW_ORCT_CSCO_166

No.	'111 Patent Claim 3	VMware NSX
		## *Logical Port Mirroring* Logical port mirroring allows you to duplicate (or "mirror") all traffic coming into or out of a logical switch port that is attached to a VM VIF or to an L2 Gateway port. The typical reasons you'll use port mirroring are: • Troubleshooting: Analyze, debug, and diagnose errors on a network by inspecting the traffic. • Compliance and monitoring: Forward all monitored traffic to a network appliance for analysis and remediation. The mirrored traffic is sent encapsulated within a GRE Tunnel to a collector so that all the original information of the packets is preserved while traversing the network to a remote location.  **Note:** Logical port mirroring can only be applied to a logical switch port that is attached to a VM VIF or to an L2 Gateway port. You *cannot* perform mirroring on a logical router port or a patch port. ### *Why use logical port mirroring in NSX?* In a virtualized environment, NSX logical port mirroring avoids pitfalls often encountered with physical port mirroring: • Logical port mirroring ensures all VM network traffic is captured. By contrast, if you implement port mirroring only in the physical network, some traffic will fail to be mirrored. This happens because virtual machines frequently communicate with other virtual machines residing on the same hypervisor host. This intra-hypervisor host traffic never enters the physical network and thus never gets mirrored. • Logical port mirroring allows you to continue mirroring a VM's traffic when that VM is migrated to another hypervisor host. • NSX allows you to mirror ports of virtual machines in the NSX domain and ports of physical applications in the same manner. NSX allows you to forward the traffic captured by a workload connected to a logical network and mirror that traffic to a collector with an IP address directly reachable from the IP address of the hypervisor guest on which the virtual machine is actually hosted. This is also true of physical applications connected to Gateway nodes.

No.	'111 Patent Claim 3	VMware NSX
		VMW_ORCT_CSCO_ 95 **logical port mirroring** Logical port mirroring allows you to duplicate (or "mirror") all traffic coming into or out of a logical switch port. VMW_ORCT_CSCO_ 4506 **Operational Tools – NSX Controller** ▪ Control Plane basics ▪ ESXi hosts and NSX Edge Logical Router VMs collect network information, which is then reported to the Controller via User World Agent (UWA) ▪ The NSX Controller CLI provides a consistent, centralized interface to verify VXLAN and Logical Routing network state information ▪ NSX Manager also provides APIs to programmatically retrieve data from the controller nodes  24 vmware VMW_ORCT_CSCO_004506

No.	'111 Patent Claim 4	VMware NSX
4[a]	The method according to claim 1, wherein the instruction is 'probe', 'mirror', or 'terminate' instruction, and	VMware NSX System discloses the method according to claim 1, wherein the instruction is 'probe', 'mirror', or 'terminate' instruction. *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;	VMware NSX 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. For example, VMware NSX System discloses checking by the vSwitch if the packet matches the rule, and redirecting matching traffic to service devices. A person of ordinary skill in the art would understand that packet traffic could be redirected to the NSX controller. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the VMware NSX 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. NSX Design Guide at 29

No.	'111 Patent Claim 4	VMware NSX
		Traffic redirection to third party vendor provides the capability to steer a particular type of traffic to a selected partner services VM. For instance, web traffic from the Internet to a set of Apache or Tomcat servers can be redirected to an L4-L7 deep packet inspection firewall for advanced protection. Traffic redirection is defined under Service Composer/Security Policy for NSX version 6.0 or under Partner Security Services tab of the DFW menu in NSX version 6.1. The Partner Security Services tab provides a powerful and easy to use user interface to define what type of traffic needs to be redirected to which partner services. It follows the same policy definition construct as DFW, providing the same options for source field, destination field and services field. The only difference is in the action field; instead of Block/Allow/Reject, a user can select between redirect/no redirect followed by a partner list. Any partner that has been registered with NSX can be successfully deployed on the platform. Additionally, log options can be enabled for this traffic redirection rule. The DFW instance on an ESXi host contains 2 separate tables. The rule table is used to store all policy rules, while the connection tracker table caches flow entries for rules with permit actions. One DFW instance is permitted per VM vNIC. A specific flow is identified by the 5-tuple information consisting source IP address, destination IP address, protocols, L4 source port, and L4 destination port fields. By default, DFW does not perform a lookup on L4 source port, but it can be configured to do so by defining a specific policy rule. NSX Design Guide at 30

DFW rules are enforced as follows:

- Rules are processed in top-to-bottom ordering.
- Each packet is checked against the top rule in the rule table before moving down the subsequent rules in the table.
- The first rule in the table that matches the traffic parameters is enforced. No subsequent rules can be enforced as the search is then terminated for that packet.

Because of this behavior, it is always recommended to put the most granular policies at the top of the rule table. This will ensure they will be enforced before more specific rules.

The DFW default policy rule, located at the bottom of the rule table, is a "catch-all" rule; packets not matching any other rules will be enforced by the default rule. After the host preparation operation, the DFW default rule is set to 'allow' action. This ensures that VM-to-VM communication is not broken during staging or migration phases. It is a best practice to then change this default rule to 'block' action and enforce access control through a positive control model (i.e., only traffic defined in the firewall policy is allowed onto the network).

Figure 22 steps through policy rule lookup and packet flow:

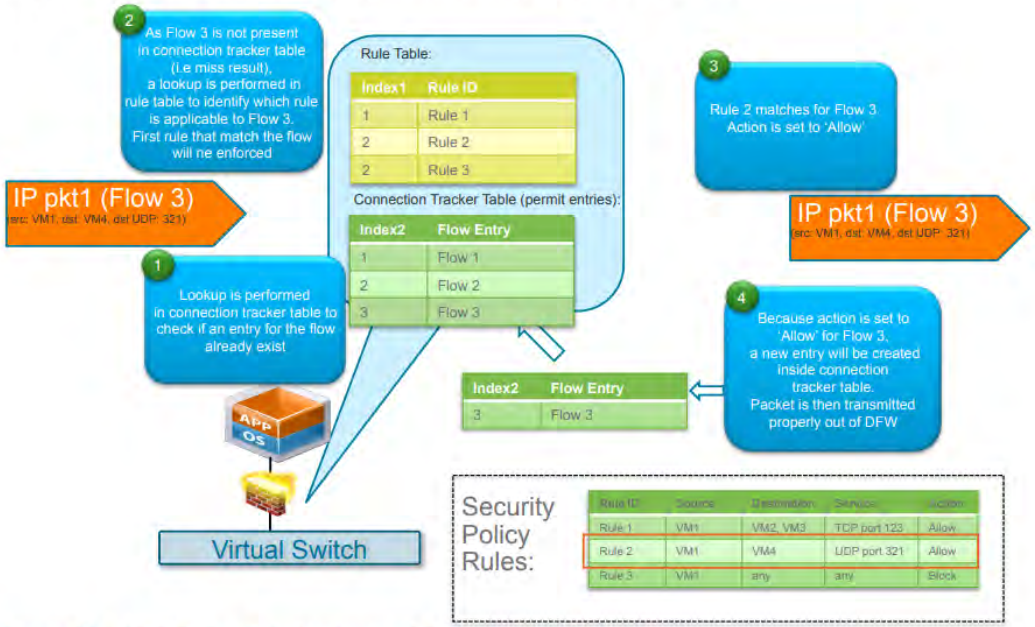
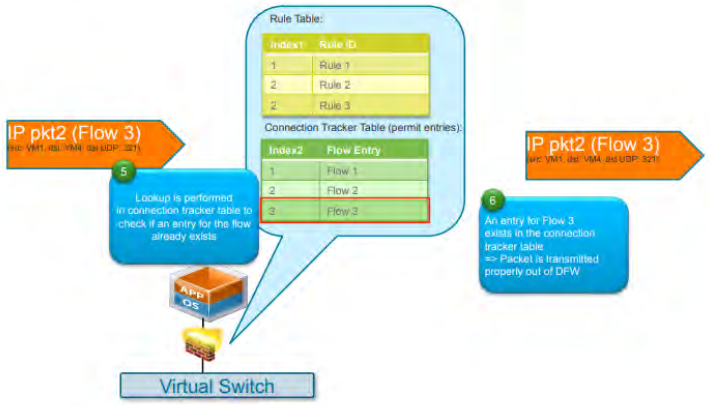


Figure 22 – DFW Policy Rule Lookup and Packet – First Packet

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1869 of 2137

No.	'111 Patent Claim 4	VMware NSX
		NSX Design Guide at 30-31 An IP packet identified as pkt1 that matches rule number 2. The order of operation is the following: 30 1. A lookup is performed in the connection tracker table to determine if an entry for the flow already exists. 2. As flow 3 is not present in the connection tracker table, a lookup is performed in the rule table to identify which rule is applicable to flow 3. The first rule that matches the flow will be enforced. 3. Rule 2 matches for flow 3. The action is set to 'Allow'. 4. Because the action is set to 'Allow' for flow 3, a new entry will be created inside the connection tracker table. The packet is then transmitted out of DFW.  Figure 23 – DFW Policy Rule Lookup and Packet – Subsequent Packets.

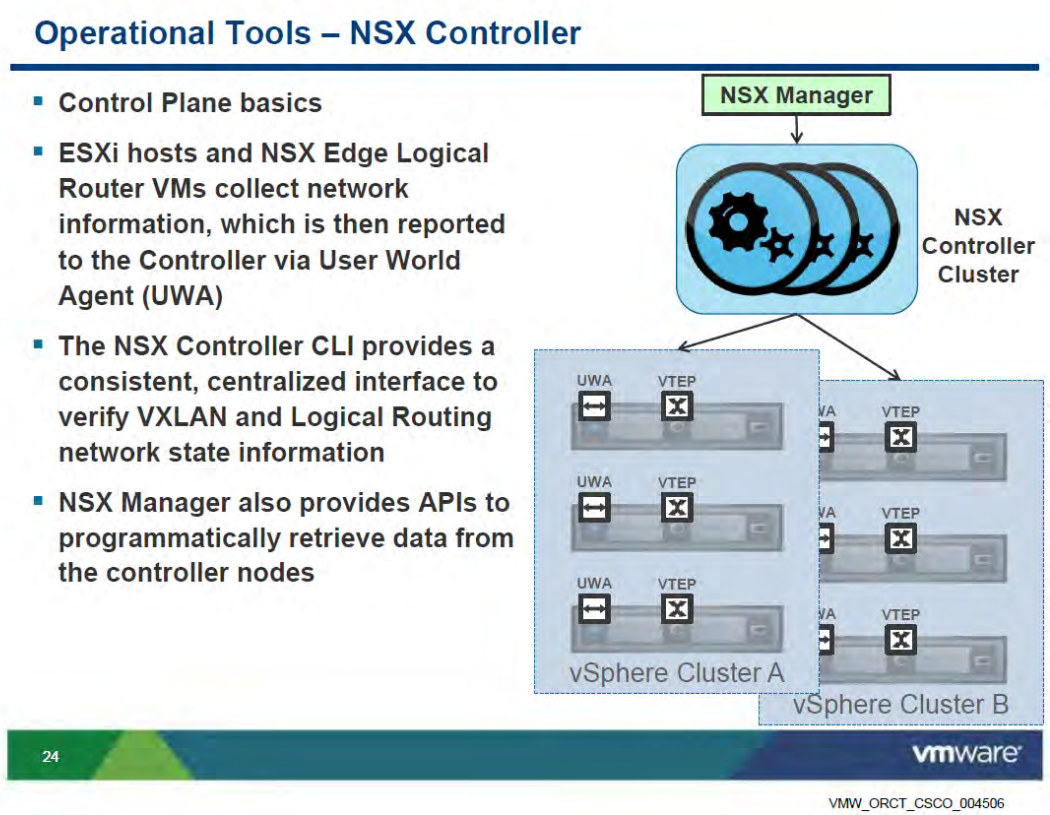
No.	'111 Patent Claim 4	VMware NSX
		VMW_ORCT_CSCO_ 88 Per-flow hashing is used to spread MDI data traffic across all available Service Nodes, for both load sharing and high-availability with respect to Service Node failures. This hashing takes place at two points in the MDI forwarding process: first when the source OVS edge switch (hypervisor or Gateway) chooses a Service Node in the source domain to handle a flow, then again when the Service Node in the source domain chooses a Service Node in the remote domain as the tunnel destination. In both cases, Service Node failures are detected using the standard OVS tunnel monitoring probes, allowing fast-failover in the case of a hardware event on a Service Node. VMW_ORCT_CSCO_ 120

No.	'111 Patent Claim 4	VMware NSX

Field	Description
Management Rendezvous Server	(Optional) A Service Node designated as Management Rendezvous server will proxy management traffic between the NSX Controller Cluster and remotely deployed Gateways. See Configure an NSX Controller Cluster or Rendezvous Server Management IP Address (page 308).
Admin Status Enabled	Un-check this checkbox to disable this transport node.
Tunnel Keep-alive Spray	To more accurately test the health of a transport node's tunnels, turn on the **Tunnel Keep-alive Spray** option. Randomizes TCP source port for STT tunnel keep-alives for better CFM packet spray across active network paths. With this option turned *on*, OVS sends CFM probes over randomly chosen VLANs in the tunnel, thereby providing a more thorough test. With this option turned *off* (the default), OVS always sends the CFMs over the same VLAN (that is, over the same physical network path). For transport nodes that run as VMs, and in all situations in which you wish to ensure that CFM probes always travel the same physical network path, you must turn *off* the **Tunnel Keep-alive Spray** option. See 3. Tunnel Troubleshooting (page 342) for instructions on monitoring tunnel health.

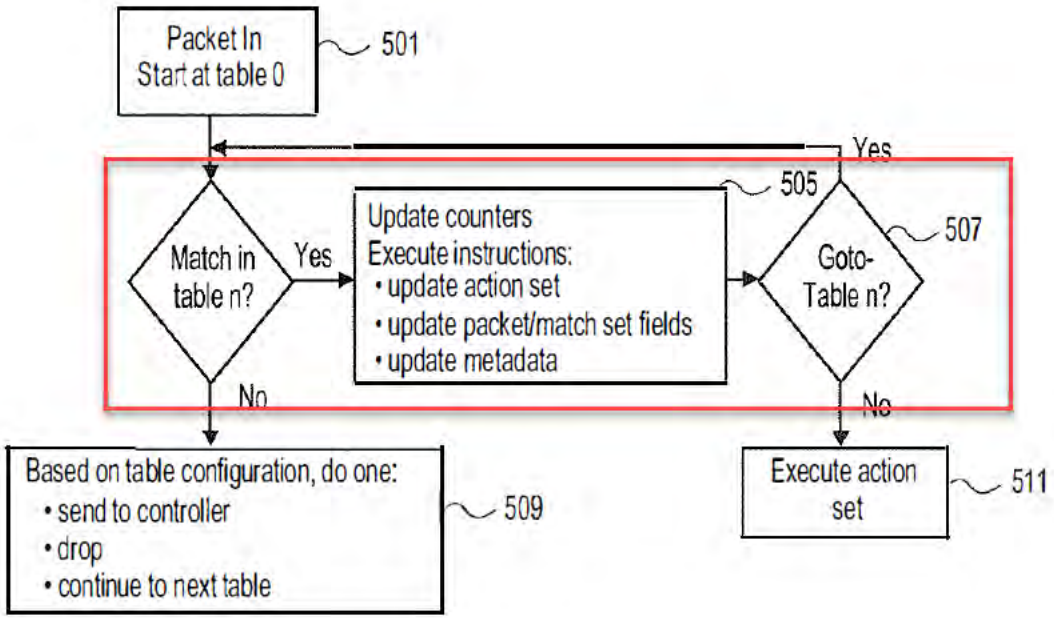
VMW_ORCT_CSCO_128

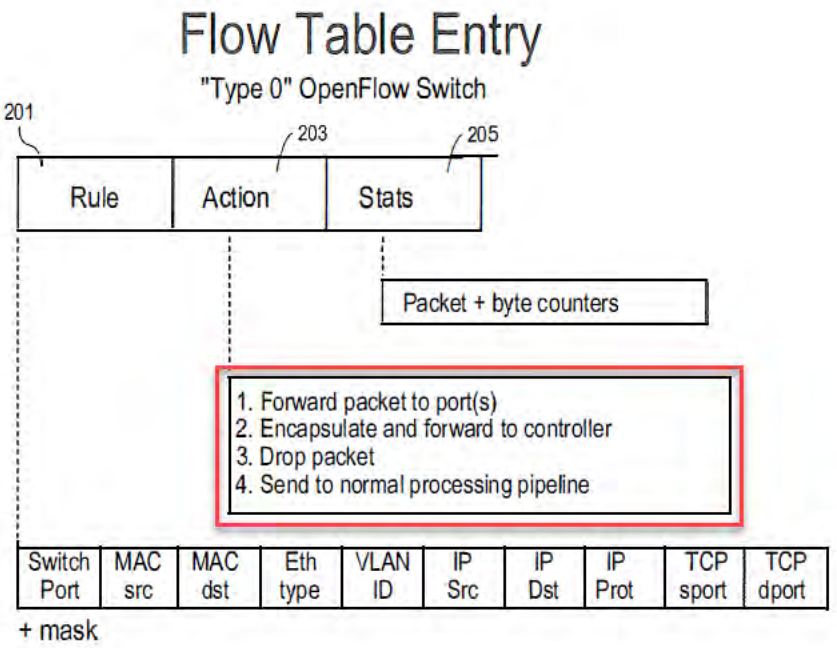
No.	'111 Patent Claim 4	VMware NSX
		Gateways in the L2 Gateway Service are responsible for choosing the active Gateway to forward traffic. The leader election process is designed to quickly fail over to a backup Gateway when the active fails, while preventing multiple Gateways from becoming active concurrently (potentially causing loops). In addition, it is responsible for detecting merges of multiple broadcast domains, or partitions of existing ones. In an L2 Gateway Service, Gateways use a modified version of 802.1ag CFM as leader election probes. Leader election probes are sent on an L2 multicast address so they reach every Gateway in the relevant broadcast domain. The Gateway with the highest priority (lowest numerical CFM ID value) is chosen as the leader and designated the active Gateway for the L2 Gateway Service. Gateways other than the active Gateway do not forward data packets to or from logical networks. Failure detection with the CFM probes occurs within 1-2 seconds. No interaction with the NSX Controller Cluster is required, meaning that failover can occur even when the Gateways cannot reach the NSX Controller. See **L2 Gateway Commands (page 221)** for information on setting failover options for a high availability L2 Gateway Service. VMW_ORCT_CSCO_376 *Verifying Tunnel Status* When implementing Overlay Logical Networks, NSX uses L2-in-L3 tunnels between transport nodes to create logical network topologies that are decoupled from the physical network topology. Problems with tunnel connectivity negatively impact the ability of NSX to create logical networks. It can be compared to having a faulty backplane in a modular switch. To highlight problems with tunnel connectivity, the NSX Controller Cluster monitors the status of all tunnels using active probes. OVS and NSX assess network tunnel health by sending CFM probe packets over the network tunnels that connect transport nodes. Based on the packet loss observed, OVS gives the tunnel a health rating between 0 (unhealthy) and 100 (healthy). You can view tunnel health assessments in NSX Manager (open the transport node whose tunnels you wish to check, open the Tunnels section, and check the Health Status column for each tunnel) and from the OVS command line (use "ovs-appctl show").

No.	'111 Patent Claim 4	VMware NSX
		VMW_ORCT_CSCO_ 4506  Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, 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[b] 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.

No.	'111 Patent Claim 4	VMware NSX
		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.") 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.") 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.") 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.")

No.	'111 Patent Claim 4	VMware NSX
		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.") 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.") Kempf at [0106] ("This encapsulates the packet and sends it to the OpenFlow controller.") Kempf at Figure 5 (annotation added)

No.	'111 Patent Claim 4	VMware NSX
		 FIG. 5 Kempf at Figure 2 (annotation added)

No.	'111 Patent Claim 4	VMware NSX
		 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

No.	'111 Patent Claim 4	VMware NSX
		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.") 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.") 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

No.	'111 Patent Claim 4	VMware NSX
		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.") 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.") 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.") 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

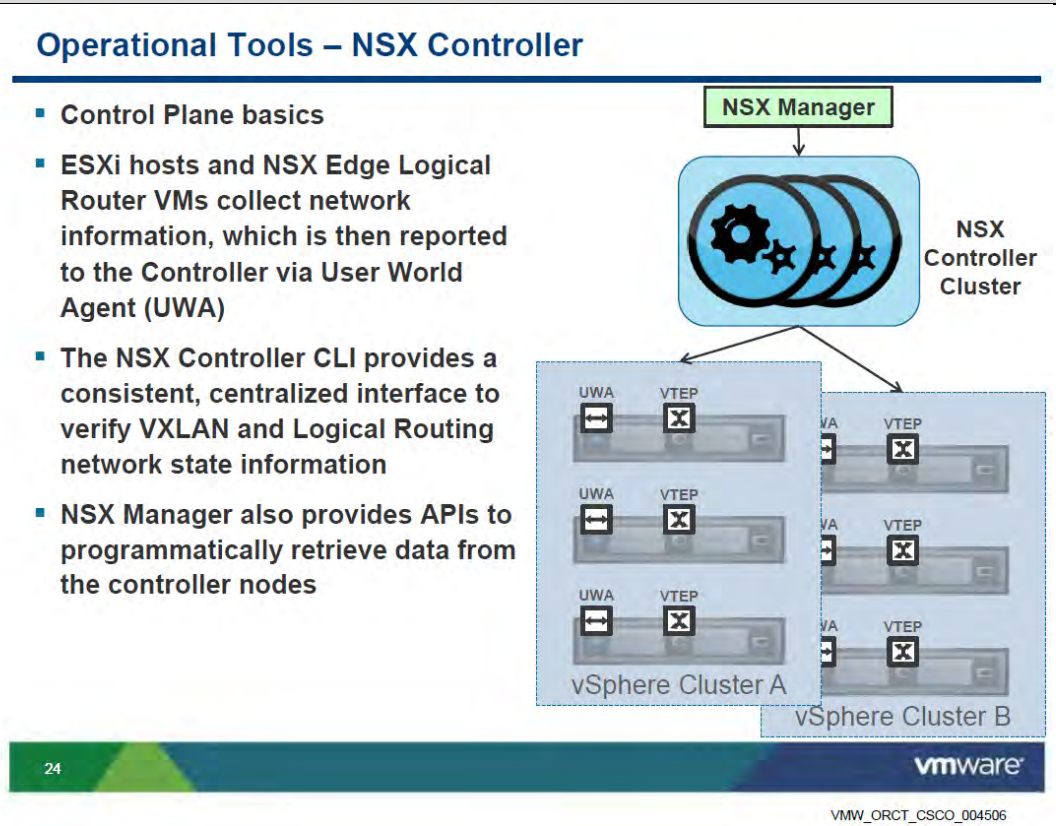
No.	'111 Patent Claim 4	VMware NSX
		portion of network traffic to the network controller 140 when certain conditions are met. Generally, network flows of interest to the network controller 140 contain larger media objects, such as videos and images. In one embodiment, the smaller flows, such as web page and text information, are not exchanged over the steering device interface 316.") 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.") 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.") 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

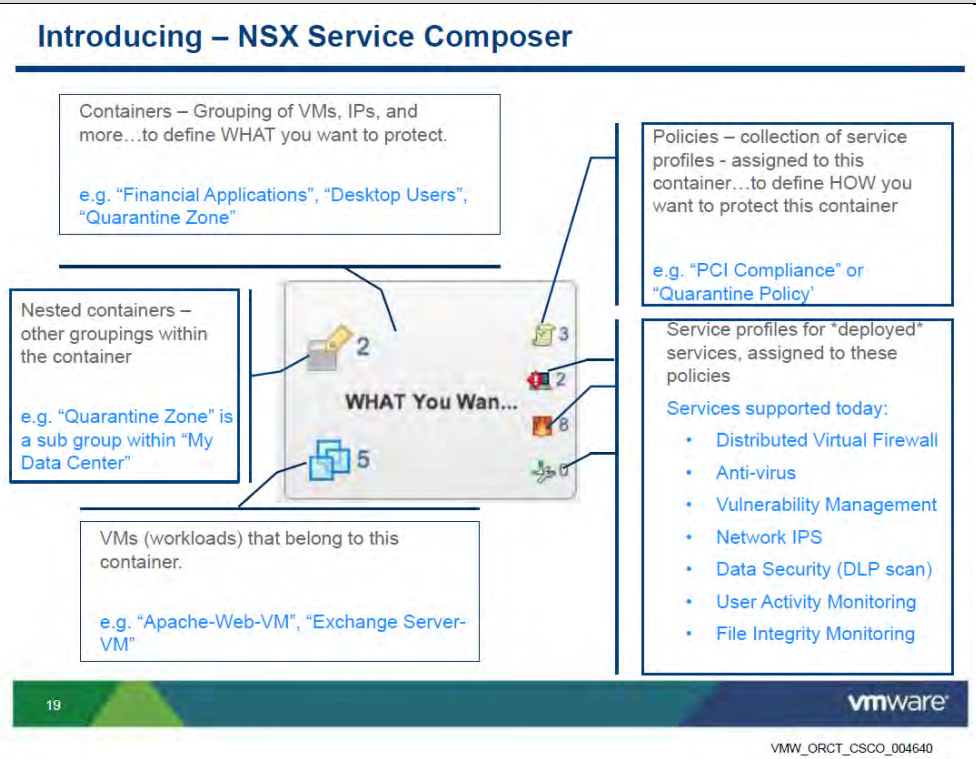
No.	'111 Patent Claim 4	VMware NSX
		HTTP GET request 512 to retrieve content from the origin server 160. The steering device 130 intercepts all requests originated from the user device 110. In one embodiment, the steering device 130 for-wards the HTTP get request 512 to the intended origin server 160 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.") 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 4	VMware NSX
4[c]	responsive to receiving the packet, analyzing the packet, by the controller;	VMware NSX System discloses responsive to receiving the packet, analyzing the packet, by the controller. For example, VMware NSX System discloses analyzing the packet through deep packet firewall inspection by the entity that receives the redirected packet. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the VMware NSX 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. Introducing VMware NSX at 2 **Distributed Firewall:** Stateful firewall capability is built into the hypervisor, delivering distributed, scale-out, high-performance firewall inspection at each virtual switch port, while tracking VM adds, moves and changes. Firewall management is dramatically simplified by enabling rules, audits and monitoring based on virtual infrastructure containers, applications, AD users/identity, and yet richer, using network virtualization and VM introspection. The distributed firewall capability also enables stateful, logical insertion of partner devices/agents e.g. F5, McAfee, Palo Alto Networks, Symantec and Trend. NSX Design Guide at 29

No.	'111 Patent Claim 4	VMware NSX
		Traffic redirection to third party vendor provides the capability to steer a particular type of traffic to a selected partner services VM. For instance, web traffic from the Internet to a set of Apache or Tomcat servers can be redirected to an L4-L7 deep packet inspection firewall for advanced protection. Traffic redirection is defined under Service Composer/Security Policy for NSX version 6.0 or under Partner Security Services tab of the DFW menu in NSX version 6.1. The Partner Security Services tab provides a powerful and easy to use user interface to define what type of traffic needs to be redirected to which partner services. It follows the same policy definition construct as DFW, providing the same options for source field, destination field and services field. The only difference is in the action field; instead of Block/Allow/Reject, a user can select between redirect/no redirect followed by a partner list. Any partner that has been registered with NSX can be successfully deployed on the platform. Additionally, log options can be enabled for this traffic redirection rule. The DFW instance on an ESXi host contains 2 separate tables. The rule table is used to store all policy rules, while the connection tracker table caches flow entries for rules with permit actions. One DFW instance is permitted per VM vNIC. A specific flow is identified by the 5-tuple information consisting source IP address, destination IP address, protocols, L4 source port, and L4 destination port fields. By default, DFW does not perform a lookup on L4 source port, but it can be configured to do so by defining a specific policy rule. VMW_ORCT_CSCO_598

No.	'111 Patent Claim 4	VMware NSX
		Logical Port Mirroring Logical port mirroring allows you to duplicate (or "mirror") all traffic coming into or out of a logical switch port that is attached to a VM VIF or to an L2 Gateway port. The typical reasons you'll use port mirroring are: • Troubleshooting: Analyze, debug, and diagnose errors on a network by inspecting the traffic. • Compliance and monitoring: Forward all monitored traffic to a network appliance for analysis and remediation. The mirrored traffic is sent encapsulated within a GRE Tunnel to a collector so that all the original information of the packets is preserved while traversing the network to a remote location. VMW_ORCT_CSCO_ 629 *Export Event Logs via Syslog* NSX supports exporting event logs to a standard syslog collector for archiving and analysis using third-party tools. NSX Controllers, transport nodes, the NSX Manager appliance, and the NVP OVS vApp can export event logs in this way. • To configure syslog export of Controller event logs, see Syslog Export of NSX Controller Event and System Logs, below. • To configure syslog export of event logs from other types of NSX appliances, see Syslog Export of NSX Appliance Event Logs, below. VMW_ORCT_CSCO_ 4506

No.	'111 Patent Claim 4	VMware NSX
		 VMW_ORCT_CSCO_4640

No.	'111 Patent Claim 4	VMware NSX
		 Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, 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(c) of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references. For example, Swenson discloses the network controller comprising a flow analyzer for analyzing and inspecting the packet.

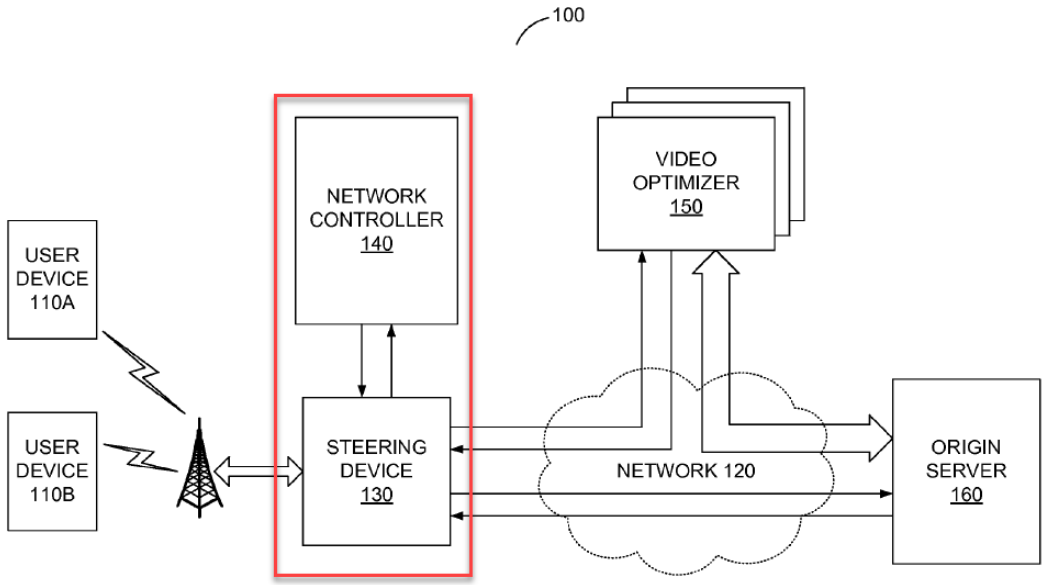
No.	'111 Patent Claim 4	VMware NSX
		Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the 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.") 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.") 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

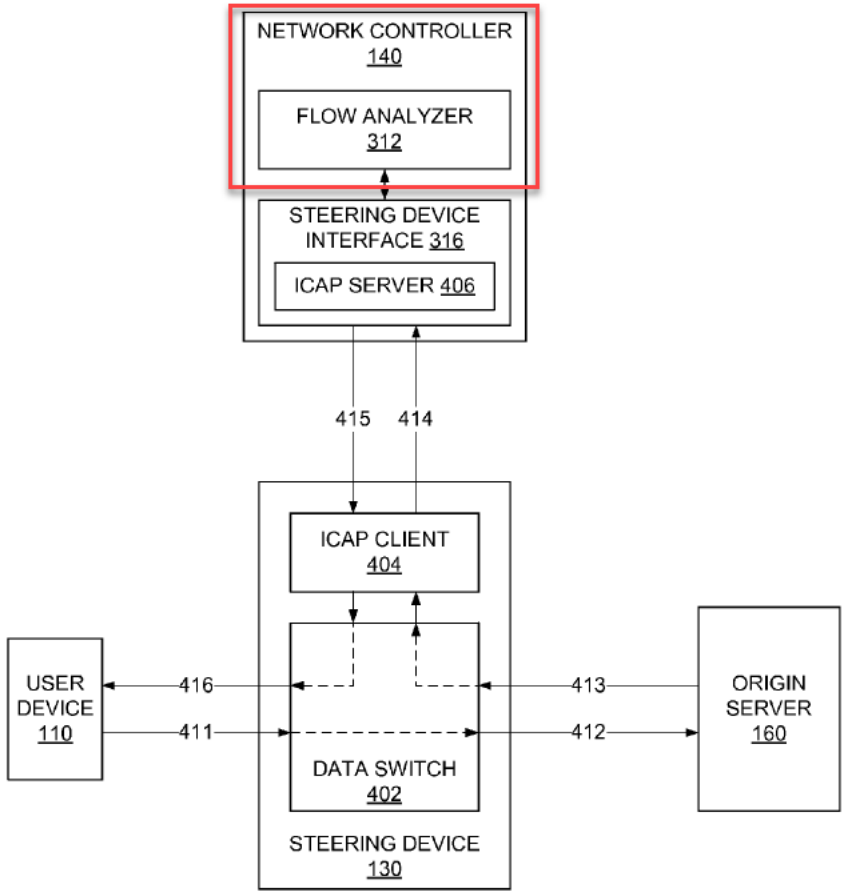
No.	'111 Patent Claim 4	VMware NSX
		devices attached, rout-ers through which the traffic is passing, geography regions, the backhaul segments, and time-of-day of the flows. For example, in a mobile network, the cell tower to which a user device is attached can be most useful, since it is the location 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.") 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.") 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.") 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

No.	'111 Patent Claim 4	VMware NSX
		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.") 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.") 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

No.	'111 Patent Claim 4	VMware NSX
		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.") 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.") 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

No.	'111 Patent Claim 4	VMware NSX
		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.") 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.") Swenson at Figure 1 (annotation added)

No.	'111 Patent Claim 4	VMware NSX
		 FIG. 1 Swenson at Figure 4A (annotation added)

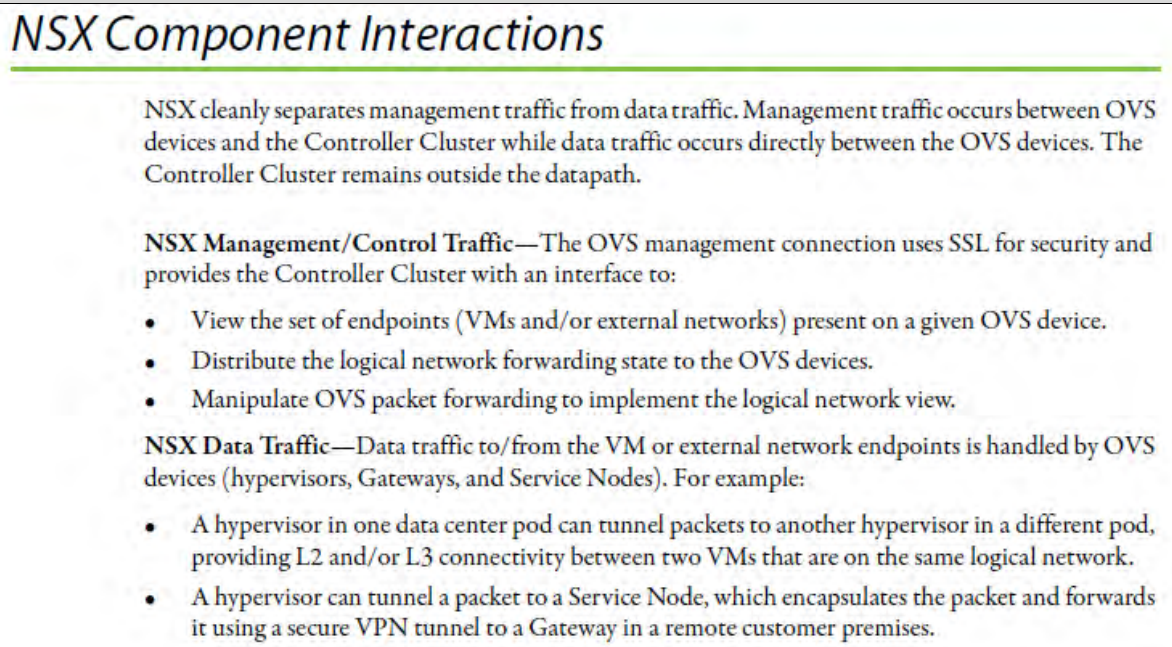
No.	'111 Patent Claim 4	VMware NSX
		 FIG. 4A

No.	'111 Patent Claim 4	VMware NSX
		For example, Copeland discloses analyzing packets received by the intrusion detection engine on the monitoring appliance. 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.") 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.") 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.")

No.	'111 Patent Claim 4	VMware NSX
		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 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.") 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.") 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.") 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

No.	'111 Patent Claim 4	VMware NSX
		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.) 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.") 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.") 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.") 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

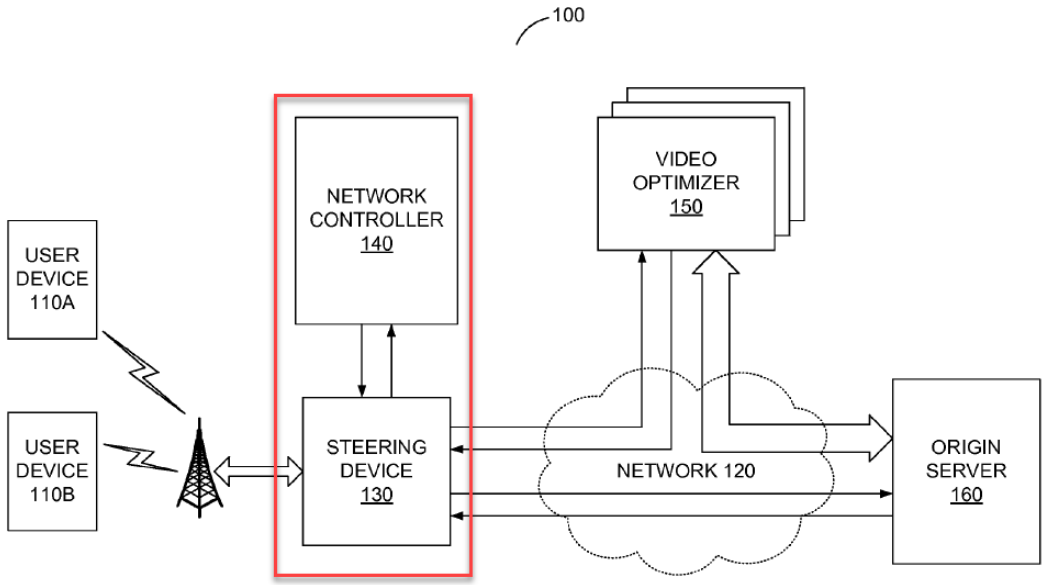
No.	'111 Patent Claim 4	VMware NSX
		differing flow characteristics. Consequently, the flow engine 155 analyzes each of the services separately.") Copeland at [0150] ("A preferred hardware configuration 800 of an embodiment that executes the functions of the above-described flow-based engine is described 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	On information and belief, the VMware NSX System discloses sending the packet, by the controller, to the network node. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the VMware NSX 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. VMW_ORCT_CSCO_58

No.	'111 Patent Claim 4	VMware NSX
		 Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, 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(d) of Exhibit E-4 renders the claim, including the present limitation, obvious. Below is an example. 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 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

No.	'111 Patent Claim 4	VMware NSX
		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.") 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.") 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

No.	'111 Patent Claim 4	VMware NSX
		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.") 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.") 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

No.	'111 Patent Claim 4	VMware NSX
		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.") 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.") 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.") Swenson at Figure 1 (annotation added)

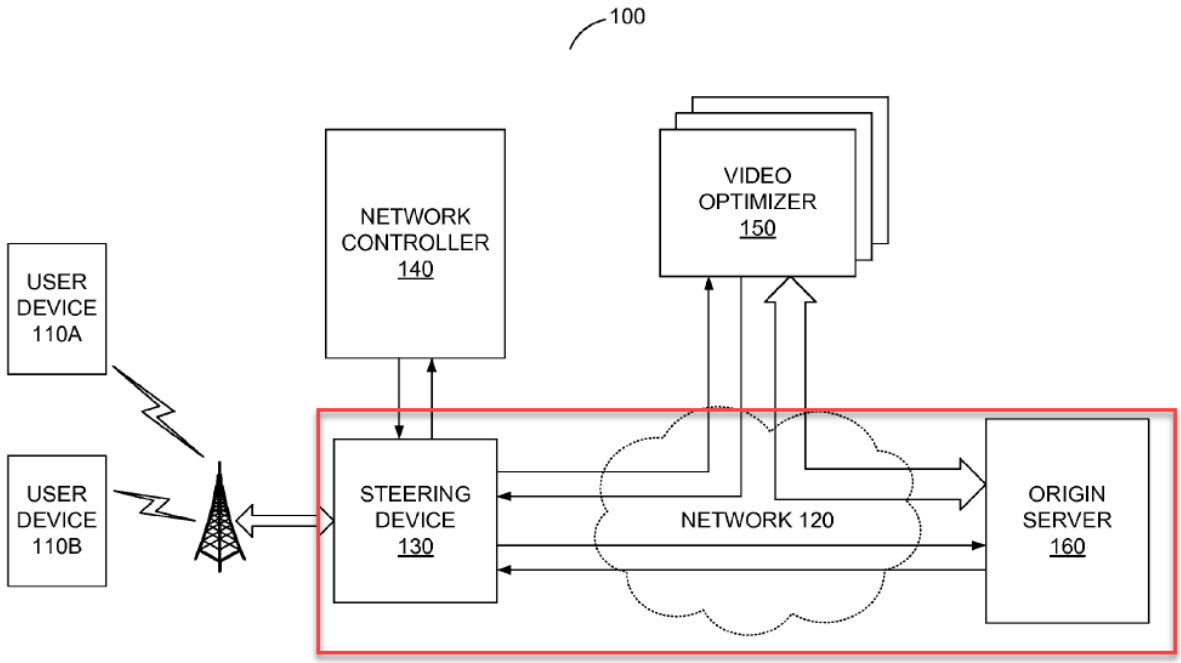
No.	'111 Patent Claim 4	VMware NSX
		 FIG. 1
4[e]	responsive to receiving the packet, sending the packet, by the network node, to the second entity.	On information and belief, VMware NSX System discloses responsive to receiving the packet, sending the packet, by the network node, to the second entity. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the VMware NSX System is found to not meet this limitation, responsive to receiving the packet, sending the packet, by the network node, to the second entity would have been obvious to a person having ordinary skill in the art, as explained below. 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

No.	'111 Patent Claim 4	VMware NSX
		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. 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. 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.") 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

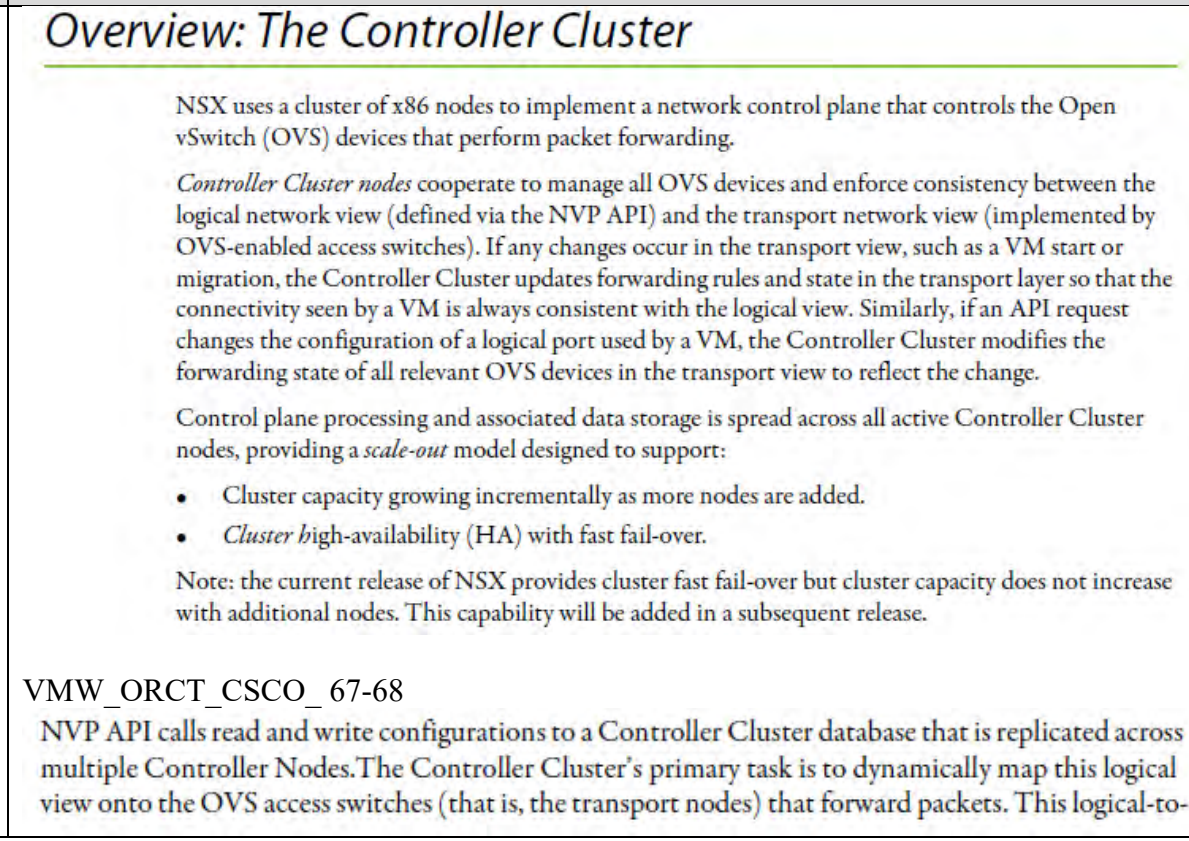
No.	'111 Patent Claim 4	VMware NSX
		controller 140 may determine that existing flows associated with a particular subscriber or other entity should be optimized. In turn, new flows (e.g., resulting from seek requests in media, new media requests, resume after pause, etc.) determined to be associated 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.") 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.") 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

No.	'111 Patent Claim 4	VMware NSX
		controllers) may be scaled independent from the steering devices handling raw HTTP throughput.") 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.") 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.") 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,

No.	’111 Patent Claim 4	VMware NSX
		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.”) Swenson at Figure 1 (annotation added)

No.	'111 Patent Claim 4	VMware NSX
		 FIG. 1

No.	'111 Patent Claim 5	VMware NSX
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.	On information and belief, VMware NSX 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. On information and belief, VMware NSX System discloses responsive to receiving the packet, sending the packet, by the network node, 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 VMware NSX System is found to not meet this 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. VMW_ORCT_CSCO_ 58 *NSX Component Interactions* NSX cleanly separates management traffic from data traffic. Management traffic occurs between OVS devices and the Controller Cluster while data traffic occurs directly between the OVS devices. The Controller Cluster remains outside the datapath. **NSX Management/Control Traffic**—The OVS management connection uses SSL for security and provides the Controller Cluster with an interface to: • View the set of endpoints (VMs and/or external networks) present on a given OVS device. • Distribute the logical network forwarding state to the OVS devices. • Manipulate OVS packet forwarding to implement the logical network view. **NSX Data Traffic**—Data traffic to/from the VM or external network endpoints is handled by OVS devices (hypervisors, Gateways, and Service Nodes). For example: • A hypervisor in one data center pod can tunnel packets to another hypervisor in a different pod, providing L2 and/or L3 connectivity between two VMs that are on the same logical network. • A hypervisor can tunnel a packet to a Service Node, which encapsulates the packet and forwards it using a secure VPN tunnel to a Gateway in a remote customer premises. VMW_ORCT_CSCO_ 63

No.	'111 Patent Claim 5	VMware NSX
		

Overview: The Controller Cluster

NSX uses a cluster of x86 nodes to implement a network control plane that controls the Open vSwitch (OVS) devices that perform packet forwarding.

Controller Cluster nodes cooperate to manage all OVS devices and enforce consistency between the logical network view (defined via the NVP API) and the transport network view (implemented by OVS-enabled access switches). If any changes occur in the transport view, such as a VM start or migration, the Controller Cluster updates forwarding rules and state in the transport layer so that the connectivity seen by a VM is always consistent with the logical view. Similarly, if an API request changes the configuration of a logical port used by a VM, the Controller Cluster modifies the forwarding state of all relevant OVS devices in the transport view to reflect the change.

Control plane processing and associated data storage is spread across all active Controller Cluster nodes, providing a *scale-out* model designed to support:

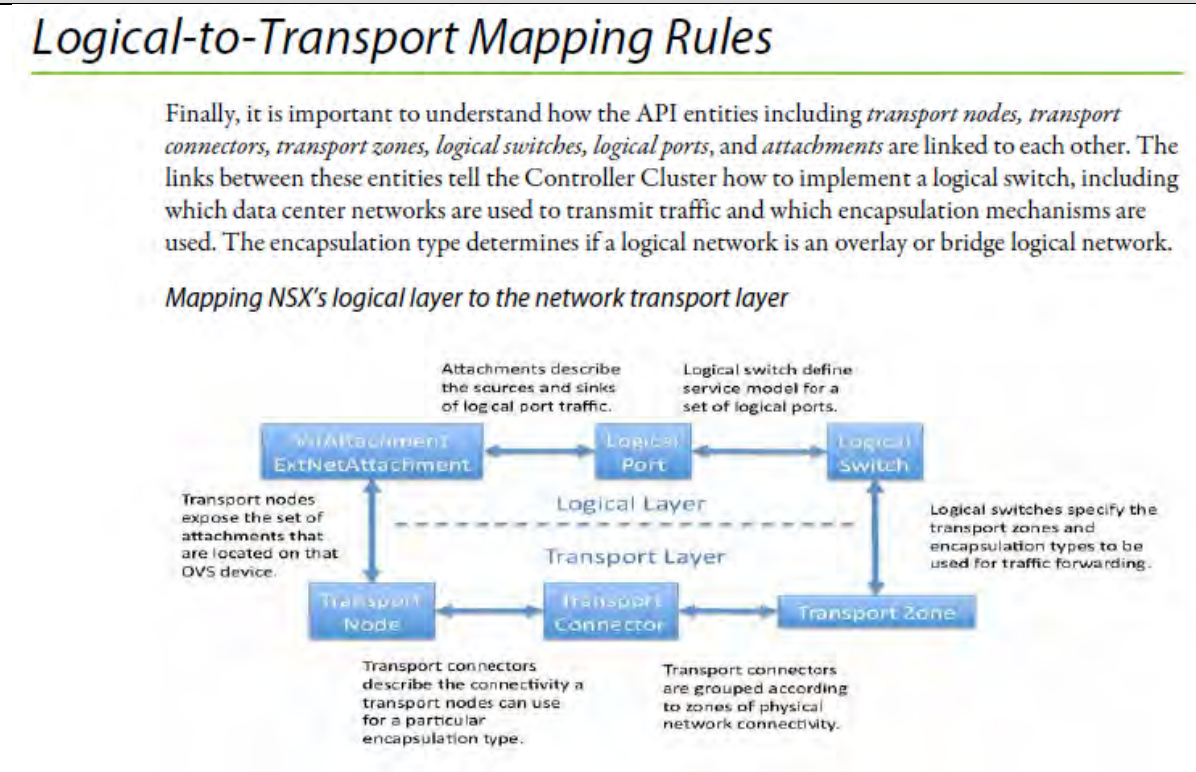
- Cluster capacity growing incrementally as more nodes are added.
- *Cluster h*igh-availability (HA) with fast fail-over.

Note: the current release of NSX provides cluster fast fail-over but cluster capacity does not increase with additional nodes. This capability will be added in a subsequent release.

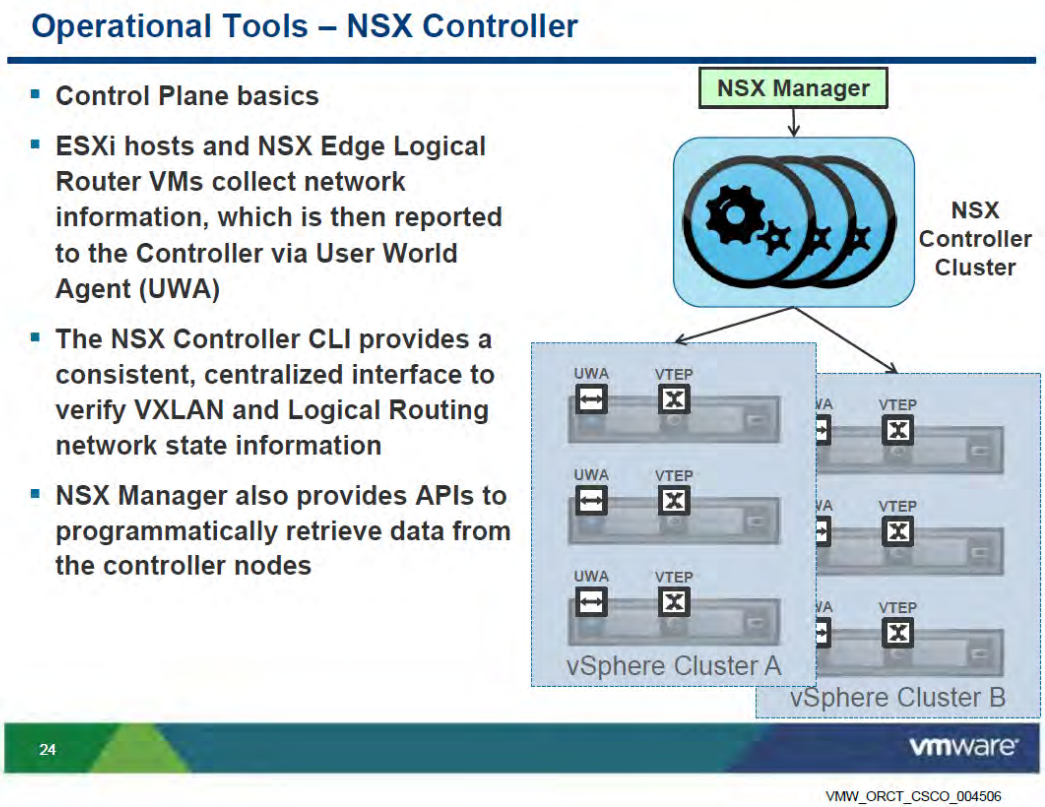
VMW_ORCT_CSCO_ 67-68

NVP API calls read and write configurations to a Controller Cluster database that is replicated across multiple Controller Nodes.The Controller Cluster's primary task is to dynamically map this logical view onto the OVS access switches (that is, the transport nodes) that forward packets. This logical-to-

No.	'111 Patent Claim 5	VMware NSX
		transport mapping is a continuous process since the Controller Cluster must react to changes in the transport view (such as a VM migration) or the logical view (such as a change to a VM's security policy). As a result, the API exposes status indicators (port status and fabric status) to show whether a logical port is currently "wired" in the transport view. The definition of a logical switch via the API includes instructions that tell the Controller Cluster how to use transport view resources to implement logical forwarding. These instructions include: • For those hypervisors and Gateways that have multiple NICs: Which NIC should be used when sending encapsulated traffic from that logical switch? This is expressed by associating a logical switch with one or more transport zones. • Which encapsulation type should be used when transmitting logical switch data across the physical network? See **Logical Switch or Logical Router Packet Forwarding Pipeline (page 44)** for descriptions of how the encapsulation mechanism determines whether a logical switch operates in *overlay* or *bridge* mode. VMW_ORCT_CSCO_ 71 *Transport Zones* Because any transport node might have multiple interfaces, the Controller Cluster must understand the topology of the data center network. This topology is expressed in NSX using *transport zones*. Each transport connector specifies the transport zone used to carry traffic for that connector. The transport zone allows the Controller Cluster to understand what transport connectors can communicate directly when implementing a logical switch. For example, with an overlay logical network, the Controller Cluster must know which hypervisor IP addresses can be used to create direct tunnel connections. VMW_ORCT_CSCO_ 73

No.	'111 Patent Claim 5	VMware NSX
		

VMW_ORCT_CSCO_4506

No.	'111 Patent Claim 5	VMware NSX
		 *See supra* at Claim 1. Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, 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 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.

No.	'111 Patent Claim 5	VMware NSX
		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.") 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.") 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.") 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

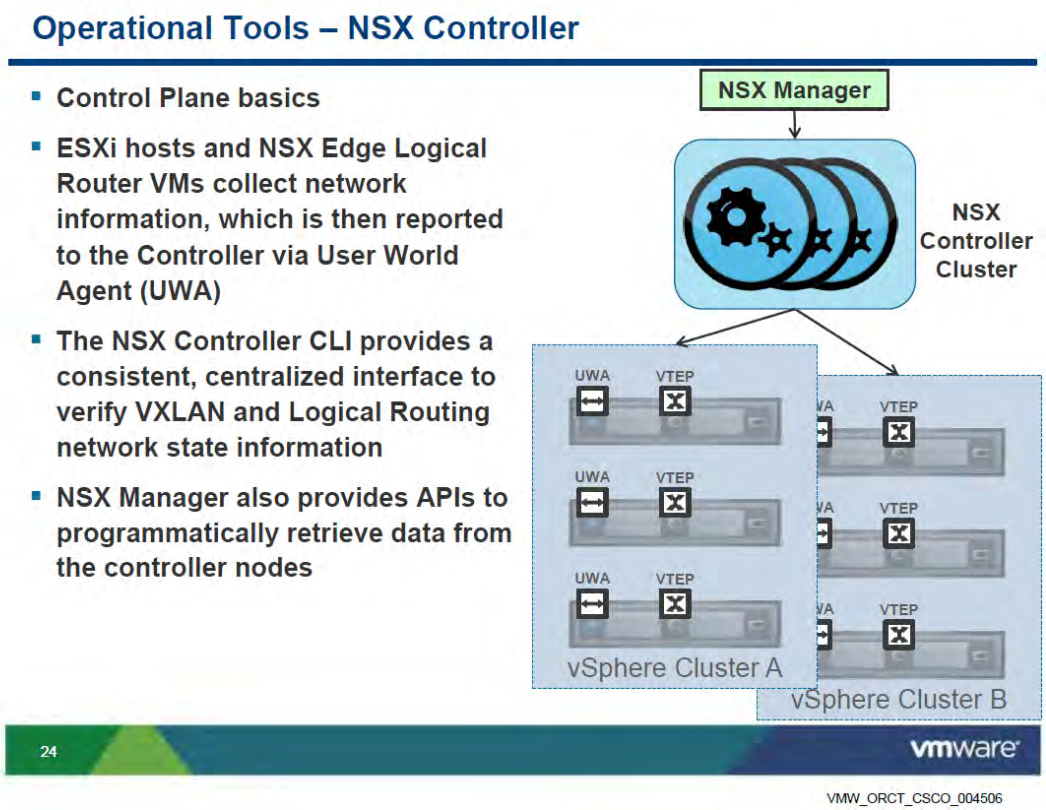
No.	'111 Patent Claim 5	VMware NSX
		purposes in the flow data. Other infor-mation, including the sampling device and the physical port on which the communications was detected may also be retained for other uses such as mitigation, where a host may be removed from the network.") Copeland at [0126]-[0129] ("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: (1) both port numbers match and no port is marked as the "server" port, or (2) the port number previously marked as the "server" port matches, or (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).") 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."") 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

No.	'111 Patent Claim 5	VMware NSX
		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.") 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: (a). both port numbers match and no port is marked as the "server" port, or (b). the port number previously marked as the "server" port matches, or (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	VMware NSX
6	The method according to claim 5, further comprising storing the received packet or a portion thereof, by the controller, in a memory.	On information and belief, VMware NSX System discloses the method according to claim 5, further comprising storing the received packet or a portion thereof, by the controller, in a memory. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the VMware NSX System 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. *See supra* at Claim 5. VMW_ORCT_CSCO_ 395

No.	'111 Patent Claim 6	VMware NSX
		Out of disk space The Controller may be out of disk space. Use the "show status" command to see if any of the partitions have 0 bytes available. The NSX CLI command show system statistics can be used to display resource utilization for disk space, disk I/O, memory, CPU and various other processes on the Controller Nodes. The command offers statistics with one-minute intervals for a window of one hour for various combinations. The show system statistics CLI command does auto-completion and can be used to view the list of metric data available. VMW_ORCT_CSCO_ 540 *Controller Cluster and Service Node Summaries* The NSX Manager "Dashboard" tab provides a summarized view of the NSX Controller Cluster nodes and Service Nodes, identifying them by name and IP address while showing the Status, System Load and System Memory consumption for each one. VMW_ORCT_CSCO_. 1259

No.	'111 Patent Claim 6	VMware NSX
		Check NSX Controller health from NSX Manager To check NSX Controller health from the UI, open the **Controller Cluster** menu and click its heading, **Controller Cluster**.  In the **Controller Cluster Nodes** table, for each Controller in the cluster you will see the name and IP address, Up/Down status, current Log Level setting, installed NSX software version, current system CPU load, current memory usage, and the **File System Usage** showing the amount of free disk space left on each major partition. By default, when an NSX Controller or other NSX appliance hits 90% full disk, it deletes down to a target of 70% full. The system also monitors other statistics to ensure that a safe amount of free space is always kept available on the disk. Users cannot change default cleanup thresholds. VMW_ORCT_CSCO_4506

No.	'111 Patent Claim 6	VMware NSX
		 VMW_ORCT_CSCO_004506 Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, 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 6 of Exhibit E-4 renders the claim, including the present limitation, obvious. Below is an example. For example, Swenson discloses the network controller storing historical network traffic data based on received packet flows. 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

No.	'111 Patent Claim 6	VMware NSX
		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.") 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.") 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

No.	'111 Patent Claim 6	VMware NSX
		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.") 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.") 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.") 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.

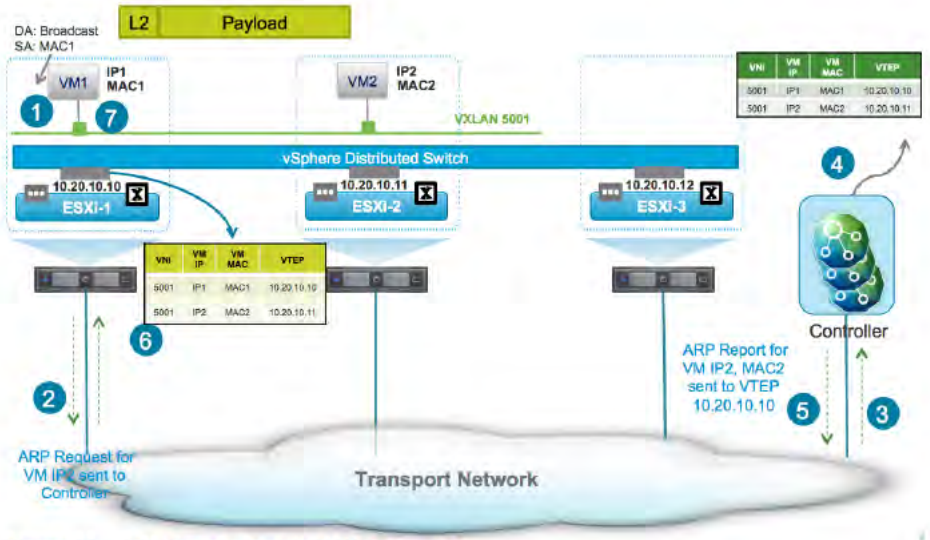
No.	'111 Patent Claim 6	VMware NSX
		When a match-ing flow is detected, the steering device 130 forwards the HTTP request and a portion of the HTTP response to the network controller 140 over the ICAP client interface 404. After receiving the request and the portion of response at the ICAP server interface 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.") 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.") For example, Chua '877 discloses logging and storing the packets, representative data of the packets, paths, and programs.

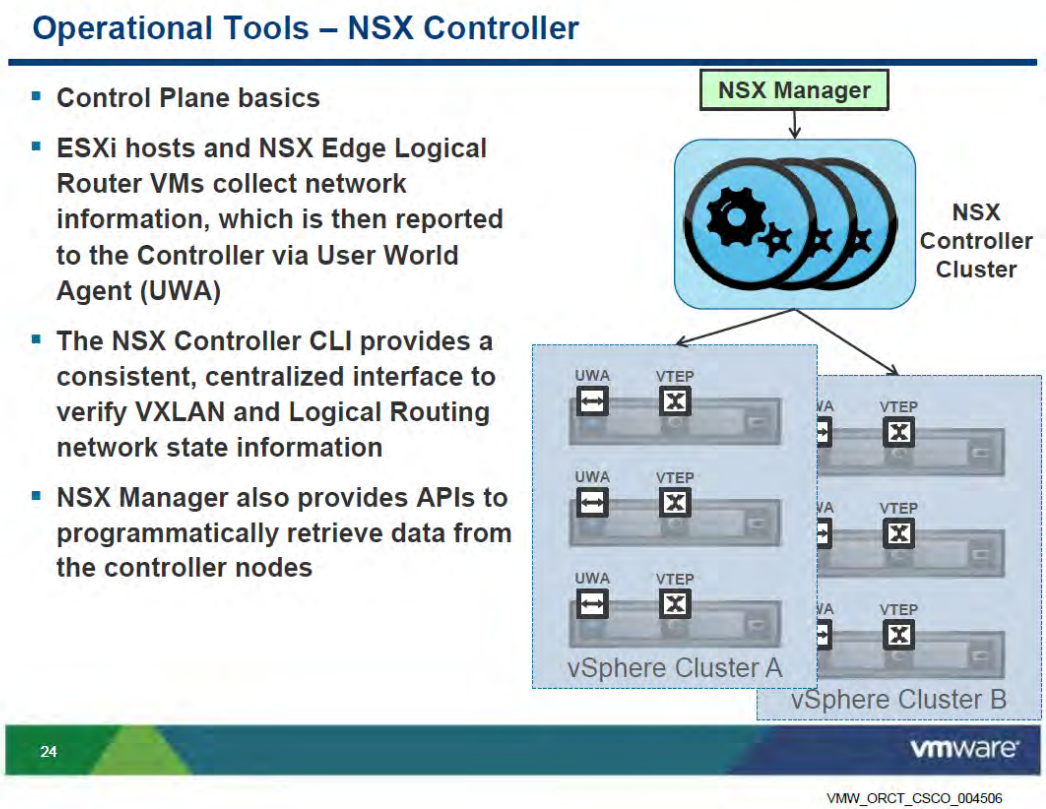
No.	'111 Patent Claim 6	VMware NSX
		Chua '877 at 7:28-54 ("SDN controller 112 may receive data as input from service devices 116. For example, SDN controller 112 may be con-figured to receive data from an intrusion detection system (IDS) device, a Denial of Service (DoS) device, a Distributed Denial 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: Allow-explicitly allow a certain network flow to proceed to its destination Block-explicitly block a certain flow from traversing SDN 106 Mirror-allow the traffic, but send a copy of the traffic for deeper inspection or recording to, e.g., one of service devices 116 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. Transform-modify or translate values of headers of packets in the network flow Encapsulate-encapsulate packets in the network flow with a particular header") 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.") 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

No.	'111 Patent Claim 6	VMware NSX
		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.") 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 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	VMware NSX
7	The method according to claim 5, further comprising responsive to the packet satisfying the criterion	VMware NSX 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.

No.	'111 Patent Claim 7	VMware NSX
	and to the instruction, sending a portion of the packet, by the network node, to the controller.	For example, VMware NSX System discloses sending by the ESXi host a control plane request based on information from a received packet to the NSX controller. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the VMware NSX System 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. *See supra* at Claim 5. NSX Design Guide at 44

No.	'111 Patent Claim 7	VMware NSX
		 Figure 35 – ARP Resolution via NSX Controller 1. VM1 generates an ARP request (e.g., L2 broadcast packet) to determine the MAC/IP mapping information for VM2. 2. ESXi-1 intercepts the request and generates a control plane request to the controller asking for the MAC/IP mapping information. 3. The controller receives the control plane request. 4. The controller checks its local ARP table for the required mapping information. 5. The mapping information is sent to ESXi-1 with a control plane ARP report message. 6. ESXi-1 receives the control plane message and updates its local table with the mapping information. At this point the VTEP where VM2 is connected is known (10.20.10.11). 7. ESXi-1 generates an ARP response on behalf of VM2 – the source MAC address in the frame is MAC2 – and delivers it to VM1. This proxy process is invisible to VM1; it simply views the reply as having come directly from VM2. VMW_ORCT_CSCO_4506

No.	'111 Patent Claim 7	VMware NSX
		 Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, 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 7 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.

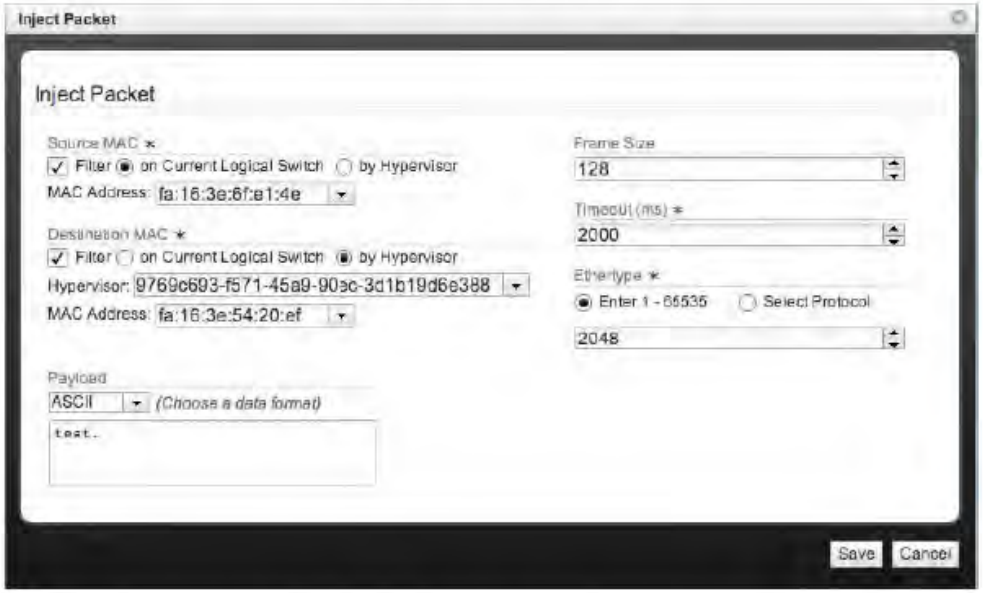
No.	'111 Patent Claim 7	VMware NSX
		Copeland at [0067] ("The host servers 130 are directly or indirectly coupled to one or more network devices 135 such as routers or switches that support providing a sampled data stream such as that provided by sFlow. In a typical preferred configuration for the present 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.") 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.") 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.") 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

No.	'111 Patent Claim 7	VMware NSX
		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.") 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: (1) both port numbers match and no port is marked as the "server" port, or (2) the port number previously marked as the "server" port matches, or (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).") 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."") 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

No.	'111 Patent Claim 7	VMware NSX
		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.") 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: (a). both port numbers match and no port is marked as the "server" port, or (b). the port number previously marked as the "server" port matches, or (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	VMware NSX
8[a]	The method according to claim 7, wherein the portion of the packet consists of multiple consecutive bytes, and	VMware NSX System discloses the method according to claim 7, wherein the portion of the packet consists of multiple consecutive bytes. For example, VMware NSX System discloses packet encapsulations consisting of multiple bytes. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the VMware NSX 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. *See supra* at Claim 7. NSX Design Guide at 20

No.	'111 Patent Claim 8	VMware NSX
		• The source and destination IP addresses used in the external IP header uniquely identify the ESXi hosts originating and terminating the VXLAN encapsulation of frames. Those are usually referred to as VXLAN Tunnel Endpoints (VTEPs). • Encapsulating the original Ethernet frame into a UDP packet increases the size of the IP packet. For this reason, increasing the MTU to a minimum of 1600 bytes is recommended for all interfaces in the physical infrastructure that will carry the frame. The MTU for the VDS uplinks of the ESXi hosts performing VXLAN encapsulation is automatically increased when preparing the host for VXLAN (from the NSX Manager UI). NSX Design Guide at 102 **Jumbo MTU** When leveraging encapsulation technologies, it is important to increase the MTU supported both on ESXi hosts as well as on all interfaces of the devices deployed in the physical network. VXLAN adds 50 Bytes to each original frame, leading to the recommendation to increase the MTU size to at least 1600 bytes. The MTU on the ESXi server, both for the internal logical switch and for the VDS uplinks, is automatically tuned to 1600 bytes when provisioning VXLAN to hosts from the NSX Manager UI. VMW_ORCT_CSCO_799-800 *Using Traceflow in NSX Manager's Logical Port Inspector* To use Traceflow in NSX Manager: 1. Open the Logical Switch Port inspector page for the port where the test packet will originate. (Network Components: Logical Switch Ports) 2. In the Traceflow section of the page, click the **Inject** button.

No.	'111 Patent Claim 8	VMware NSX
		3. In the Inject Packet window, choose the **Source MAC** and **Destination MAC** addresses. To specify a port, provide its MAC address in one of the following ways: – Uncheck the **Filter** option and type the MAC address (see MAC address input formats (page 76). – Check the Filter option and choose on **Current Logical Switch** to see the list of local ports, or choose **by Hypervisor** and specify a hypervisor whose ports you want to choose from. Select the MAC address. You can further filter the list by typing a partial, *colon-separated* MAC address and pressing the up/down keys on your keyboard.  4. Optionally, you may provide a **Payload** for the packet. If you do not provide a payload, Traceflow inserts a default payload. Traceflow test packets always include the Ethernet header that Traceflow constructs based on your input. Using the Payload field, you can add other headers such as such as IP headers, TCP headers, and so on, as well as payload data.

No.	'111 Patent Claim 8	VMware NSX
		5. Choose the **Frame Size**, taking care to leave sufficient space for all headers and the payload you are sending. (Traceflow appends a few bytes of metadata at the end of the payload.) 6. Specify the **Timeout** in milliseconds (min. 1000 ms; max. 10000 ms). This specifies how long Traceflow will wait, in total, to observe whether the packet has been delivered to all destinations in the NSX network. 7. Click **Save** to run the test. Results are returned only upon completion of the timeout period, and not immediately upon each delivery as would be the case with a tool such as ping. Traceflow reports its observations as of the end of the timeout period, and no subsequent observations will be returned. VMW_ORCT_CSCO_ 1025 *View logical port statistics in NSX Manager* To retrieve logical port counters: 1. In the NSX Manager, click on the **Network Components** tab, and **Logical Switch Ports** or **Logical Router Ports**. 2. Click on the **Name** (UUID) of the port you want to inspect. This opens the logical port inspection window. In the **Statistics** table, you will see the following statistics: – **Rx Bytes**: Number of Received Bytes – **Rx Errors**: Number of received packets with error (alignment, CRC). Note that usually for Virtual Interface attached to a virtual machine this number stays at 0 since there are no physical media adding constraints on the information. – **Rx Packets**: Number of Packets received – **Tx Bytes**: Number of Transmit Bytes – **Tx Errors**: Number of transmitted packets with error (alignment, CRC). Note that usually for Virtual Interface attached to a virtual machine this number stays at 0 since there are no physical media adding constraints on the information. – **Tx Packets**: Number of Packet Transmitted

No.	'111 Patent Claim 8	VMware NSX
		VMW_ORCT_CSCO_ 1030 *Retrieve per-Collector per transport node mirrored-packet statistics* NSX allows you to review how many packets have been copied from a specific transport node to a given Collector. To review these statistics: 1. Click on the API Navigator 2. Search for the method GET /ws.v1/transport-nodes/<UUID>/mirror/<collector-ip>/ statistics This will return: • TX Byte: Number of bytes sent to the Collector from the transport node • TX Packets: Number of Packets sent to the Collector from the transport node Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, 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 8(a) of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references. For example, Kempf discloses consecutive bytes of a packet header field. 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

No.	'111 Patent Claim 8	VMware NSX
		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.") 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.") 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.") 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

No.	'111 Patent Claim 8	VMware NSX
		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.") 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.") 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.") 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

No.	'111 Patent Claim 8	VMware NSX
		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: Write-Metadata (GTP-TEID, 0x FFFFFFFF) Apply-Actions (Set-Output-Port LOCAL_GTP _U_ENCAP)") 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. 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.	VMware NSX System discloses wherein the instruction comprises identification of the consecutive bytes in the packet. For example, VMware NSX System discloses identifying the multiple bytes of the packet encapsulation. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the VMware NSX 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. NSX Design Guide at 20

No.	'111 Patent Claim 8	VMware NSX
		• The source and destination IP addresses used in the external IP header uniquely identify the ESXi hosts originating and terminating the VXLAN encapsulation of frames. Those are usually referred to as VXLAN Tunnel Endpoints (VTEPs). • Encapsulating the original Ethernet frame into a UDP packet increases the size of the IP packet. For this reason, increasing the MTU to a minimum of 1600 bytes is recommended for all interfaces in the physical infrastructure that will carry the frame. The MTU for the VDS uplinks of the ESXi hosts performing VXLAN encapsulation is automatically increased when preparing the host for VXLAN (from the NSX Manager UI). NSX Design Guide at 102 **Jumbo MTU** When leveraging encapsulation technologies, it is important to increase the MTU supported both on ESXi hosts as well as on all interfaces of the devices deployed in the physical network. VXLAN adds 50 Bytes to each original frame, leading to the recommendation to increase the MTU size to at least 1600 bytes. The MTU on the ESXi server, both for the internal logical switch and for the VDS uplinks, is automatically tuned to 1600 bytes when provisioning VXLAN to hosts from the NSX Manager UI. VMW_ORCT_CSCO_1025

No.	'111 Patent Claim 8	VMware NSX
		View logical port statistics in NSX Manager To retrieve logical port counters: 1. In the NSX Manager, click on the **Network Components** tab, and **Logical Switch Ports** or **Logical Router Ports**. 2. Click on the **Name** (UUID) of the port you want to inspect. This opens the logical port inspection window. In the **Statistics** table, you will see the following statistics: – **Rx Bytes**: Number of Received Bytes – **Rx Errors**: Number of received packets with error (alignment, CRC). Note that usually for Virtual Interface attached to a virtual machine this number stays at 0 since there are no physical media adding constraints on the information. – **Rx Packets**: Number of Packets received – **Tx Bytes**: Number of Transmit Bytes – **Tx Errors**: Number of transmitted packets with error (alignment, CRC). Note that usually for Virtual Interface attached to a virtual machine this number stays at 0 since there are no physical media adding constraints on the information. – **Tx Packets**: Number of Packet Transmitted Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, 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 8(b) of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references. For example, Kempf discloses rules in which the flow table includes matching to the consecutive bytes of a packet header. 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

No.	'111 Patent Claim 8	VMware NSX
		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.") 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.") 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

No.	'111 Patent Claim 8	VMware NSX
		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.") 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.") 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.") 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.")

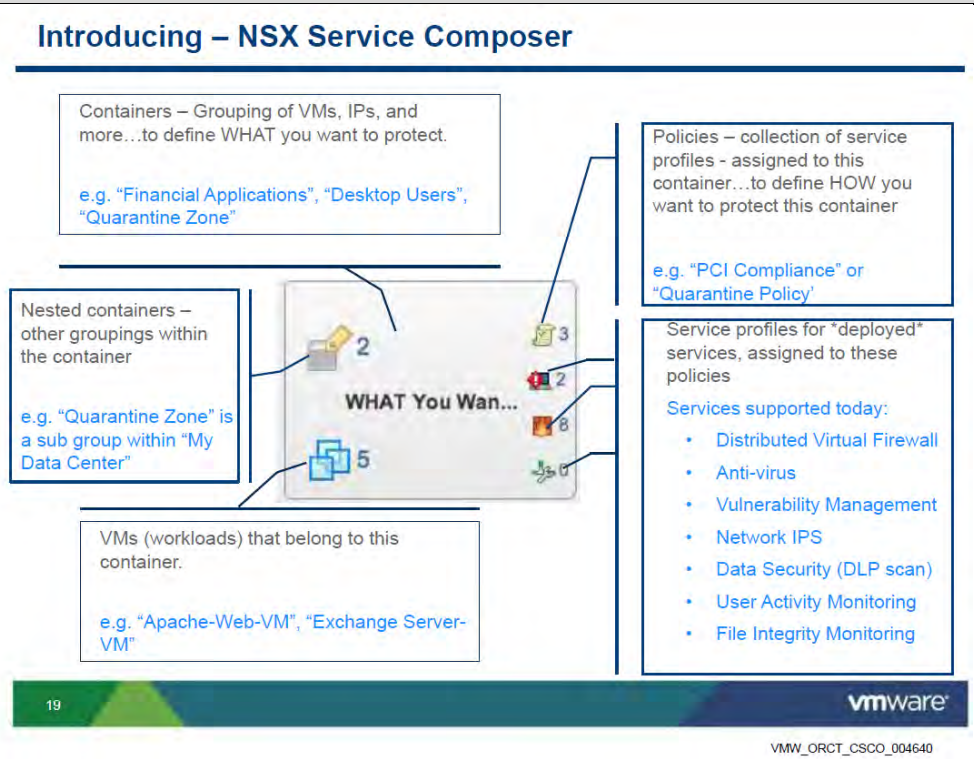
No.	'111 Patent Claim 8	VMware NSX
		Kempf at [0101] ("In one embodiment, the system implements han-dling of user data plane packets requiring GTP-U encapsula-tion with extension headers, sequence numbers, and N-PDU numbers. User data plane packets that require extension head-ers, sequence numbers, or 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: Write-Metadata (GTP-TEID, 0x FFFFFFFF) Apply-Actions (Set-Output-Port LOCAL_GTP _U_ENCAP)") For example, Copeland discloses identifying the sampled packet headers comprised of fragmented packets of smaller byte sizes. 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.") 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.") 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

No.	'111 Patent Claim 8	VMware NSX
		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	VMware NSX
9	The method according to claim 5, further comprising responsive to receiving the packet, analyzing the packet, by the controller.	VMware NSX System discloses the method according to claim 5, further comprising responsive to receiving the packet, analyzing the packet, by the controller. For example, VMware NSX System discloses analyzing the packet through deep packet firewall inspection by the entity that receives the redirected packet. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the VMware NSX System is found to not meet this limitation, further comprising 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. *See supra* at Claim 5. NSX Design Guide at 29

No.	'111 Patent Claim 9	VMware NSX
		Traffic redirection to third party vendor provides the capability to steer a particular type of traffic to a selected partner services VM. For instance, web traffic from the Internet to a set of Apache or Tomcat servers can be redirected to an L4-L7 deep packet inspection firewall for advanced protection. Traffic redirection is defined under Service Composer/Security Policy for NSX version 6.0 or under Partner Security Services tab of the DFW menu in NSX version 6.1. The Partner Security Services tab provides a powerful and easy to use user interface to define what type of traffic needs to be redirected to which partner services. It follows the same policy definition construct as DFW, providing the same options for source field, destination field and services field. The only difference is in the action field; instead of Block/Allow/Reject, a user can select between redirect/no redirect followed by a partner list. Any partner that has been registered with NSX can be successfully deployed on the platform. Additionally, log options can be enabled for this traffic redirection rule. The DFW instance on an ESXi host contains 2 separate tables. The rule table is used to store all policy rules, while the connection tracker table caches flow entries for rules with permit actions. One DFW instance is permitted per VM vNIC. A specific flow is identified by the 5-tuple information consisting source IP address, destination IP address, protocols, L4 source port, and L4 destination port fields. By default, DFW does not perform a lookup on L4 source port, but it can be configured to do so by defining a specific policy rule. VMW_ORCT_CSCO_ 598

No.	'111 Patent Claim 9	VMware NSX
		Logical Port Mirroring Logical port mirroring allows you to duplicate (or "mirror") all traffic coming into or out of a logical switch port that is attached to a VM VIF or to an L2 Gateway port. The typical reasons you'll use port mirroring are: • Troubleshooting: Analyze, debug, and diagnose errors on a network by inspecting the traffic. • Compliance and monitoring: Forward all monitored traffic to a network appliance for analysis and remediation. The mirrored traffic is sent encapsulated within a GRE Tunnel to a collector so that all the original information of the packets is preserved while traversing the network to a remote location. VMW_ORCT_CSCO_ 629 *Export Event Logs via Syslog* NSX supports exporting event logs to a standard syslog collector for archiving and analysis using third-party tools. NSX Controllers, transport nodes, the NSX Manager appliance, and the NVP OVS vApp can export event logs in this way. • To configure syslog export of Controller event logs, see Syslog Export of NSX Controller Event and System Logs, below. • To configure syslog export of event logs from other types of NSX appliances, see Syslog Export of NSX Appliance Event Logs, below. VMW_ORCT_CSCO_ 4640

No.	'111 Patent Claim 9	VMware NSX
		 Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, 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 9 of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references. For example, Swenson discloses the network controller comprising a flow analyzer for analyzing and inspecting the packet.

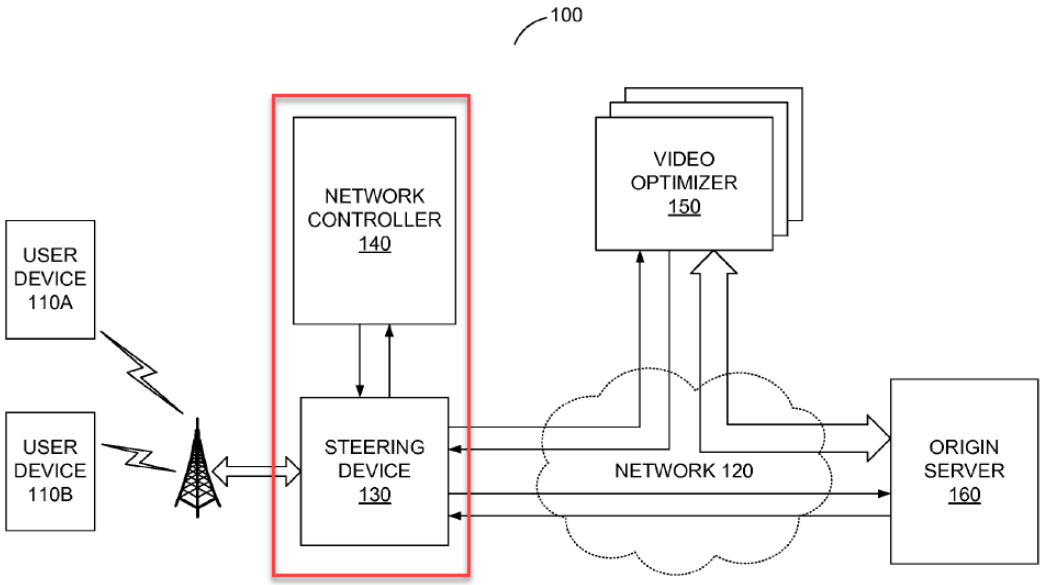
No.	'111 Patent Claim 9	VMware NSX
		Swenson at [0026] ("The steering device 130 may be a load balancer or a router located between the user device 110 and the network 120. The steering device 130 provides the user device 110 with access to the network and thus, provides the gateway through which the 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.") 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.") 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

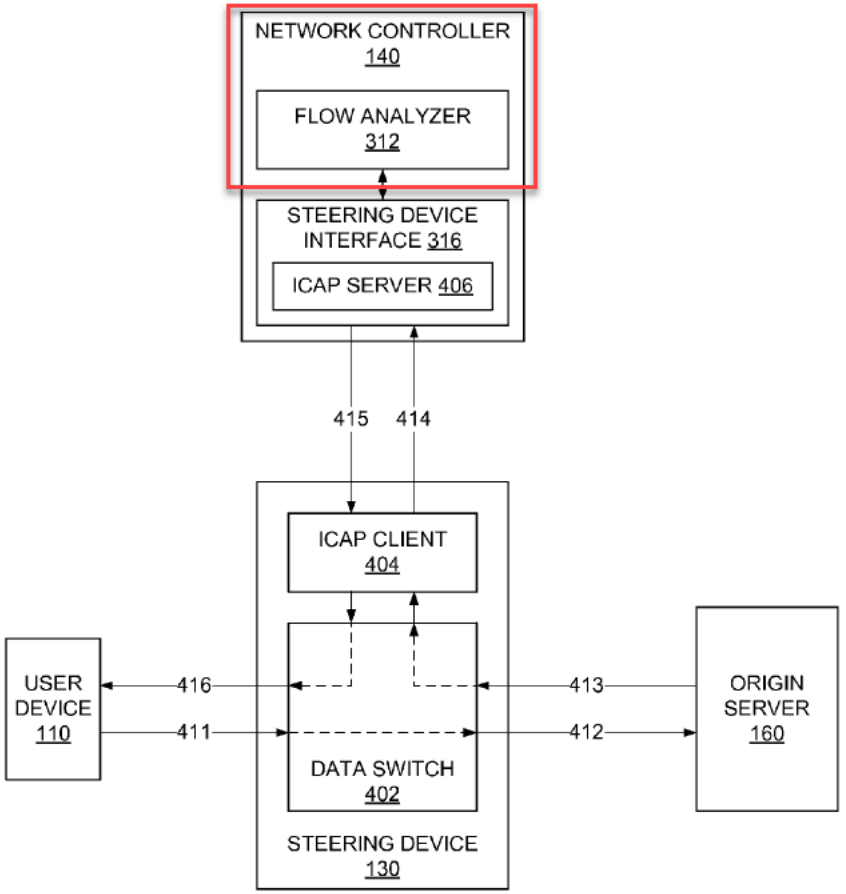
No.	'111 Patent Claim 9	VMware NSX
		devices attached, rout-ers through which the traffic is passing, geography regions, the backhaul segments, and time-of-day of the flows. For example, in a mobile network, the cell tower to which a user device is attached can be most useful, since it is the location 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.") 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.") 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.")

No.	'111 Patent Claim 9	VMware NSX
		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.") 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.") 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

No.	'111 Patent Claim 9	VMware NSX
		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.") 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.") 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

No.	'111 Patent Claim 9	VMware NSX
		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.") 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.") Swenson at Figure 1 (annotation added)

No.	'111 Patent Claim 9	VMware NSX
		 FIG. 1 Swenson at Figure 4A (annotation added)

No.	'111 Patent Claim 9	VMware NSX
		 FIG. 4A

No.	'111 Patent Claim 9	VMware NSX
		For example, Copeland discloses analyzing packets received by the intrusion detection engine on the monitoring appliance. 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.") 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.") 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.")

No.	'111 Patent Claim 9	VMware NSX
		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 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.") 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.") 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.") 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

No.	'111 Patent Claim 9	VMware NSX
		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.) 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.") 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.") 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.") 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

No.	'111 Patent Claim 9	VMware NSX
		differing flow characteristics. Consequently, the flow engine 155 analyzes each of the services separately.") 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	VMware NSX
12	The method according to claim 9, wherein the analyzing comprises applying security or data analytic application.	VMware NSX System discloses the method according to claim 9, wherein the analyzing comprises applying security or data analytic application. For example, VMware NSX System discloses analyzing the packet through security services, distributed firewall inspection, and monitoring by the entity that receives the redirected packet. *See supra* at Claim 9. NSX Design Guide at 9-10

ii. NSX Services Platform

NSX is a services platform that enables both security services and distributed firewall.

These services are either a built-in part of NSX or available from 3rd party vendors. Existing physical devices – examples include physical load balancers, firewalls, syslog collectors, and monitoring devices – can also be integrated with NSX. All services can be orchestrated via the NSX consumption model (explained in section 2.1.3 below).

2.1.1.1 Networking and Edge Services

NSX provides built-in networking and edge services including logical load balancing, L2/L3 VPN services, edge firewalling, and DHCP/NAT.



Figure 6 - NSX Logical Network Services

Edge Firewall: Edge firewall services are part of the NSX Edge Services Gateway (ESG). The Edge firewall provides essential perimeter firewall protection which can be used in addition to a physical perimeter firewall. The ESG-based firewall is useful in developing PCI zones, multi-tenant environments, or dev-ops style connectivity without forcing the inter-tenant or inter-zone traffic onto the physical network.

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1958 of 2137

No.	'111 Patent Claim 12	VMware NSX
		NSX Design Guide at 11 NSX includes built-in security services – distributed firewalling for east-west L2-L4 traffic, edge firewalling for north-south traffic, and SpoofGuard for validation of IP/MAC identity. **Distributed Firewall** – Security enforcement is done directly at the kernel and vNIC level. This enables highly scalable firewall rule enforcement by avoiding bottlenecks on physical appliances. The firewall is distributed in kernel, minimizing CPU overhead while enabling line-rate performance. NSX also provides an extensible framework, allowing security vendors to provide an umbrella of security services. Popular offerings include anti-virus/anti-malware/anti-bot solutions, L7 firewalling, IPS/IDS (host and network based) services, file integrity monitoring, and vulnerability management of guest VMs. NSX Design Guide at 29

No.	'111 Patent Claim 12	VMware NSX
		Traffic redirection to third party vendor provides the capability to steer a particular type of traffic to a selected partner services VM. For instance, web traffic from the Internet to a set of Apache or Tomcat servers can be redirected to an L4-L7 deep packet inspection firewall for advanced protection. Traffic redirection is defined under Service Composer/Security Policy for NSX version 6.0 or under Partner Security Services tab of the DFW menu in NSX version 6.1. The Partner Security Services tab provides a powerful and easy to use user interface to define what type of traffic needs to be redirected to which partner services. It follows the same policy definition construct as DFW, providing the same options for source field, destination field and services field. The only difference is in the action field; instead of Block/Allow/Reject, a user can select between redirect/no redirect followed by a partner list. Any partner that has been registered with NSX can be successfully deployed on the platform. Additionally, log options can be enabled for this traffic redirection rule. The DFW instance on an ESXi host contains 2 separate tables. The rule table is used to store all policy rules, while the connection tracker table caches flow entries for rules with permit actions. One DFW instance is permitted per VM vNIC. A specific flow is identified by the 5-tuple information consisting source IP address, destination IP address, protocols, L4 source port, and L4 destination port fields. By default, DFW does not perform a lookup on L4 source port, but it can be configured to do so by defining a specific policy rule. VMW_ORCT_CSCO_ 598

No.	'111 Patent Claim 12	VMware NSX
		Logical Port Mirroring Logical port mirroring allows you to duplicate (or "mirror") all traffic coming into or out of a logical switch port that is attached to a VM VIF or to an L2 Gateway port. The typical reasons you'll use port mirroring are: • Troubleshooting: Analyze, debug, and diagnose errors on a network by inspecting the traffic. • Compliance and monitoring: Forward all monitored traffic to a network appliance for analysis and remediation. The mirrored traffic is sent encapsulated within a GRE Tunnel to a collector so that all the original information of the packets is preserved while traversing the network to a remote location. VMW_ORCT_CSCO_ 629 *Export Event Logs via Syslog* NSX supports exporting event logs to a standard syslog collector for archiving and analysis using third-party tools. NSX Controllers, transport nodes, the NSX Manager appliance, and the NVP OVS vApp can export event logs in this way. • To configure syslog export of Controller event logs, see Syslog Export of NSX Controller Event and System Logs, below. • To configure syslog export of event logs from other types of NSX appliances, see Syslog Export of NSX Appliance Event Logs, below. VMW_ORCT_CSCO_ 3404

No.	'111 Patent Claim 12	VMware NSX
		Edge Firewall Edge Firewall provides perimeter security functionality including firewall, Network Address Translation (NAT) as well as Site to site IPSec and SSL VPN functionality. This solution is available in the virtual machine form factor and can be deployed in a High Availability mode. **Working with Firewall Rules** Distributed Firewall rules and Edge Firewall rules can be managed in a centralized manner on the Firewall tab. In a multi-tenant environment, providers can define high-level traffic flow rules on the centralized Firewall user interface. Rules defined on the centralized level are referred to as pre rules. Tenants can then add rules at an individual NSX Edge level, which are referred to as local rules. Each traffic session is checked against the top rule in the Firewall table before moving down the subsequent rules in the table. The first rule in the table that matches the traffic parameters is enforced. Rules are displayed (and enforced) in the following order: 1 User defined pre rules have the highest priority, and are enforced in top-to-bottom ordering with a per-virtual NIC level precedence. 2 Auto plumbed rules. 3 Local rules defined at an NSX Edge level. 4 Service Composer rules - a separate section for each policy. You cannot edit these rules in the Firewall table, but you can add rules at the top of a security policy firewall rules section. If you do so, you must re-synchronize the rules in Service Composer. For more information, see Chapter 11, "Service Composer," on page 121. 5 Default Distributed Firewall rule Note that firewall rules are enforced only on clusters on which you have enabled firewall. For information on preparing clusters, see the *NSX Installation and Upgrade Guide*. VMW_ORCT_CSCO_4033

Distributed Firewall is a hypervisor kernel-embedded firewall that provides visibility and control for virtualized workloads and networks. You can create access control policies based on VMware vCenter objects like datacenters and clusters, virtual machine names and tags, network constructs like IP/VLAN/VXLAN addresses, as well as user group identity from Active Directory. Firewall rules are enforced at the vNIC level of each virtual machine to provide consistent access control even when the virtual machine gets vMotioned. The hypervisor-embedded nature of the firewall delivers close to line rate throughput to enable higher workload consolidation on physical servers. The distributed nature of the firewall provides a scale-out architecture that automatically extends firewall capacity when additional hosts are added to a prepared cluster.

Distributed Firewall offers multiple sets of configurable rules: Layer 3 (L3) rules (General tab) and Layer 2 (L2) rules (Ethernet tab). Layer 2 firewall rules are processed before Layer 3 rules. The default firewall rule allows all L3 and L2 traffic to pass through all clusters in your infrastructure. The default rule is always at the bottom of the rules table and cannot be deleted or added to. However, you can change the Action element of the rule from Allow to Block, add comments for the rule, and indicate whether traffic for that rule should be logged.

Firewall rules are created at the global scope, but you can then narrow the scope (datacenter, cluster, distributed virtual port group, network, virtual machine, vNIC, or virtual wire) at which you want to apply the rule by using the AppliedTo keyword.

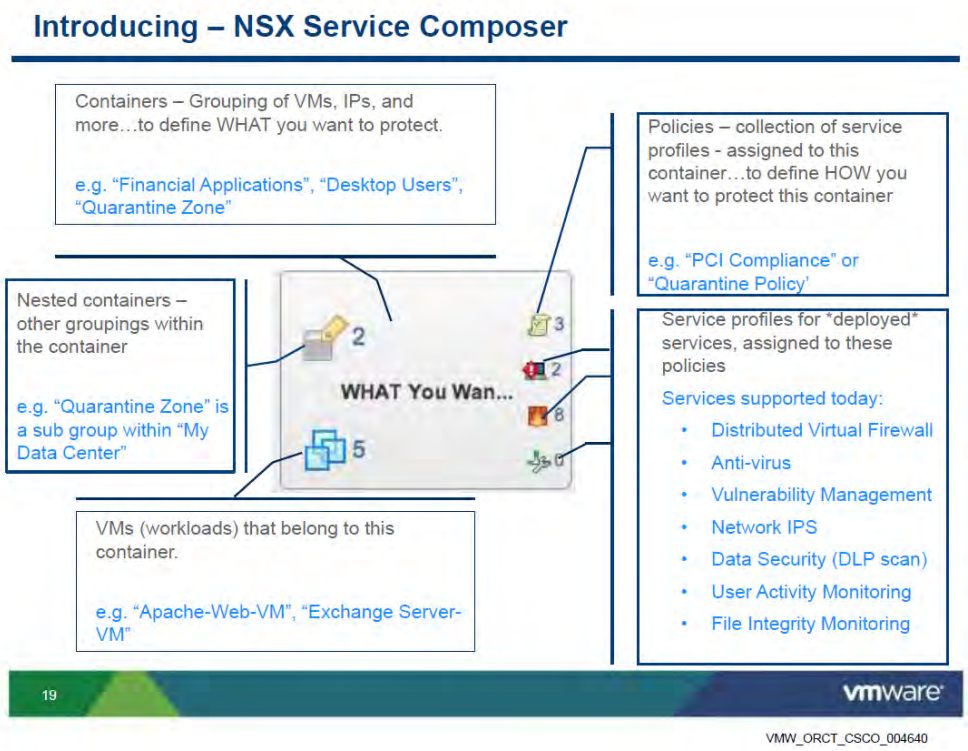
User defined firewall rules are enforced in top-to-bottom ordering, with a per-virtual NIC level precedence. Each traffic session is checked against the top rule in the Firewall table before moving down the subsequent rules in the table. The first rule in the table that matches the traffic parameters is enforced.

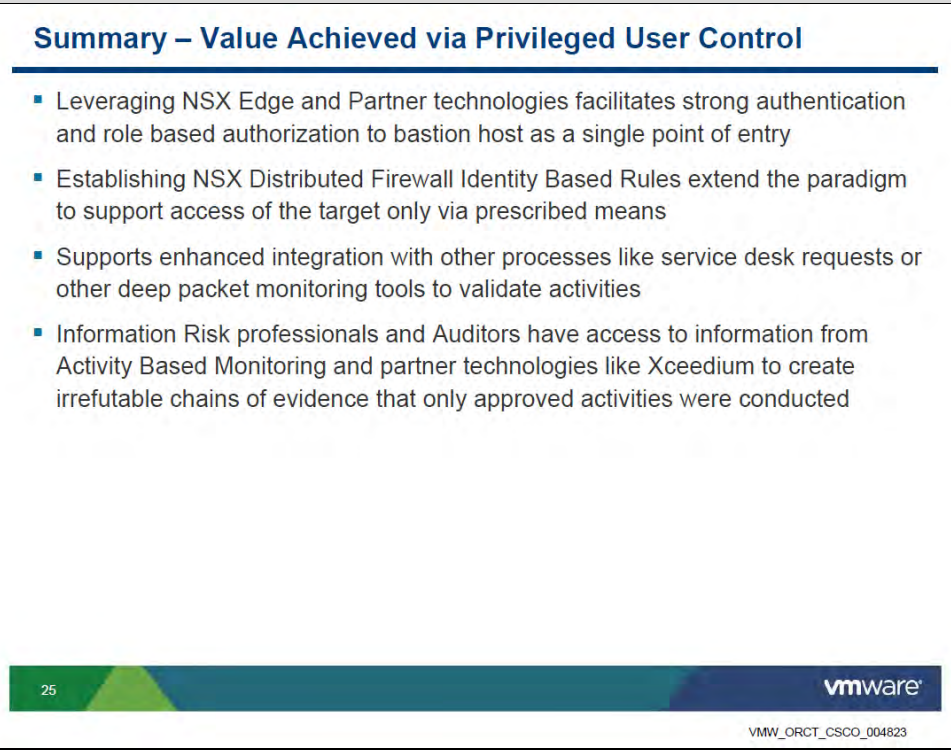
The followng table lists the elements that can be used in firewall rules.

Table 9-1. Firewall rule elements

Element	Keyword for API	Used In
application	Application	service
application group	ApplicationGroup	service
cluster compute resource	ClusterComputeResource	AppliedTo
datacenter	Datacenter	source/destination AppliedTo
distributed virtual port group	DistributedVirtualPortgroup	source/destination AppliedTo
global root	GlobalRoot	source/destination
IPv4 addresses	Ipv4Address	source/destination
IPv6 addresses	Ipv6Address	source/destination
logical switch	VirtualWire	source/destination AppliedTo
network	Network	AppliedTo

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1963 of 2137

No.	'111 Patent Claim 12	VMware NSX
		VMW_ORCT_CSCO_ 4640  VMW_ORCT_CSCO_ 4823

No.	'111 Patent Claim 12	VMware NSX
		

No.	'111 Patent Claim 13	VMware NSX
13	The method according to claim 9, wherein the analyzing comprises applying security application	VMware NSX System discloses the method according to claim 9, wherein the analyzing comprises applying security application that comprises firewall or intrusion detection functionality. For example, VMware NSX System discloses analyzing the packet through security services, distributed firewall inspection, and monitoring by the entity that receives the redirected packet. *See supra* at Claim 9. NSX Design Guide at 9-10

No.	'111 Patent Claim 13	VMware NSX
	that comprises firewall or intrusion detection functionality.	ii. NSX Services Platform NSX is a services platform that enables both security services and distributed firewall. 9 These services are either a built-in part of NSX or available from 3rd party vendors. Existing physical devices – examples include physical load balancers, firewalls, syslog collectors, and monitoring devices – can also be integrated with NSX. All services can be orchestrated via the NSX consumption model (explained in section 2.1.3 below). 2.1.1.1 Networking and Edge Services NSX provides built-in networking and edge services including logical load balancing, L2/L3 VPN services, edge firewalling, and DHCP/NAT.  Figure 6 - NSX Logical Network Services **Edge Firewall:** Edge firewall services are part of the NSX Edge Services Gateway (ESG). The Edge firewall provides essential perimeter firewall protection which can be used in addition to a physical perimeter firewall. The ESG-based firewall is useful in developing PCI zones, multi-tenant environments, or dev-ops style connectivity without forcing the inter-tenant or inter-zone traffic onto the physical network. NSX Design Guide at 11

No.	'111 Patent Claim 13	VMware NSX
		NSX includes built-in security services – distributed firewalling for east-west L2-L4 traffic, edge firewalling for north-south traffic, and SpoofGuard for validation of IP/MAC identity. **Distributed Firewall** – Security enforcement is done directly at the kernel and vNIC level. This enables highly scalable firewall rule enforcement by avoiding bottlenecks on physical appliances. The firewall is distributed in kernel, minimizing CPU overhead while enabling line-rate performance. NSX also provides an extensible framework, allowing security vendors to provide an umbrella of security services. Popular offerings include anti-virus/anti-malware/anti-bot solutions, L7 firewalling, IPS/IDS (host and network based) services, file integrity monitoring, and vulnerability management of guest VMs. NSX Design Guide at 29

No.	'111 Patent Claim 13	VMware NSX
		Traffic redirection to third party vendor provides the capability to steer a particular type of traffic to a selected partner services VM. For instance, web traffic from the Internet to a set of Apache or Tomcat servers can be redirected to an L4-L7 deep packet inspection firewall for advanced protection. Traffic redirection is defined under Service Composer/Security Policy for NSX version 6.0 or under Partner Security Services tab of the DFW menu in NSX version 6.1. The Partner Security Services tab provides a powerful and easy to use user interface to define what type of traffic needs to be redirected to which partner services. It follows the same policy definition construct as DFW, providing the same options for source field, destination field and services field. The only difference is in the action field; instead of Block/Allow/Reject, a user can select between redirect/no redirect followed by a partner list. Any partner that has been registered with NSX can be successfully deployed on the platform. Additionally, log options can be enabled for this traffic redirection rule. The DFW instance on an ESXi host contains 2 separate tables. The rule table is used to store all policy rules, while the connection tracker table caches flow entries for rules with permit actions. One DFW instance is permitted per VM vNIC. A specific flow is identified by the 5-tuple information consisting source IP address, destination IP address, protocols, L4 source port, and L4 destination port fields. By default, DFW does not perform a lookup on L4 source port, but it can be configured to do so by defining a specific policy rule. VMW_ORCT_CSCO_ 598

No.	'111 Patent Claim 13	VMware NSX
		Logical Port Mirroring Logical port mirroring allows you to duplicate (or "mirror") all traffic coming into or out of a logical switch port that is attached to a VM VIF or to an L2 Gateway port. The typical reasons you'll use port mirroring are: • Troubleshooting: Analyze, debug, and diagnose errors on a network by inspecting the traffic. • Compliance and monitoring: Forward all monitored traffic to a network appliance for analysis and remediation. The mirrored traffic is sent encapsulated within a GRE Tunnel to a collector so that all the original information of the packets is preserved while traversing the network to a remote location. VMW_ORCT_CSCO_ 629 *Export Event Logs via Syslog* NSX supports exporting event logs to a standard syslog collector for archiving and analysis using third-party tools. NSX Controllers, transport nodes, the NSX Manager appliance, and the NVP OVS vApp can export event logs in this way. • To configure syslog export of Controller event logs, see Syslog Export of NSX Controller Event and System Logs, below. • To configure syslog export of event logs from other types of NSX appliances, see Syslog Export of NSX Appliance Event Logs, below. VMW_ORCT_CSCO_ 3404

No.	'111 Patent Claim 13	VMware NSX
		Edge Firewall Edge Firewall provides perimeter security functionality including firewall, Network Address Translation (NAT) as well as Site to site IPSec and SSL VPN functionality. This solution is available in the virtual machine form factor and can be deployed in a High Availability mode. **Working with Firewall Rules** Distributed Firewall rules and Edge Firewall rules can be managed in a centralized manner on the Firewall tab. In a multi-tenant environment, providers can define high-level traffic flow rules on the centralized Firewall user interface. Rules defined on the centralized level are referred to as pre rules. Tenants can then add rules at an individual NSX Edge level, which are referred to as local rules. Each traffic session is checked against the top rule in the Firewall table before moving down the subsequent rules in the table. The first rule in the table that matches the traffic parameters is enforced. Rules are displayed (and enforced) in the following order: 1 User defined pre rules have the highest priority, and are enforced in top-to-bottom ordering with a per-virtual NIC level precedence. 2 Auto plumbed rules. 3 Local rules defined at an NSX Edge level. 4 Service Composer rules - a separate section for each policy. You cannot edit these rules in the Firewall table, but you can add rules at the top of a security policy firewall rules section. If you do so, you must re-synchronize the rules in Service Composer. For more information, see Chapter 11, "Service Composer," on page 121. 5 Default Distributed Firewall rule Note that firewall rules are enforced only on clusters on which you have enabled firewall. For information on preparing clusters, see the *NSX Installation and Upgrade Guide*. VMW_ORCT_CSCO_4033

Distributed Firewall is a hypervisor kernel-embedded firewall that provides visibility and control for virtualized workloads and networks. You can create access control policies based on VMware vCenter objects like datacenters and clusters, virtual machine names and tags, network constructs like IP/VLAN/VXLAN addresses, as well as user group identity from Active Directory. Firewall rules are enforced at the vNIC level of each virtual machine to provide consistent access control even when the virtual machine gets vMotioned. The hypervisor-embedded nature of the firewall delivers close to line rate throughput to enable higher workload consolidation on physical servers. The distributed nature of the firewall provides a scale-out architecture that automatically extends firewall capacity when additional hosts are added to a prepared cluster.

Distributed Firewall offers multiple sets of configurable rules: Layer 3 (L3) rules (General tab) and Layer 2 (L2) rules (Ethernet tab). Layer 2 firewall rules are processed before Layer 3 rules. The default firewall rule allows all L3 and L2 traffic to pass through all clusters in your infrastructure. The default rule is always at the bottom of the rules table and cannot be deleted or added to. However, you can change the Action element of the rule from Allow to Block, add comments for the rule, and indicate whether traffic for that rule should be logged.

Firewall rules are created at the global scope, but you can then narrow the scope (datacenter, cluster, distributed virtual port group, network, virtual machine, vNIC, or virtual wire) at which you want to apply the rule by using the AppliedTo keyword.

User defined firewall rules are enforced in top-to-bottom ordering, with a per-virtual NIC level precedence. Each traffic session is checked against the top rule in the Firewall table before moving down the subsequent rules in the table. The first rule in the table that matches the traffic parameters is enforced.

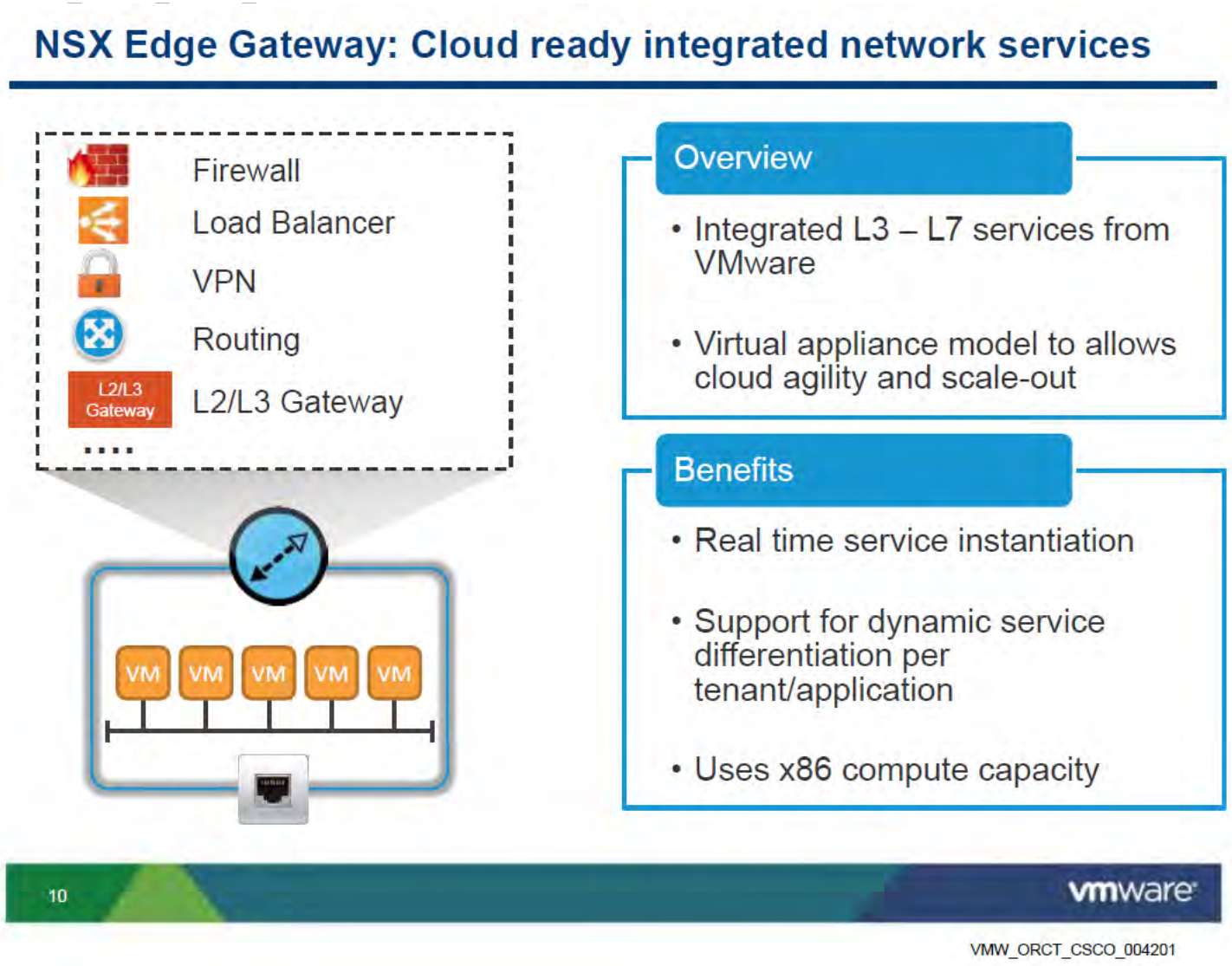
The followng table lists the elements that can be used in firewall rules.

Table 9-1. Firewall rule elements

Element	Keyword for API	Used In
application	Application	service
application group	ApplicationGroup	service
cluster compute resource	ClusterComputeResource	AppliedTo
datacenter	Datacenter	source/destination AppliedTo
distributed virtual port group	DistributedVirtualPortgroup	source/destination AppliedTo
global root	GlobalRoot	source/destination
IPv4 addresses	Ipv4Address	source/destination
IPv6 addresses	Ipv6Address	source/destination
logical switch	VirtualWire	source/destination AppliedTo
network	Network	AppliedTo

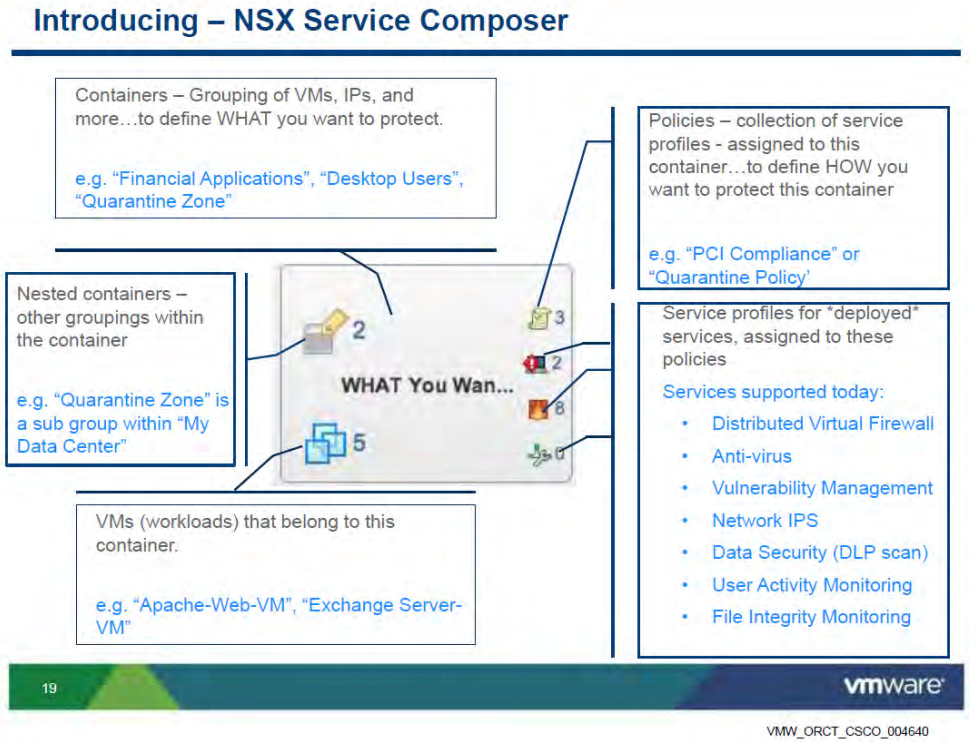
Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1971 of 2137

No.	'111 Patent Claim 13	VMware NSX
		VMW_ORCT_CSCO_ 4201



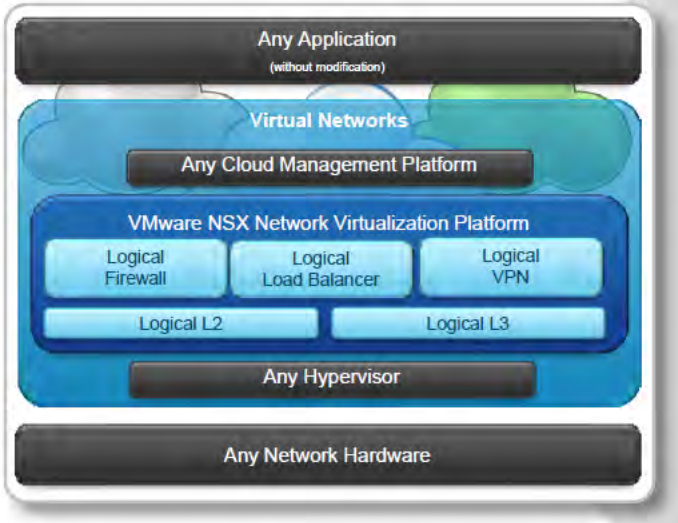
No.	'111 Patent Claim 13	VMware NSX
		VMW_ORCT_CSCO_ 4215



No.	'111 Patent Claim 13	VMware NSX
		VMW_ORCT_CSCO_ 4285 VMW_ORCT_CSCO_ 4640 

No.	'111 Patent Claim 13	VMware NSX

VMware NSX – Networking & Security Capabilities



Logical Switching– Layer 2 over Layer 3, decoupled from the physical network

Logical Routing– Routing between virtual networks without exiting the software container

Logical Firewall – Distributed Firewall, Kernel Integrated, High Performance

Logical Load Balancer – Application Load Balancing in software

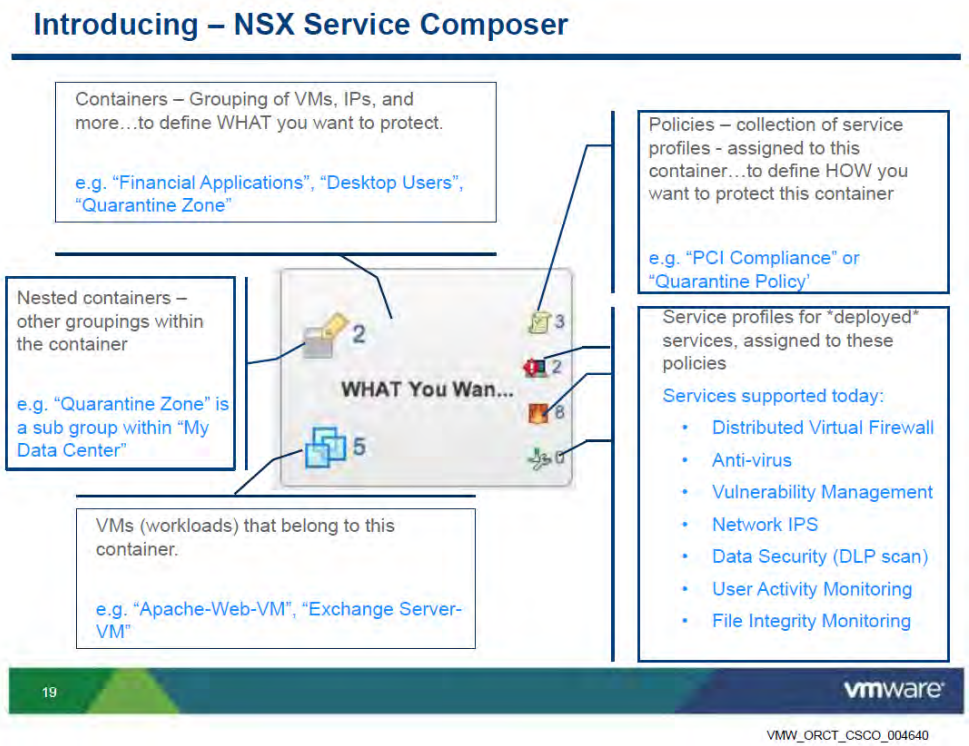
Logical VPN – Site-to-Site & Remote Access VPN in software

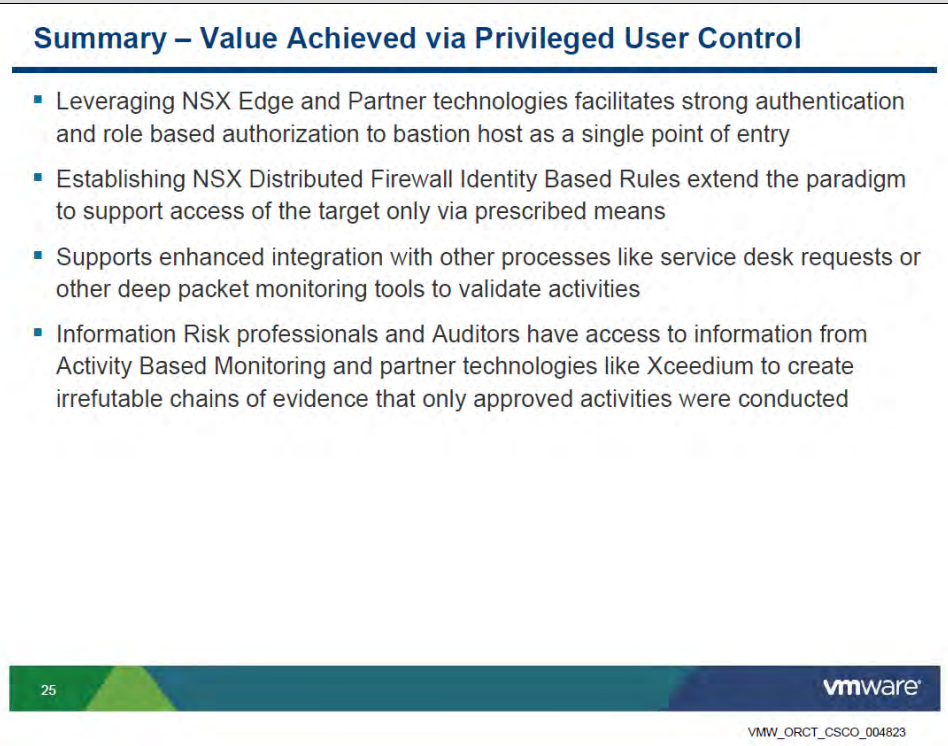
NSX API – RESTful API for integration into any Cloud Management Platform

Partner Eco-System

vmware

VMW_ORCT_CSCO_004285

No.	'111 Patent Claim 13	VMware NSX
		VMW_ORCT_CSCO_ 4640  VMW_ORCT_CSCO_ 4823

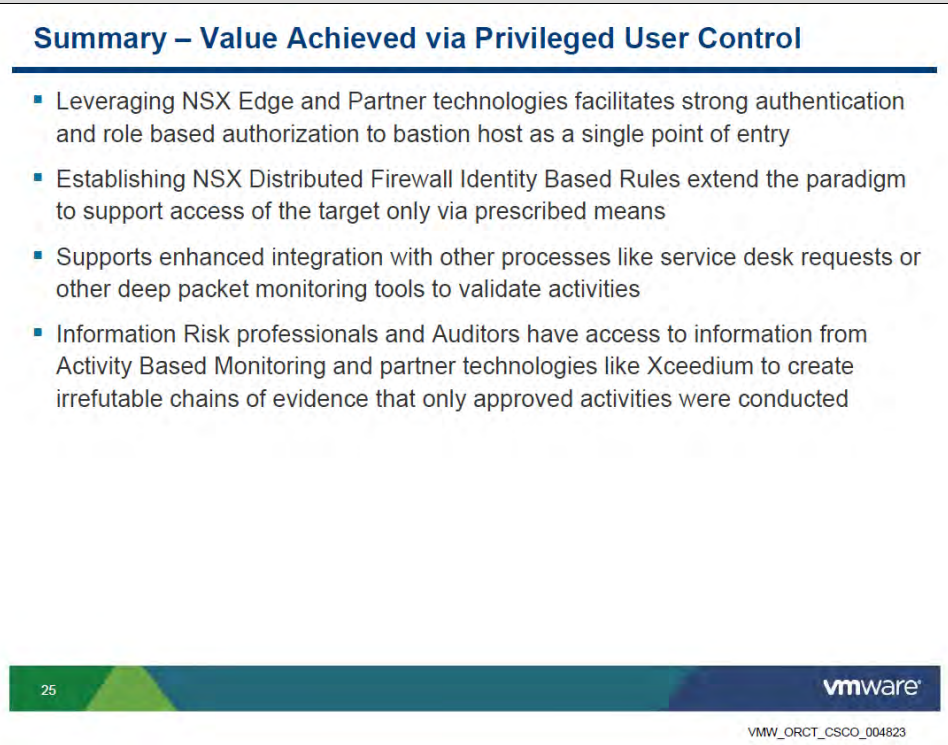
No.	'111 Patent Claim 13	VMware NSX
		

No.	'111 Patent Claim 14	VMware NSX
14	The method according to claim 9, wherein the analyzing comprises performing Deep Packet Inspection (DPI) or using a DPI	Vmware NSX 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. For example, Vmware NSX System discloses analyzing the packet through firewall deep packet inspection by the entity that receives the redirected packet. *See supra* at Claim 9.

No.	'111 Patent Claim 14	VMware NSX
	engine on the packet.	NSX Design Guide at 29 Traffic redirection to third party vendor provides the capability to steer a particular type of traffic to a selected partner services VM. For instance, web traffic from the Internet to a set of Apache or Tomcat servers can be redirected to an L4-L7 deep packet inspection firewall for advanced protection. Traffic redirection is defined under Service Composer/Security Policy for NSX version 6.0 or under Partner Security Services tab of the DFW menu in NSX version 6.1. The Partner Security Services tab provides a powerful and easy to use user interface to define what type of traffic needs to be redirected to which partner services. It follows the same policy definition construct as DFW, providing the same options for source field, destination field and services field. The only difference is in the action field; instead of Block/Allow/Reject, a user can select between redirect/no redirect followed by a partner list. Any partner that has been registered with NSX can be successfully deployed on the platform. Additionally, log options can be enabled for this traffic redirection rule. The DFW instance on an ESXi host contains 2 separate tables. The rule table is used to store all policy rules, while the connection tracker table caches flow entries for rules with permit actions. One DFW instance is permitted per VM vNIC. A specific flow is identified by the 5-tuple information consisting source IP address, destination IP address, protocols, L4 source port, and L4 destination port fields. By default, DFW does not perform a lookup on L4 source port, but it can be configured to do so by defining a specific policy rule. VMW_ORCT_CSCO_ 94

No.	'111 Patent Claim 14	VMware NSX
		flow and flow-based networking RFC 3697 defines a network flow as "a sequence of packets sent from a particular source to a particular unicast, anycast, or multicast destination that the source desires to label as a flow." A flow is any unique stream of network traffic (as identified by its packet type, source and destination) that can be categorized based on inspection of a packet's fields. Flow-based networking creates a flow table that pairs each flow with an action that determines whether and how packets in that flow will be forwarded. VMW_ORCT_CSCO_ 368 *Traceflow, the NSX Packet Tracing Tool* NSX Manager and the NVP API provide the Traceflow tool to allow you to inspect the path of a packet as it travels from one logical port on the logical network to another logical port on the same network. Traceflow traces the transport node-level path of a packet injected at a logical port. The trace packet traverses the logical switch overlay, but is not visible to interfaces attached to the logical switch. In other words, no packet is actually delivered to the test packet's intended recipients. VMW_ORCT_CSCO_ 371

No.	'111 Patent Claim 14	VMware NSX
		Using Traceflow via the NVP API Below we describe using Traceflow via the NVP API, as viewed in the NSX Manager: 1. In NSX Manager, open the **API Inspector**. 2. Select the command, **Execute traceflow**, which runs the API command `POST /ws.v1/lswitch/<lswitch-uuid>/lport/<lport-uuid>/traceflow` 3. Specify the test packet's field values. These fields are described in more detail in the preceding section, Using Traceflow in NSX Manager's Port Connection Page (page 335). The fields: – source logical switch and source logical port in the **Lswitch UUID** (lswitch-uuid) and **Lport UUID** (lport-uuid) fields – (optional) *True* or *False* for **Ignore Cluster Majority** (_ignore_cluster_majority). This setting allows the call to proceed even if the Controller that fields the request is not connected to a Controller Cluster majority. – the destination port in the **Dst MAC** (dst_mac) field. – the Ethertype of the packet in **Eth Type** (eth_type) – the **Frame Size** (frame_size) – optional user-specified data in the **Payload** (payload) field – the originating MAC address in **Src MAC** (src_mac) – (optional) the **Src Transport Node UUID** (src_transport_node_uuid) field specifies the ID of the transport node backing the logical port that will send the test packet. This is useful in cases in which a logical port is backed by multiple transport node ports (that is, ports on separate transport nodes). For example, two or more Service Nodes are typically deployed to provide a high-availability MDI connection. In such cases, you can specify the desired transport node's ID here. If you leave this field empty, Traceflow chooses a port at random. – the **Timeout** (timeout) in milliseconds (min. 1000 ms; max. 10000 ms). This specifies how long Traceflow will wait, in total, to observe whether the packet has been delivered to all destinations in the NSX network. 4. Click **Submit Request** to run the test. See Interpreting Traceflow Results (page 337) for help understanding the results. VMW_ORCT_CSCO_ 4823

No.	'111 Patent Claim 14	VMware NSX
		

No.	'111 Patent Claim 15	VMware NSX
15[a]	The method according to claim 9, wherein the packet comprises distinct header and payload fields, and	VMware NSX System discloses the method according to claim 9, wherein the packet comprises distinct header and payload fields. For example, VMware NSX System discloses packets with specific header and payload fields. *See supra* at Claim 9. NSX Design Guide at 29

No.	'111 Patent Claim 15	VMware NSX
		Traffic redirection to third party vendor provides the capability to steer a particular type of traffic to a selected partner services VM. For instance, web traffic from the Internet to a set of Apache or Tomcat servers can be redirected to an L4-L7 deep packet inspection firewall for advanced protection. Traffic redirection is defined under Service Composer/Security Policy for NSX version 6.0 or under Partner Security Services tab of the DFW menu in NSX version 6.1. The Partner Security Services tab provides a powerful and easy to use user interface to define what type of traffic needs to be redirected to which partner services. It follows the same policy definition construct as DFW, providing the same options for source field, destination field and services field. The only difference is in the action field; instead of Block/Allow/Reject, a user can select between redirect/no redirect followed by a partner list. Any partner that has been registered with NSX can be successfully deployed on the platform. Additionally, log options can be enabled for this traffic redirection rule. The DFW instance on an ESXi host contains 2 separate tables. The rule table is used to store all policy rules, while the connection tracker table caches flow entries for rules with permit actions. One DFW instance is permitted per VM vNIC. A specific flow is identified by the 5-tuple information consisting source IP address, destination IP address, protocols, L4 source port, and L4 destination port fields. By default, DFW does not perform a lookup on L4 source port, but it can be configured to do so by defining a specific policy rule. NSX Design Guide at 30

DFW rules are enforced as follows:

- Rules are processed in top-to-bottom ordering.
- Each packet is checked against the top rule in the rule table before moving down the subsequent rules in the table.
- The first rule in the table that matches the traffic parameters is enforced. No subsequent rules can be enforced as the search is then terminated for that packet.

Because of this behavior, it is always recommended to put the most granular policies at the top of the rule table. This will ensure they will be enforced before more specific rules.

The DFW default policy rule, located at the bottom of the rule table, is a "catch-all" rule; packets not matching any other rules will be enforced by the default rule. After the host preparation operation, the DFW default rule is set to 'allow' action. This ensures that VM-to-VM communication is not broken during staging or migration phases. It is a best practice to then change this default rule to 'block' action and enforce access control through a positive control model (i.e., only traffic defined in the firewall policy is allowed onto the network).

Figure 22 steps through policy rule lookup and packet flow:

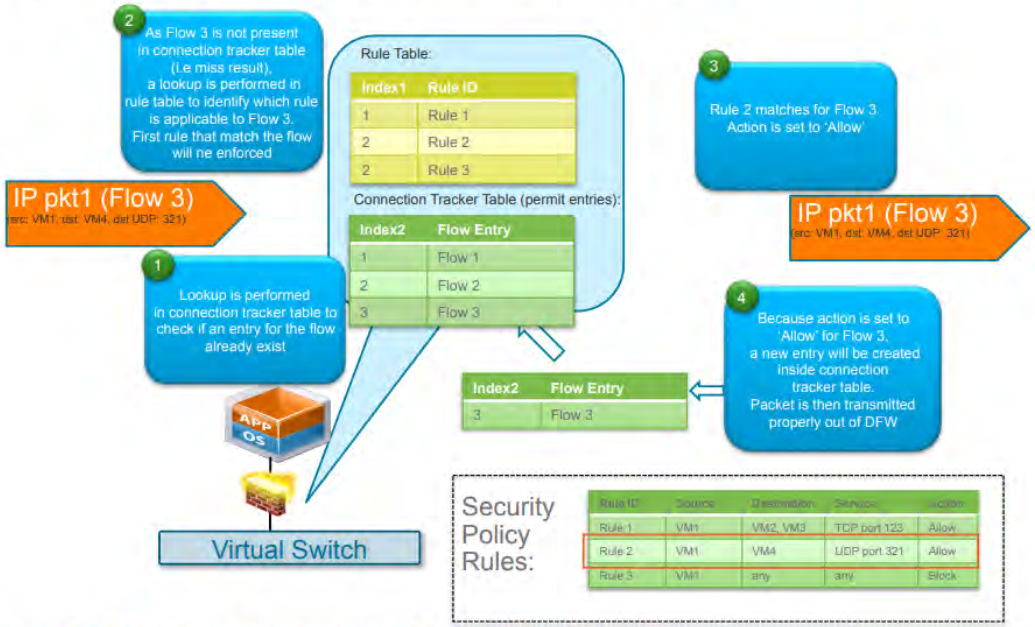
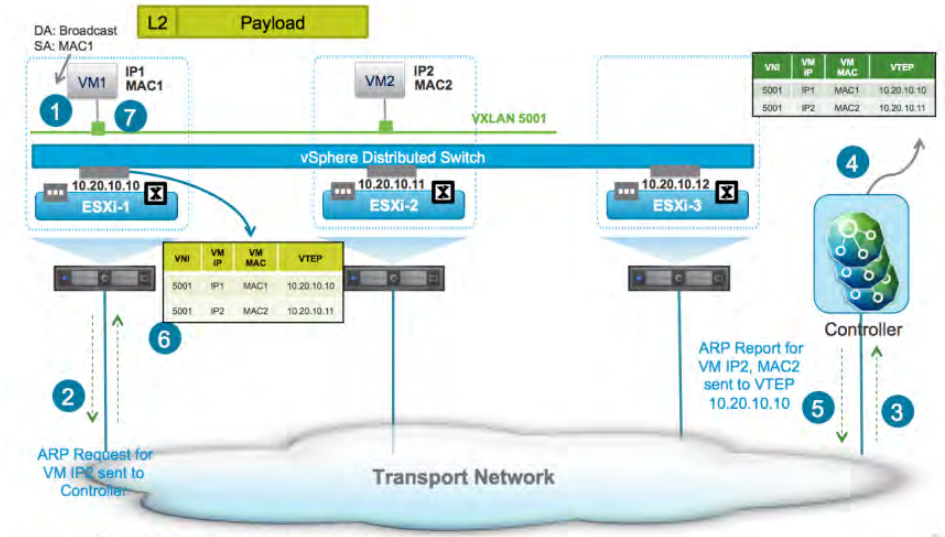


Figure 22 – DFW Policy Rule Lookup and Packet – First Packet

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1983 of 2137

No.	'111 Patent Claim 15	VMware NSX
		NSX Design Guide at 43-44 Figure 35 shows how ARP resolution is performed leveraging the control plane with the NSX controller when there is a need to establish unicast communication between virtual machines belonging to the same logical switch (i.e., VXLAN segment). 43  Figure 35 – ARP Resolution via NSX Controller

No.	'111 Patent Claim 15	VMware NSX
		NSX Design Guide at 45

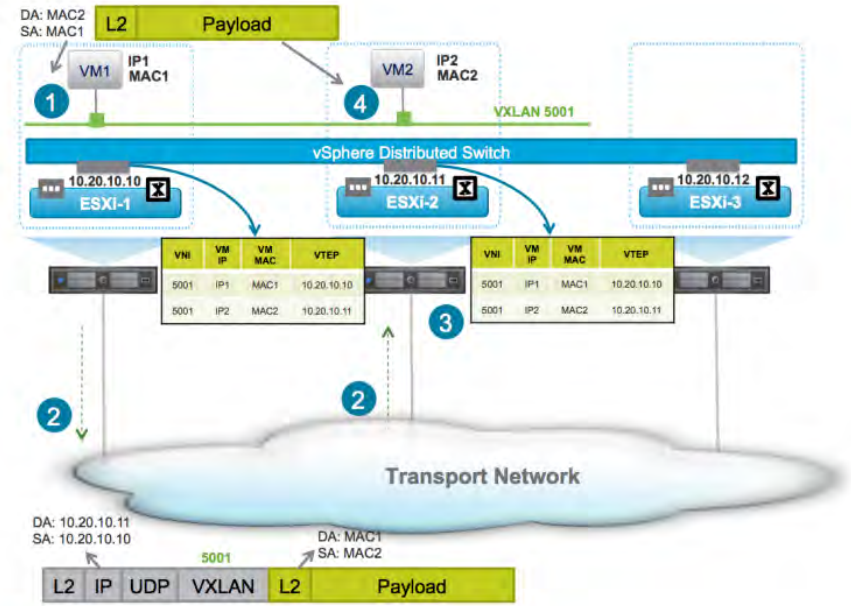
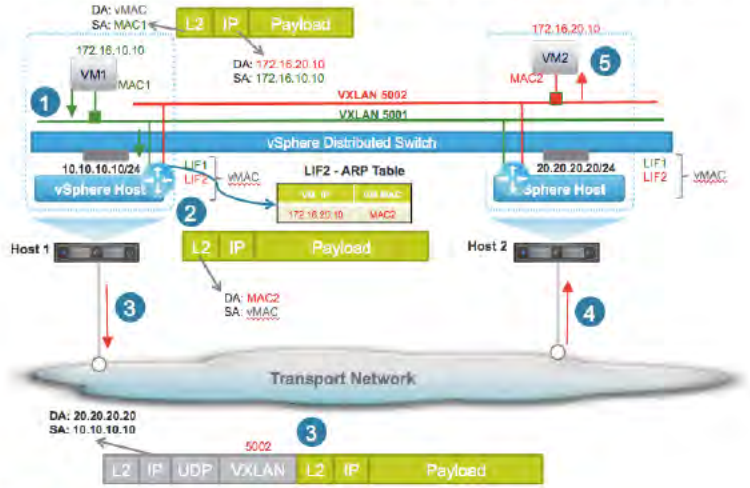


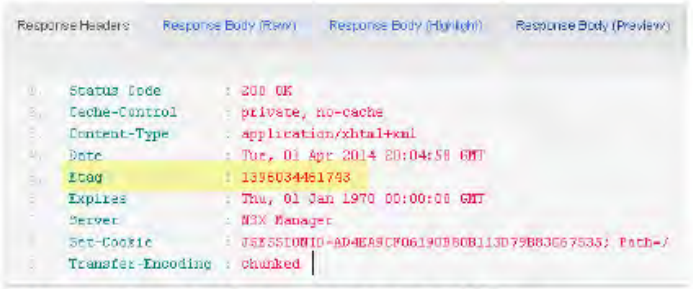
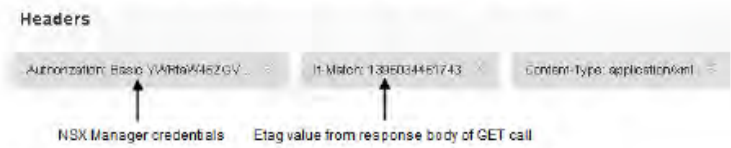
Figure 36 - Intra Logical Switch Unicast Communication

1. VM1 generates a data packet directed to VM2.
2. ESXi-1 learned about the location of VM2 from the ARP report received from the controller through control plane learning. It encapsulates the packet originated by VM1 in a VXLAN packet destined to the VTEP of ESXi2 – 10.20.10.11.
3. ESXi-2 receives the packet and leverages the information in the external IP header to learn about the location of VM1, associating VM1 MAC and IP addresses to the VTEP of ESXi-1.
4. The frame is delivered to VM2.

NSX Design Guide at 54

No.	'111 Patent Claim 15	VMware NSX
		The required steps to establish routed communication between two virtual machines connected to separate logical segments are shown in Figure 46.  Figure 46 - Routed Communication between Virtual Machines 1. VM1 sends a packet to VM2 connected to a different VXLAN segment. The packet is sent to the VM1 default gateway interfaces located on the local DLR (172.16.10.1). 2. A routing lookup is performed on the local DLR, which determines that the destination subnet is directly connected to DLR LIF2. A lookup is performed in the LIF2 ARP table to determine the MAC address associated with the VM2 IP address. If the ARP information is not available, the DLR will generate an ARP request on VXLAN 5002 to determine the required mapping information. 3. An L2 lookup is performed in the local MAC table to determine how to reach VM2. The original packet is VXLAN encapsulated and sent to the VTEP of ESXi2 (20.20.20.20). 4. ESXi-2 decapsulates the packet and performs an L2 lookup in the local MAC table associated to VXLAN 5002 segment. 5. The packet is delivered to the destination VM2. VMW_ORCT_CSCO_1444

No.	'111 Patent Claim 15	VMware NSX
		Table 9-2. Extensions for HTTP/HTTPS Protocol **Monitor Extension** / **Description** no-body — Do not wait for document body: stop reading after headers. Note that this still does an HTTP GET or POST, not a HEAD. max-age=*SECONDS* — Warn if document is more than SECONDS old. The number can also be in the form 10m for minutes, 10h for hours, or 10d for days. content-type=*STRING* — specify Content-Type header media type in POST calls. VMW_ORCT_CSCO_2035-36 **Modify Firewall Configuration** Follow the procedure below to modify the firewall configuration. 1 Run a GET call for the firewall configuration. 2 Extract the XML from the response body of the GET call and modify it as required. 3 From the Response Header in Step 1, copy the Etag header value.

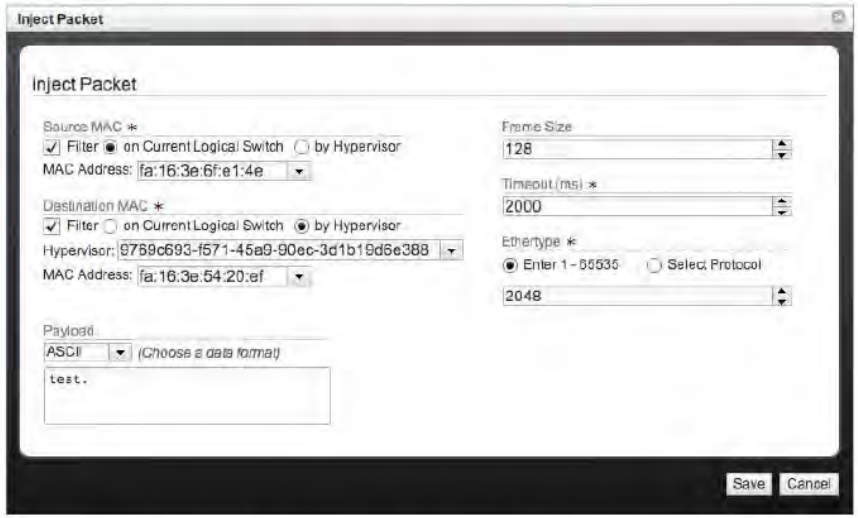
No.	'111 Patent Claim 15	VMware NSX
		 4 Add the number as the If-Match header in the PUT call.  5 Pass the modified XML as the Request Body in a PUT call. ■ Not all fields are required while sending the request. Refer to the optional field in the schema definition of various objects. All the optional fields are safe to be ignored while sending the configuration to server. For example, if an IP Set is referenced in the rule only IPSet and Type is needed in the Source/Destination objects and not Name and isValid tags. ■ IDs for new objects (rule/section) should be removed or set to zero. ■ If new entities (sections/rules) have been sent in the request, the response will contain the system-generated ids, which are assigned to these new entities. These ID identifies the resource and can be used in the urls if you want to operate on these entities using those URLs. VMW_ORCT_CSCO_ 2383-84

No.	'111 Patent Claim 15	VMware NSX

		Phase 1 Policy Not Matching
		The following lists Phase 1 Policy Not Matching Error logs.
		NSX Edge
		NSX Edge hangs in STATE_MAIN_I1 state. Look in /var/log/messages for information showing that the peer sent back an IKE message with "NO_PROPOSAL_CHOSEN" set.
		`000 #1: "s1-c1":500 STATE_MAIN_I1 (sent MI1, expecting MR1); EVENT_RETRANSMIT in 7s; nodpd; idle; import:admin initiate`
		`000 #1: pending Phase 2 for "s1-c1" replacing #0`
		`Aug 26 12:31:25 weiqing-desktop ipsec[6569]: | got payload 0x800(ISAKMP_NEXT_N) needed: 0x0 opt: 0x0`
		`Aug 26 12:31:25 weiqing-desktop ipsec[6569]: | ***parse ISAKMP Notification Payload:`
		`Aug 26 12:31:25 weiqing-desktop ipsec[6569]: | next payload type: ISAKMP_NEXT_NONE`
		`Aug 26 12:31:25 weiqing-desktop ipsec[6569]: | length: 96`
		`Aug 26 12:31:25 weiqing-desktop ipsec[6569]: | DOI: ISAKMP_DOI_IPSEC`
		`Aug 26 12:31:25 weiqing-desktop ipsec[6569]: | protocol ID: 0`
		`Aug 26 12:31:25 weiqing-desktop ipsec[6569]: | SPI size: 0`
		`Aug 26 12:31:25 weiqing-desktop ipsec[6569]: | Notify Message Type: NO_PROPOSAL_CHOSEN`
		`Aug 26 12:31:25 weiqing-desktop ipsec[6569]: "s1-c1" #1: ignoring informational payload, type NO_PROPOSAL_CHOSEN msgid=00000000`
		VMW_ORCT_CSCO_. at 799-801

No.	'111 Patent Claim 15	VMware NSX
		Using Traceflow in NSX Manager's Logical Port Inspector To use Traceflow in NSX Manager: 1. Open the Logical Switch Port inspector page for the port where the test packet will originate. (Network Components: Logical Switch Ports) 2. In the Traceflow section of the page, click the **Inject** button.

3. In the Inject Packet window, choose the **Source MAC** and **Destination MAC** addresses. To specify a port, provide its MAC address in one of the following ways:
   - Uncheck the **Filter** option and type the MAC address (see MAC address input formats (page 76).
   - Check the Filter option and choose on **Current Logical Switch** to see the list of local ports, or choose **by Hypervisor** and specify a hypervisor whose ports you want to choose from. Select the MAC address. You can further filter the list by typing a partial, *colon-separated* MAC address and pressing the up/down keys on your keyboard.



4. Optionally, you may provide a **Payload** for the packet. If you do not provide a payload, Traceflow inserts a default payload. Traceflow test packets always include the Ethernet header that Traceflow constructs based on your input. Using the Payload field, you can add other headers such as such as IP headers, TCP headers, and so on, as well as payload data.
5. Choose the **Frame Size**, taking care to leave sufficient space for all headers and the payload you are sending. (Traceflow appends a few bytes of metadata at the end of the payload.)
6. Specify the **Timeout** in milliseconds (min. 1000 ms; max. 10000 ms). This specifies how long Traceflow will wait, in total, to observe whether the packet has been delivered to all destinations in the NSX network.
7. Click **Save** to run the test. Results are returned only upon completion of the timeout period, and not immediately upon each delivery as would be the case with a tool such as ping. Traceflow reports its observations as of the end of the timeout period, and no subsequent observations will be returned.

The results appear in the Traceflow section of the window. Click the triangle to expand the results. See Interpreting Traceflow Results (page 335) for help understanding the results.

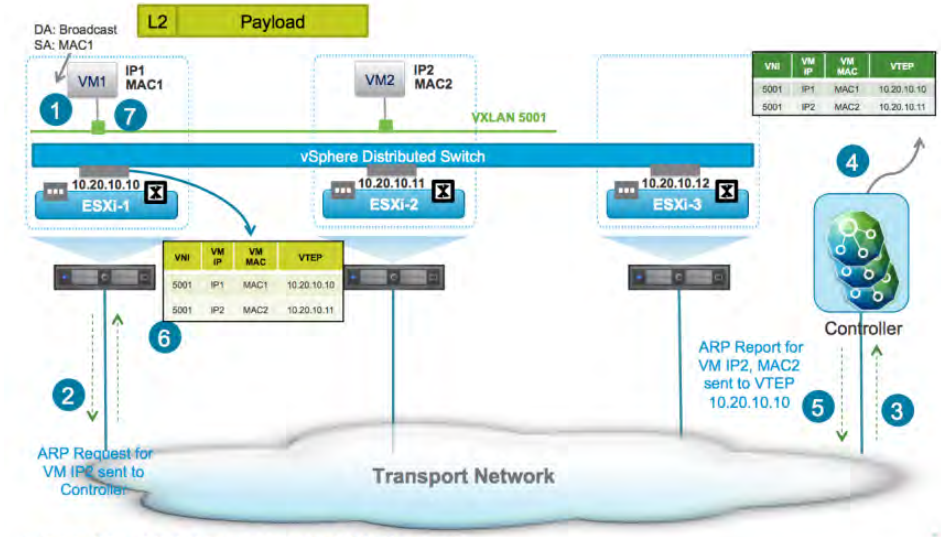
The API provides more fields that let you specify the test packet's contents in finer detail. See the next section, Using Traceflow via the NVP API.

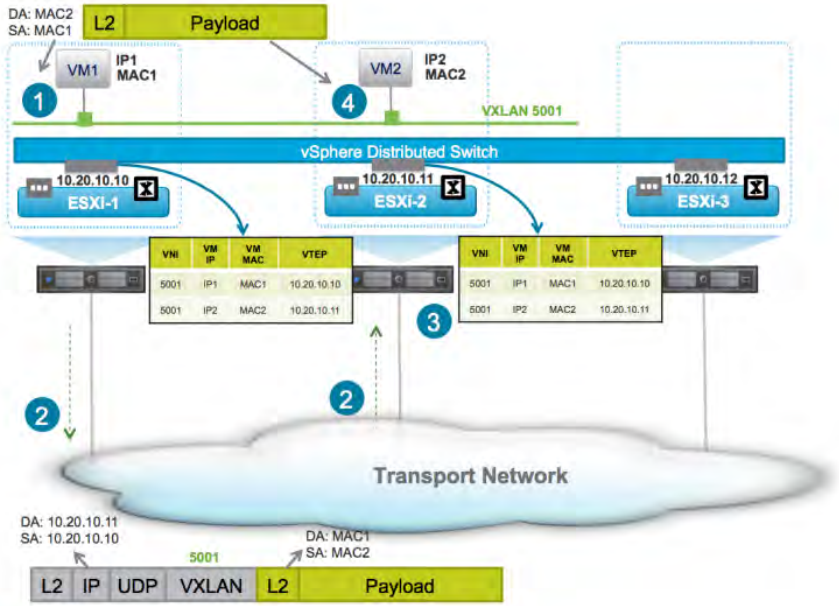
Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 1991 of 2137

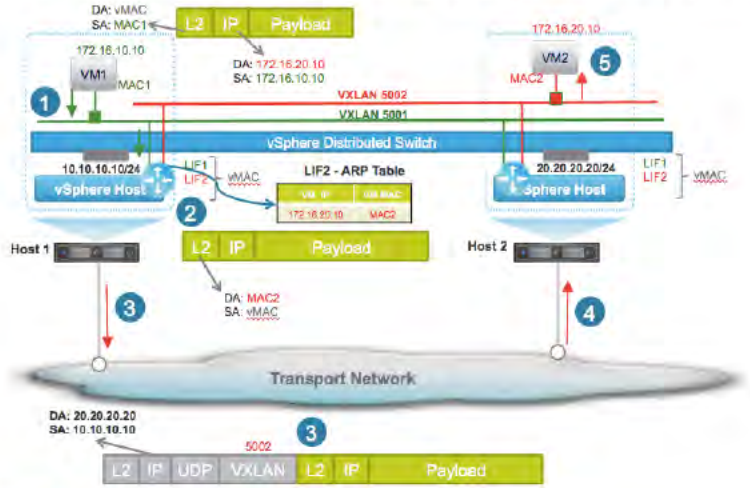
No.	'111 Patent Claim 15	VMware NSX
		Using Traceflow via the NVP API Below we describe using Traceflow via the NVP API, as viewed in the NSX Manager: 1. In NSX Manager, open the **API Inspector**. 2. Select the command, **Execute traceflow**, which runs the API command `POST /ws.v1/lswitch/<lswitch-uuid>/lport/<lport-uuid>/traceflow` 3. Specify the test packet's field values. These fields are described in more detail in the preceding section, Using Traceflow in NSX Manager's Port Connection Page (page 333). The fields: – source logical switch and source logical port in the **Lswitch UUID** (lswitch-uuid) and **Lport UUID** (lport-uuid) fields – (optional) *True* or *False* for **Ignore Cluster Majority** (_ignore_cluster_majority). This setting allows the call to proceed even if the Controller that fields the request is not connected to a Controller Cluster majority. – the destination port in the **Dst MAC** (dst_mac) field. – the Ethertype of the packet in **Eth Type** (eth_type) – the **Frame Size** (frame_size) – optional user-specified data in the **Payload** (payload) field – the originating MAC address in **Src MAC** (src_mac) – (optional) the **Src Transport Node UUID** (src_transport_node_uuid) field specifies the ID of the transport node backing the logical port that will send the test packet. This is useful in cases in which a logical port is backed by multiple transport node ports (that is, ports on separate transport nodes). For example, two or more Service Nodes are typically deployed to provide a high-availability MDI connection. In such cases, you can specify the desired transport node's ID here. If you leave this field empty, Traceflow chooses a port at random. – the **Timeout** (timeout) in milliseconds (min. 1000 ms; max. 10000 ms). This specifies how long Traceflow will wait, in total, to observe whether the packet has been delivered to all destinations in the NSX network. 4. Click **Submit Request** to run the test. See Interpreting Traceflow Results (page 335) for help understanding the results.

No.	'111 Patent Claim 15	VMware NSX
15[b]	wherein the analyzing comprises checking part of, or whole of, the payload field.	VMware NSX System discloses wherein the analyzing comprises checking part of, or whole of, the payload field. For example, VMware NSX System discloses using a deep packet inspection firewall to analyze web traffic at the L4-L7 payload layers. NSX Design Guide at 29

No.	'111 Patent Claim 15	VMware NSX
		Traffic redirection to third party vendor provides the capability to steer a particular type of traffic to a selected partner services VM. For instance, web traffic from the Internet to a set of Apache or Tomcat servers can be redirected to an L4-L7 deep packet inspection firewall for advanced protection. Traffic redirection is defined under Service Composer/Security Policy for NSX version 6.0 or under Partner Security Services tab of the DFW menu in NSX version 6.1. The Partner Security Services tab provides a powerful and easy to use user interface to define what type of traffic needs to be redirected to which partner services. It follows the same policy definition construct as DFW, providing the same options for source field, destination field and services field. The only difference is in the action field; instead of Block/Allow/Reject, a user can select between redirect/no redirect followed by a partner list. Any partner that has been registered with NSX can be successfully deployed on the platform. Additionally, log options can be enabled for this traffic redirection rule. The DFW instance on an ESXi host contains 2 separate tables. The rule table is used to store all policy rules, while the connection tracker table caches flow entries for rules with permit actions. One DFW instance is permitted per VM vNIC. A specific flow is identified by the 5-tuple information consisting source IP address, destination IP address, protocols, L4 source port, and L4 destination port fields. By default, DFW does not perform a lookup on L4 source port, but it can be configured to do so by defining a specific policy rule. NSX Design Guide at 43-44

No.	'111 Patent Claim 15	VMware NSX
		Figure 35 shows how ARP resolution is performed leveraging the control plane with the NSX controller when there is a need to establish unicast communication between virtual machines belonging to the same logical switch (i.e., VXLAN segment). 43  Figure 35 – ARP Resolution via NSX Controller NSX Design Guide at 45

No.	'111 Patent Claim 15	VMware NSX
		 Figure 36 - Intra Logical Switch Unicast Communication 1. VM1 generates a data packet directed to VM2. 2. ESXi-1 learned about the location of VM2 from the ARP report received from the controller through control plane learning. It encapsulates the packet originated by VM1 in a VXLAN packet destined to the VTEP of ESXi2 – 10.20.10.11. 3. ESXi-2 receives the packet and leverages the information in the external IP header to learn about the location of VM1, associating VM1 MAC and IP addresses to the VTEP of ESXi-1. 4. The frame is delivered to VM2. NSX Design Guide at 54

No.	'111 Patent Claim 15	VMware NSX
		The required steps to establish routed communication between two virtual machines connected to separate logical segments are shown in Figure 46.  Figure 46 - Routed Communication between Virtual Machines 1. VM1 sends a packet to VM2 connected to a different VXLAN segment. The packet is sent to the VM1 default gateway interfaces located on the local DLR (172.16.10.1). 2. A routing lookup is performed on the local DLR, which determines that the destination subnet is directly connected to DLR LIF2. A lookup is performed in the LIF2 ARP table to determine the MAC address associated with the VM2 IP address. If the ARP information is not available, the DLR will generate an ARP request on VXLAN 5002 to determine the required mapping information. 3. An L2 lookup is performed in the local MAC table to determine how to reach VM2. The original packet is VXLAN encapsulated and sent to the VTEP of ESXi2 (20.20.20.20). 4. ESXi-2 decapsulates the packet and performs an L2 lookup in the local MAC table associated to VXLAN 5002 segment. 5. The packet is delivered to the destination VM2. VMW_ORCT_CSCO_2383-84

No.	'111 Patent Claim 15	VMware NSX

		Phase 1 Policy Not Matching
		The following lists Phase 1 Policy Not Matching Error logs.
		NSX Edge
		NSX Edge hangs in STATE_MAIN_I1 state. Look in /var/log/messages for information showing that the peer sent back an IKE message with "NO_PROPOSAL_CHOSEN" set.
		`000 #1: "s1-c1":500 STATE_MAIN_I1 (sent MI1, expecting MR1); EVENT_RETRANSMIT in 7s; nodpd; idle; import:admin initiate`
		`000 #1: pending Phase 2 for "s1-c1" replacing #0`
		`Aug 26 12:31:25 weiqing-desktop ipsec[6569]: | got payload 0x800(ISAKMP_NEXT_N) needed: 0x0 opt: 0x0`
		`Aug 26 12:31:25 weiqing-desktop ipsec[6569]: | ***parse ISAKMP Notification Payload:`
		`Aug 26 12:31:25 weiqing-desktop ipsec[6569]: | next payload type: ISAKMP_NEXT_NONE`
		`Aug 26 12:31:25 weiqing-desktop ipsec[6569]: | length: 96`
		`Aug 26 12:31:25 weiqing-desktop ipsec[6569]: | DOI: ISAKMP_DOI_IPSEC`
		`Aug 26 12:31:25 weiqing-desktop ipsec[6569]: | protocol ID: 0`
		`Aug 26 12:31:25 weiqing-desktop ipsec[6569]: | SPI size: 0`
		`Aug 26 12:31:25 weiqing-desktop ipsec[6569]: | Notify Message Type: NO_PROPOSAL_CHOSEN`
		`Aug 26 12:31:25 weiqing-desktop ipsec[6569]: "s1-c1" #1: ignoring informational payload, type NO_PROPOSAL_CHOSEN msgid=00000000`

No.	'111 Patent Claim 16	VMware NSX
16[a]	The method according to claim 1, wherein the packet comprises distinct header and payload fields,	VMware NSX System discloses the method according to claim 1, wherein the packet comprises distinct header and payload fields. *See supra* at Claim 1, 15[a].
16[b]	the header comprises one or more flag bits, and	VMware NSX System discloses the header comprises one or more flag bits. For example, VMware NSX System discloses packets with specific bit sets in the header. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the VMware NSX 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. NSX Design Guide at 39

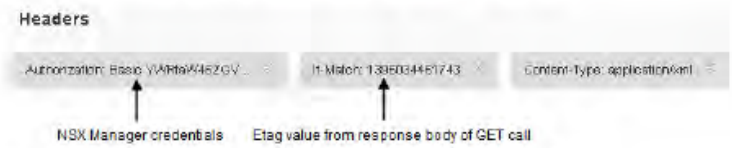
No.	'111 Patent Claim 16	VMware NSX
		Figure 30 illustrates the unicast mode replication mechanism. This process is defined as follows: 1. VM1 generates a BUM frame to be sent to each VM connected to Logical Switch 5001. In this instance there is no need to specify a multicast group associated to this VXLAN segment. 2. ESXi1 references its local VTEP table. This table is filled with the information received via control plane communication with the controller nodes. The check validates the need to replicate the packet to the other VTEP belonging to the local segment, ESXi2, as well as to the UTEP part of remote segments, ESXi3. The unicast copy sent to the UTEP has a specific bit set in the VXLAN header – the "REPLICATE_LOCALLY" bit – as an indication to the UTEP that this frame is coming from a remote VTEP segment and may need to be locally replicated. 3. The UTEP receives the frame, references the local VTEP table, and replicates it to all the ESXi hosts which are part of the local VTEP segment with at least one VM connected to VXLAN 5001. In this example, that is simply ESXi-4. NSX Design Guide at 41 3. At the same time ESXi-1 looks at the local VTEP table and determines the need to replicate the packet to the MTEP part of remote segments, in this case ESXi3. The unicast copy is sent to the MTEP with the corresponding bit set in the VXLAN header as an indication to the MTEP that this frame is coming from a remote VTEP segment and needs to be locally re-injected in the network. 4. The MTEP creates a multicast packet and sends it to the physical network where will be replicated by the local L2 switching infrastructure. VMW_ORCT_CSCO_ 3709 (NSX CLI Commands)

No.	'111 Patent Claim 16	VMware NSX
		Example vShield# show arp IP address HW type Flags HW address Mask Device 192.0.2.130 0x1 0x6 00:00:00:00:00:81 * virteth1 192.168.110.1 0x1 0x2 00:0F:90:D5:36:C1 * mgmt VMW_ORCT_CSCO_3839 **Delete a MACset** You can delete a MACset. The trailing boolean flag indicates forced or unforced delete. With forced delete, the object is deleted even if used in other places such as firewall rules, causing invalid referrals. For unforced delete, the object is deleted only if it is not used by other configuration; otherwise the delete fails. **Example 4-22.** Delete a MACset Request: DELETE https://<nsxmgr-ip>/api/2.0/services/macset/<macset-id> VMW_ORCT_CSCO_4070-71

No.	'111 Patent Claim 16	VMware NSX
		Description of Tags This section describes the tags specific to Service Composer management. **Common Tags** ■ executionOrderCategory - Category to which the action belongs to (endpoint, firewall or traffic_steering) ■ actionType - Defines the type of action belonging to a given executionOrderCategory ■ isEnabled - Indicates whether an action is enabled ■ isActionEnforced - Enforces an action of a parent policy on its child policies for a given actionType and executionOrderCategory. Note that in a policy hierarchy, for a given actionType and executionOrderCategory, there can be only one action which can be marked as enforced. ■ isActive - In a security policy hierarchy, an action within a policy may or may not be active based on the precedence of the policy or usage of isActionEnforced flag in that hierarchy ■ securityPolicy - Parent policy in an action ■ secondarySecurityGroup - Applicable for actions which need secondary security groups, say a source-destination firewall rule **Output only Tags** ■ executionOrder - Defines the sequence in which actions belonging to an executionOrderCategory are executed. Note that this is not an input parameter and its value is implied by the index in the list.

No.	'111 Patent Claim 16	VMware NSX
		Firewall Category Tags ■ applications - Applications / application groups on which the rules areto be applied ■ logged - Flag to enable logging of the traffic that is hit by this rule ■ action - Allow or block the traffic ■ direction - Direction of traffic towards primary security group. Possible values: inbound, outbound, intra ■ outsideSecondaryContainer - Flag to specify outside i.e. outside securitygroup-3 **Endpoint Category Tags** ■ serviceId - ID of the service(as registered with the service insertion module). If this tag is null, the functionality type (as defined in actionType tag) is not applied which will also result in blocking the actions(of given functionality type) that are inherited from the parent seicrity policy. This is true if there is no action of enfore type. ■ vendorTemplateId - ID of specific vendor configuration. ■ invalidServiceId - Flag to indicate that the service that was referenced in this rule is deleted, which make the rule ineffective(or deviate from the original intent that existed while configuring the rule). You must either modify this rule by adding correct Service or delete this rule. ■ invalidVendorTemplateId - Flag to indicate that the vendor template that was referenced in this rule is deleted, which make the rule ineffective(or deviate from the original intent that existed while configuring the rule). You must either fix this rule by adding correct Service or delete this rule. ■ serviceName -Name of the service ■ vendorTemplateName - Name of vendor template **TrafficSteering/NetX Category Tags** ■ redirect - Flag to indicate whether to redirect the traffic or not ■ serviceProfile - Service profile for which redirection is being configured ■ logged - Flag to enable logging of the traffic that is hit by this rule VMW_ORCT_CSCO_1444

No.	'111 Patent Claim 16	VMware NSX
		Table 9-2. Extensions for HTTP/HTTPS Protocol **Monitor Extension** / **Description** no-body — Do not wait for document body: stop reading after headers. Note that this still does an HTTP GET or POST, not a HEAD. max-age=*SECONDS* — Warn if document is more than SECONDS old. The number can also be in the form 10m for minutes, 10h for hours, or 10d for days. content-type=*STRING* — specify Content-Type header media type in POST calls. VMW_ORCT_CSCO_2035-36 **Modify Firewall Configuration** Follow the procedure below to modify the firewall configuration. 1 Run a GET call for the firewall configuration. 2 Extract the XML from the response body of the GET call and modify it as required. 3 From the Response Header in Step 1, copy the Etag header value.

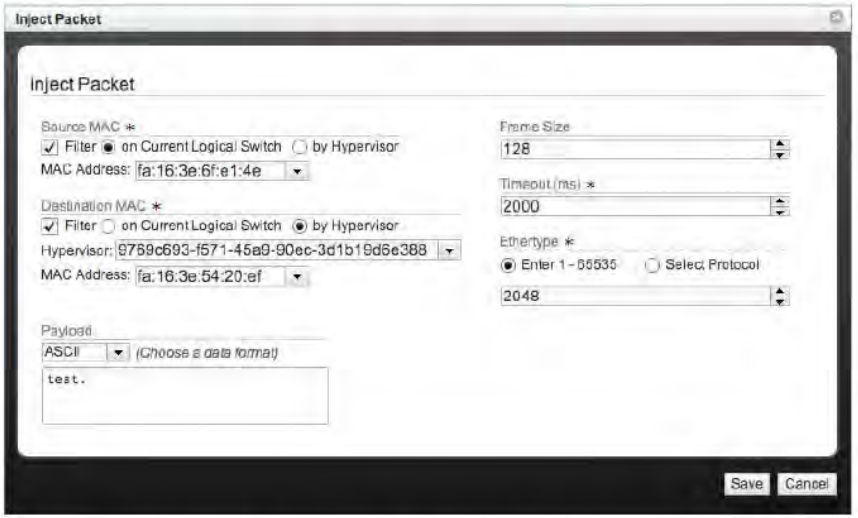
No.	'111 Patent Claim 16	VMware NSX
		 4 Add the number as the If-Match header in the PUT call.  5 Pass the modified XML as the Request Body in a PUT call. ■ Not all fields are required while sending the request. Refer to the optional field in the schema definition of various objects. All the optional fields are safe to be ignored while sending the configuration to server. For example, if an IP Set is referenced in the rule only IPSet and Type is needed in the Source/Destination objects and not Name and isValid tags. ■ IDs for new objects (rule/section) should be removed or set to zero. ■ If new entities (sections/rules) have been sent in the request, the response will contain the system-generated ids, which are assigned to these new entities. These ID identifies the resource and can be used in the urls if you want to operate on these entities using those URLs. VMW_ORCT_CSCO_2383-84

No.	'111 Patent Claim 16	VMware NSX

		Phase 1 Policy Not Matching
		The following lists Phase 1 Policy Not Matching Error logs.
		NSX Edge
		NSX Edge hangs in STATE_MAIN_I1 state. Look in /var/log/messages for information showing that the peer sent back an IKE message with "NO_PROPOSAL_CHOSEN" set.
		`000 #1: "s1-c1":500 STATE_MAIN_I1 (sent MI1, expecting MR1); EVENT_RETRANSMIT in 7s; nodpd; idle; import:admin initiate`
		`000 #1: pending Phase 2 for "s1-c1" replacing #0`
		`Aug 26 12:31:25 weiqing-desktop ipsec[6569]: | got payload 0x800(ISAKMP_NEXT_N) needed: 0x0 opt: 0x0`
		`Aug 26 12:31:25 weiqing-desktop ipsec[6569]: | ***parse ISAKMP Notification Payload:`
		`Aug 26 12:31:25 weiqing-desktop ipsec[6569]: | next payload type: ISAKMP_NEXT_NONE`
		`Aug 26 12:31:25 weiqing-desktop ipsec[6569]: | length: 96`
		`Aug 26 12:31:25 weiqing-desktop ipsec[6569]: | DOI: ISAKMP_DOI_IPSEC`
		`Aug 26 12:31:25 weiqing-desktop ipsec[6569]: | protocol ID: 0`
		`Aug 26 12:31:25 weiqing-desktop ipsec[6569]: | SPI size: 0`
		`Aug 26 12:31:25 weiqing-desktop ipsec[6569]: | Notify Message Type: NO_PROPOSAL_CHOSEN`
		`Aug 26 12:31:25 weiqing-desktop ipsec[6569]: "s1-c1" #1: ignoring informational payload, type NO_PROPOSAL_CHOSEN msgid=00000000`
		VMW_ORCT_CSCO_ 799-801

No.	'111 Patent Claim 16	VMware NSX
		Using Traceflow in NSX Manager's Logical Port Inspector To use Traceflow in NSX Manager: 1. Open the Logical Switch Port inspector page for the port where the test packet will originate. (Network Components: Logical Switch Ports) 2. In the Traceflow section of the page, click the **Inject** button.

3. In the Inject Packet window, choose the **Source MAC** and **Destination MAC** addresses. To specify a port, provide its MAC address in one of the following ways:
   - Uncheck the **Filter** option and type the MAC address (see MAC address input formats (page 76).
   - Check the Filter option and choose on **Current Logical Switch** to see the list of local ports, or choose **by Hypervisor** and specify a hypervisor whose ports you want to choose from. Select the MAC address. You can further filter the list by typing a partial, *colon-separated* MAC address and pressing the up/down keys on your keyboard.



4. Optionally, you may provide a **Payload** for the packet. If you do not provide a payload, Traceflow inserts a default payload. Traceflow test packets always include the Ethernet header that Traceflow constructs based on your input. Using the Payload field, you can add other headers such as such as IP headers, TCP headers, and so on, as well as payload data.
5. Choose the **Frame Size**, taking care to leave sufficient space for all headers and the payload you are sending. (Traceflow appends a few bytes of metadata at the end of the payload.)
6. Specify the **Timeout** in milliseconds (min. 1000 ms; max. 10000 ms). This specifies how long Traceflow will wait, in total, to observe whether the packet has been delivered to all destinations in the NSX network.
7. Click **Save** to run the test. Results are returned only upon completion of the timeout period, and not immediately upon each delivery as would be the case with a tool such as ping. Traceflow reports its observations as of the end of the timeout period, and no subsequent observations will be returned.

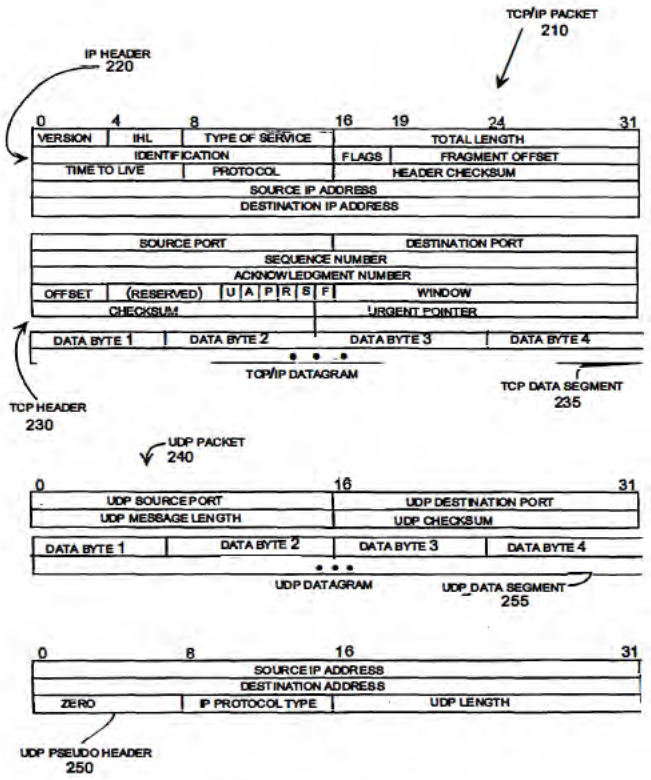
The results appear in the Traceflow section of the window. Click the triangle to expand the results. See Interpreting Traceflow Results (page 335) for help understanding the results.

The API provides more fields that let you specify the test packet's contents in finer detail. See the next section, Using Traceflow via the NVP API.

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 2008 of 2137

No.	'111 Patent Claim 16	VMware NSX
		Using Traceflow via the NVP API Below we describe using Traceflow via the NVP API, as viewed in the NSX Manager: 1. In NSX Manager, open the **API Inspector**. 2. Select the command, **Execute traceflow**, which runs the API command `POST /ws.v1/lswitch/<lswitch-uuid>/lport/<lport-uuid>/traceflow` 3. Specify the test packet's field values. These fields are described in more detail in the preceding section, Using Traceflow in NSX Manager's Port Connection Page (page 333). The fields: – source logical switch and source logical port in the **Lswitch UUID** (lswitch-uuid) and **Lport UUID** (lport-uuid) fields – (optional) *True* or *False* for **Ignore Cluster Majority** (_ignore_cluster_majority). This setting allows the call to proceed even if the Controller that fields the request is not connected to a Controller Cluster majority. – the destination port in the **Dst MAC** (dst_mac) field. – the Ethertype of the packet in **Eth Type** (eth_type) – the **Frame Size** (frame_size) – optional user-specified data in the **Payload** (payload) field – the originating MAC address in **Src MAC** (src_mac) – (optional) the **Src Transport Node UUID** (src_transport_node_uuid) field specifies the ID of the transport node backing the logical port that will send the test packet. This is useful in cases in which a logical port is backed by multiple transport node ports (that is, ports on separate transport nodes). For example, two or more Service Nodes are typically deployed to provide a high-availability MDI connection. In such cases, you can specify the desired transport node's ID here. If you leave this field empty, Traceflow chooses a port at random. – the **Timeout** (timeout) in milliseconds (min. 1000 ms; max. 10000 ms). This specifies how long Traceflow will wait, in total, to observe whether the packet has been delivered to all destinations in the NSX network. 4. Click **Submit Request** to run the test. See Interpreting Traceflow Results (page 335) for help understanding the results. Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, 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

No.	'111 Patent Claim 16	VMware NSX
		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. For example, Copeland discloses packet headers with flag bits. Copeland at Figure 2

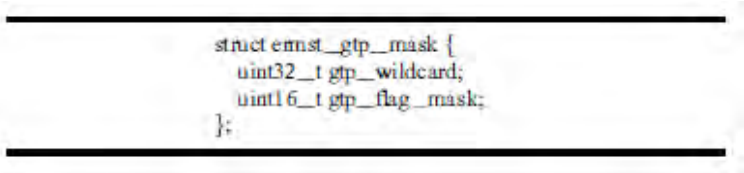
No.	'111 Patent Claim 16	VMware NSX
		 FIG. 2 Copeland at [0076] ("FIG. 2 illustrates an exemplary TCP/IP packet or datagram 210 and an exemplary UDP datagram 240. In a typical TCP/IP packet like 210, each packet typically includes a header portion comprising an IP header 220 and a TCP header 230, followed 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

No.	'111 Patent Claim 16	VMware NSX
		(TCP) header follows the Internet protocol header and specifies the port numbers for the associated service.") 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.") 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.") 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 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	VMware NSX
		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.") 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.") As another example, Kempf discloses packet headers with flag bits. 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.")

No.	'111 Patent Claim 16	VMware NSX
		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.") 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.") 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.") 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

No.	'111 Patent Claim 16	VMware NSX
		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.") 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).") 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.") 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.

No.	'111 Patent Claim 16	VMware NSX
		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.") 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:  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.	VMware NSX System discloses wherein the packet-applicable criterion is that one or more of the flag bits is set.

No.	'111 Patent Claim 16	VMware NSX
		For example, VMware NSX System discloses using the specific bit set in the header to determine the routing and replication of the packet in accordance with a rule. Thus, at least under the apparent claim scope alleged by Orckit's Infringement Disclosures, this limitation is met. To the extent that the VMware NSX System 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. NSX Design Guide at 39 Figure 30 illustrates the unicast mode replication mechanism. This process is defined as follows: 1. VM1 generates a BUM frame to be sent to each VM connected to Logical Switch 5001. In this instance there is no need to specify a multicast group associated to this VXLAN segment. 2. ESXi1 references its local VTEP table. This table is filled with the information received via control plane communication with the controller nodes. The check validates the need to replicate the packet to the other VTEP belonging to the local segment, ESXi2, as well as to the UTEP part of remote segments, ESXi3. The unicast copy sent to the UTEP has a specific bit set in the VXLAN header – the "REPLICATE_LOCALLY" bit – as an indication to the UTEP that this frame is coming from a remote VTEP segment and may need to be locally replicated. 3. The UTEP receives the frame, references the local VTEP table, and replicates it to all the ESXi hosts which are part of the local VTEP segment with at least one VM connected to VXLAN 5001. In this example, that is simply ESXi-4. NSX Design Guide at 41

No.	'111 Patent Claim 16	VMware NSX
		3. At the same time ESXi-1 looks at the local VTEP table and determines the need to replicate the packet to the MTEP part of remote segments, in this case ESXi3. The unicast copy is sent to the MTEP with the corresponding bit set in the VXLAN header as an indication to the MTEP that this frame is coming from a remote VTEP segment and needs to be locally re-injected in the network. 4. The MTEP creates a multicast packet and sends it to the physical network where will be replicated by the local L2 switching infrastructure. VMW_ORCT_CSCO_ 2205-2206 **Distributed Firewall** Distributed firewall is a hypervisor kernel-embedded firewall that provides visibility and control for virtualized workloads and networks. You can create access control policies based on VMware vCenter objects like datacenters and clusters and virtual machine names; network constructs like IP or IPSet addresses, VLAN (DVS port-groups), VXLAN (logical switches), security groups, as well as user group identity from Active Directory. Firewall rules are enforced at the vNIC level of each virtual machine to provide consistent access control even when the virtual machine gets vMotioned. The hypervisor-embedde nature of the firewall delivers close to line rate throughput to enable higher workload consolidation on physical servers. The distributed nature of the firewall provides a scale-out architecture that automatically extends firewall capacity when additional hosts are added to a datacenter. For L2 packets, distributed firewall creates a cache for performance boost. L3 packets are processed in the following sequence: 1 All packets are checked for an existing state. This is done for SYNs too so that bogus or retransmitted SYNs for existing sessions can be detected. 2 If a state match is found, the packets are processed. 3 If a state match is not found, the packet is processed through the rules until a match is found. ■ For TCP packets, a state is set only for packets with a SYN flag. However, rules that do not specify a protocol (service ANY), can match TCP packets with any combination of flags. ■ For UDP packets, 5-tuple details are extracted from the packet. If a state does not exist in the state table, a new state is created using the extracted 5-tuple details. Subsequently received packets are matched against the state that was just created. ■ For ICMP packets, ICMP type, code, and packet direction are used to create a state.

No.	'111 Patent Claim 16	VMware NSX
		VMW_ORCT_CSCO_4068-71

Working with Security Policies

A security policy is a set of Endpoint, firewall, and network introspection services that can be applied to a security group.

For information on creating a security group, see "Working with Security Groups" on page 53.

Creating a Security Policy

When creating a security policy, a parent security policy can be specified if required. The security policy inherits services from the parent security policy. Security group bindings and actions can also be specified while creating the policy. Note that execution order of actions in a category is implied by their order in the list. The response of the call has Location header populated with the URI using which the created object can be fetched.

Prerequisites

Ensure that:

- the required VMware built in services (such as Distributed Firewall, Data Security, and Endpoint) are installed. See *NSX Installation and Upgrade Guide*.
- the required partner services have been registered with NSX Manager.
- the required security groups have been created.

No.	'111 Patent Claim 16	VMware NSX
		Description of Tags This section describes the tags specific to Service Composer management. **Common Tags** ■ executionOrderCategory - Category to which the action belongs to (endpoint, firewall or traffic_steering) ■ actionType - Defines the type of action belonging to a given executionOrderCategory ■ isEnabled - Indicates whether an action is enabled ■ isActionEnforced - Enforces an action of a parent policy on its child policies for a given actionType and executionOrderCategory. Note that in a policy hierarchy, for a given actionType and executionOrderCategory, there can be only one action which can be marked as enforced. ■ isActive - In a security policy hierarchy, an action within a policy may or may not be active based on the precedence of the policy or usage of isActionEnforced flag in that hierarchy ■ securityPolicy - Parent policy in an action ■ secondarySecurityGroup - Applicable for actions which need secondary security groups, say a source-destination firewall rule **Output only Tags** ■ executionOrder - Defines the sequence in which actions belonging to an executionOrderCategory are executed. Note that this is not an input parameter and its value is implied by the index in the list.

No.	'111 Patent Claim 16	VMware NSX
		Firewall Category Tags ■ applications - Applications / application groups on which the rules areto be applied ■ logged - Flag to enable logging of the traffic that is hit by this rule ■ action - Allow or block the traffic ■ direction - Direction of traffic towards primary security group. Possible values: inbound, outbound, intra ■ outsideSecondaryContainer - Flag to specify outside i.e. outside securitygroup-3 **Endpoint Category Tags** ■ serviceId - ID of the service(as registered with the service insertion module). If this tag is null, the functionality type (as defined in actionType tag) is not applied which will also result in blocking the actions(of given functionality type) that are inherited from the parent seicrity policy. This is true if there is no action of enfore type. ■ vendorTemplateId - ID of specific vendor configuration. ■ invalidServiceId - Flag to indicate that the service that was referenced in this rule is deleted, which make the rule ineffective(or deviate from the original intent that existed while configuring the rule). You must either modify this rule by adding correct Service or delete this rule. ■ invalidVendorTemplateId - Flag to indicate that the vendor template that was referenced in this rule is deleted, which make the rule ineffective(or deviate from the original intent that existed while configuring the rule). You must either fix this rule by adding correct Service or delete this rule. ■ serviceName -Name of the service ■ vendorTemplateName - Name of vendor template **TrafficSteering/NetX Category Tags** ■ redirect - Flag to indicate whether to redirect the traffic or not ■ serviceProfile - Service profile for which redirection is being configured ■ logged - Flag to enable logging of the traffic that is hit by this rule VMW_ORCT_CSCO_2067

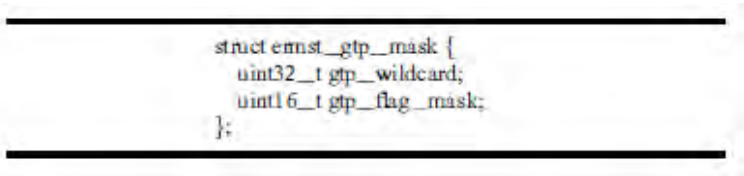
No.	'111 Patent Claim 16	VMware NSX
		Service Composer helps you provision and assign network and security services to applications in a virtual infrastructure. You map these services to a security group, and the services are applied to the virtual machines in the security group. **Security Group** You begin by creating a security group to define assets that you want to protect. Security groups may be static (including specific virtual machines) or dynamic where membership may be defined in one or more of the following ways: ■ vCenter containers (clusters, port groups, or datacenters) ■ Security tags, IPset, MACset, or even other security groups. For example, you may include a criteria to add all members tagged with the specified security tag (such as AntiVirus.virusFound) to the security group. ■ Directory Groups (if NSX Manager is registered with Active Directory) ■ Regular expressions such as virtual machines with name *VM1* Note that security group membership changes constantly. For example, a virtual machine tagged with the AntiVirus.virusFound tag is moved into the Quarantine security group. When the virus is cleaned and this tag is removed from the virtual machine, it again moves out of the Quarantine security group. **Security Policy** A security policy is a collection of the following service configurations. **Table 10-1.** Security services contained in a security policy **Service** | **Description** | **Applies to** Firewall rules | Rules that define the traffic to be allowed to, from, or within the security group. | vNIC Endpoint service | Data Security or third party solution provider services such as anti-virus or vulnerability management services. | virtual machines Network introspection services | Services that monitor your network such as IPS. | virtual machines

No.	'111 Patent Claim 16	VMware NSX
		Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, 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 16[c] of Exhibit E-4 renders the claim, including the present limitation, obvious. Below are examples of two such references. For example, Copeland discloses packet specific characteristics including flag bits that are set. 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 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.") 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.")

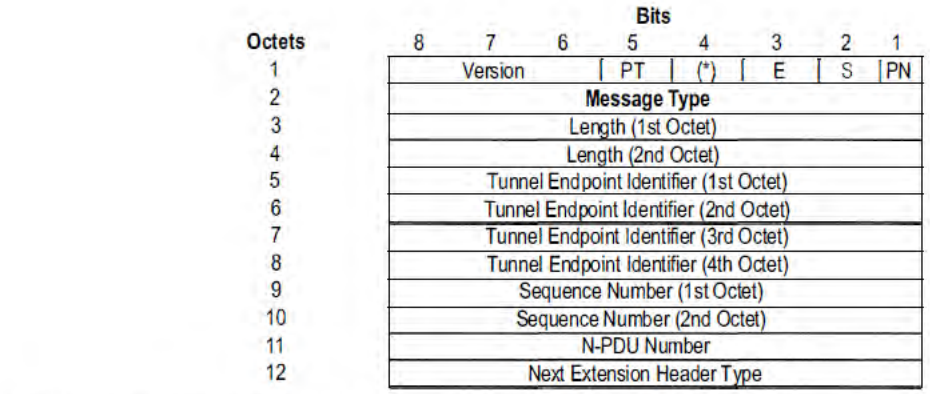
No.	'111 Patent Claim 16	VMware NSX
		Copeland at [0089] ("FIG. 3 illustrates an exemplary TCP/IP session 300. As discussed in reference to FIG. 2, the SYN flag is set whenever one host initiates a session with another host. In the initial packet, Hostl sends a message with only the SYN flag set. The 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.") 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).") 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.") As another example, Kempf flow table matches in which the flag bits is set, 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

No.	'111 Patent Claim 16	VMware NSX
		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.") 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.") 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.") 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,

No.	'111 Patent Claim 16	VMware NSX
		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.") 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.") 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).") 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

No.	'111 Patent Claim 16	VMware NSX
		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.") 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.") 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:  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

No.	'111 Patent Claim 16	VMware NSX
		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

No.	'111 Patent Claim 17	VMware NSX
17[a]	The method according to claim 16, wherein the packet is an Transmission Control Protocol (TCP) packet, and	VMware NSX System discloses the method according to claim 16, wherein the packet is an Transmission Control Protocol (TCP) packet. For example, VMware NSX System discloses packets used in TCP communication channels *See supra* at Claim 16. NSX Design Guide at 22 The user world agent process (netcpa) establishes TCP over SSL communication channels to the controller cluster nodes. Controller nodes leverage this control-plane channel with the ESXi hypervisors to populate local tables (e.g., MAC address table, ARP table, and VTEP table) to keep track of where the workloads are connected in the deployed logical networks. NSX Design Guide at 30

DFW rules are enforced as follows:

- Rules are processed in top-to-bottom ordering.
- Each packet is checked against the top rule in the rule table before moving down the subsequent rules in the table.
- The first rule in the table that matches the traffic parameters is enforced. No subsequent rules can be enforced as the search is then terminated for that packet.

Because of this behavior, it is always recommended to put the most granular policies at the top of the rule table. This will ensure they will be enforced before more specific rules.

The DFW default policy rule, located at the bottom of the rule table, is a "catch-all" rule; packets not matching any other rules will be enforced by the default rule. After the host preparation operation, the DFW default rule is set to 'allow' action. This ensures that VM-to-VM communication is not broken during staging or migration phases. It is a best practice to then change this default rule to 'block' action and enforce access control through a positive control model (i.e., only traffic defined in the firewall policy is allowed onto the network).

Figure 22 steps through policy rule lookup and packet flow:

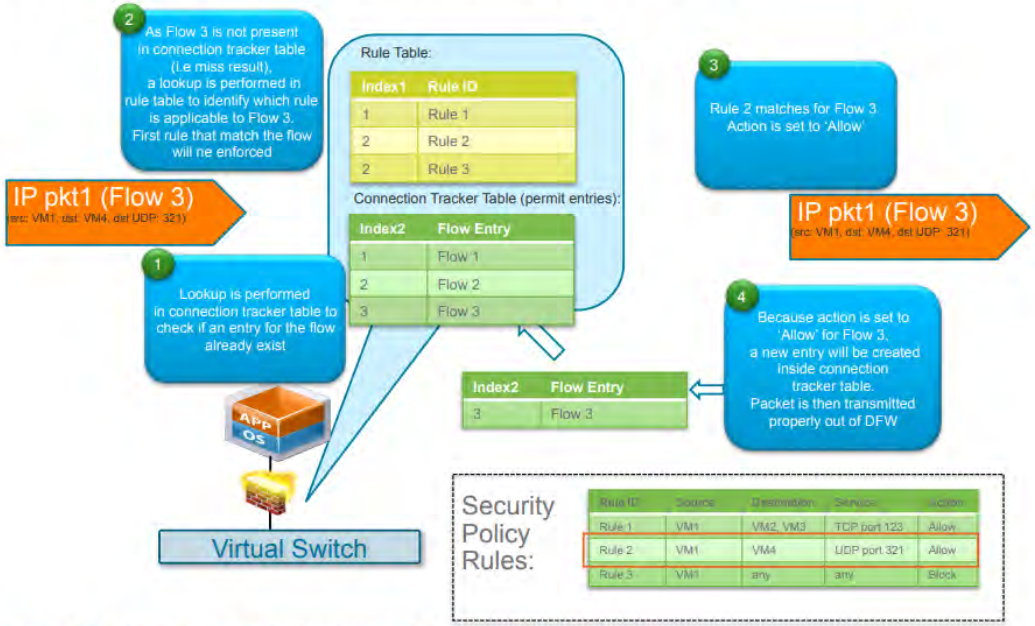


Figure 22 – DFW Policy Rule Lookup and Packet – First Packet

No.	'111 Patent Claim 17	VMware NSX
		VMW_ORCT_CSCO_ 2205-2206 **Distributed Firewall** Distributed firewall is a hypervisor kernel-embedded firewall that provides visibility and control for virtualized workloads and networks. You can create access control policies based on VMware vCenter objects like datacenters and clusters and virtual machine names; network constructs like IP or IPSet addresses, VLAN (DVS port-groups), VXLAN (logical switches), security groups, as well as user group identity from Active Directory. Firewall rules are enforced at the vNIC level of each virtual machine to provide consistent access control even when the virtual machine gets vMotioned. The hypervisor-embedde nature of the firewall delivers close to line rate throughput to enable higher workload consolidation on physical servers. The distributed nature of the firewall provides a scale-out architecture that automatically extends firewall capacity when additional hosts are added to a datacenter. For L2 packets, distributed firewall creates a cache for performance boost. L3 packets are processed in the following sequence: 1 All packets are checked for an existing state. This is done for SYNs too so that bogus or retransmitted SYNs for existing sessions can be detected. 2 If a state match is found, the packets are processed. 3 If a state match is not found, the packet is processed through the rules until a match is found. ■ For TCP packets, a state is set only for packets with a SYN flag. However, rules that do not specify a protocol (service ANY), can match TCP packets with any combination of flags. ■ For UDP packets, 5-tuple details are extracted from the packet. If a state does not exist in the state table, a new state is created using the extracted 5-tuple details. Subsequently received packets are matched against the state that was just created. ■ For ICMP packets, ICMP type, code, and packet direction are used to create a state. VMW_ORCT_CSCO_ 2212

No.	'111 Patent Claim 17	VMware NSX
		10 Point to the **Action** cell of the new rule and click ⊞. Make appropriate selections as described in the table below and click **OK**.

Action	Results in
Allow	Allows traffic from or to the specified source(s), destination(s), and service(s).
Block	Blocks traffic from or to the specified source(s), destination(s), and service(s).
Reject	Sends reject message for unaccepted packets. RST packets are sent for TCP connections. ICMP messages with administratively prohibited code are sent for UDP, ICMP, and other IP connections.
Log	Logs all sessions matching this rule. Enabling logging can affect performance.
Do not log	Does not log sessions.

VMW_ORCT_CSCO_ 2231

No.	'111 Patent Claim 17	VMware NSX
		Add a Private Network Add the network that you want the remote user to be able to access. **Procedure** 1 In the **SSL Vpn-Plus** tab, select **Private Networks** from the left panel. 2 Click the **Add** ( ) icon 3 Type the private network IP address. 4 Type the netmask of the private network. 5 (Optional) Type a description for the network. 6 Specify whether you want to send private network and internet traffic over the SSL VPN-Plus enabled NSX Edge or directly to the private server by bypassing the NSX Edge. 7 If you selected **Send traffic over the tunnel**, select **Enable TCP Optimization** to optimize the internet speed. Conventional full-access SSL VPNs tunnel sends TCP/IP data in a second TCP/IP stack for encryption over the internet. This results in application layer data being encapsulated twice in two separate TCP streams. When packet loss occurs (which happens even under optimal internet conditions), a performance degradation effect called TCP-over-TCP meltdown occurs. In essence, two TCP instruments are correcting a single packet of IP data, undermining network throughput and causing connection timeouts. TCP Optimization eliminates this TCP-over-TCP problem, ensuring optimal performance. VMW_ORCT_CSCO_ 2261

No.	'111 Patent Claim 17	VMware NSX
		The NSX Edge load balancer enables network traffic to follow multiple paths to a specific destination. It distributes incoming service requests evenly among multiple servers in such a way that the load distribution is transparent to users. Load balancing thus helps in achieving optimal resource utilization, maximizing throughput, minimizing response time, and avoiding overload. NSX Edge provides load balancing up to Layer 7. You map an external, or public, IP address to a set of internal servers for load balancing. The load balancer accepts TCP, HTTP, or HTTPS requests on the external IP address and decides which internal server to use. Port 8090 is the default listening port for TCP, port 80 is the default port for HTTP, and port 443 is the default port for HTTPs. You must have a working NSX Edge instance before you can load balancing. For information on setting up NSX Edge, see "NSX Edge Operations," on page 188. For information on configuring an NSX Edge certificate, see "Working with Certificates," on page 188.
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.	VMware NSX 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. For example, VMware NSX System discloses specific bits set in the packet header. A person of ordinary skill in the art would understand that this bit set could be comprised of 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 VMware NSX 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. NSX Design Guide at 39

No.	'111 Patent Claim 17	VMware NSX
		Figure 30 illustrates the unicast mode replication mechanism. This process is defined as follows: 1. VM1 generates a BUM frame to be sent to each VM connected to Logical Switch 5001. In this instance there is no need to specify a multicast group associated to this VXLAN segment. 2. ESXi1 references its local VTEP table. This table is filled with the information received via control plane communication with the controller nodes. The check validates the need to replicate the packet to the other VTEP belonging to the local segment, ESXi2, as well as to the UTEP part of remote segments, ESXi3. The unicast copy sent to the UTEP has a specific bit set in the VXLAN header – the "REPLICATE_LOCALLY" bit – as an indication to the UTEP that this frame is coming from a remote VTEP segment and may need to be locally replicated. 3. The UTEP receives the frame, references the local VTEP table, and replicates it to all the ESXi hosts which are part of the local VTEP segment with at least one VM connected to VXLAN 5001. In this example, that is simply ESXi-4. NSX Design Guide at 41 3. At the same time ESXi-1 looks at the local VTEP table and determines the need to replicate the packet to the MTEP part of remote segments, in this case ESXi3. The unicast copy is sent to the MTEP with the corresponding bit set in the VXLAN header as an indication to the MTEP that this frame is coming from a remote VTEP segment and needs to be locally re-injected in the network. 4. The MTEP creates a multicast packet and sends it to the physical network where will be replicated by the local L2 switching infrastructure. VMW_ORCT_CSCO_2205-2206

No.	'111 Patent Claim 17	VMware NSX
		Distributed Firewall Distributed firewall is a hypervisor kernel-embedded firewall that provides visibility and control for virtualized workloads and networks. You can create access control policies based on VMware vCenter objects like datacenters and clusters and virtual machine names; network constructs like IP or IPSet addresses, VLAN (DVS port-groups), VXLAN (logical switches), security groups, as well as user group identity from Active Directory. Firewall rules are enforced at the vNIC level of each virtual machine to provide consistent access control even when the virtual machine gets vMotioned. The hypervisor-embedde nature of the firewall delivers close to line rate throughput to enable higher workload consolidation on physical servers. The distributed nature of the firewall provides a scale-out architecture that automatically extends firewall capacity when additional hosts are added to a datacenter. For L2 packets, distributed firewall creates a cache for performance boost. L3 packets are processed in the following sequence: 1 All packets are checked for an existing state. This is done for SYNs too so that bogus or retransmitted SYNs for existing sessions can be detected. 2 If a state match is found, the packets are processed. 3 If a state match is not found, the packet is processed through the rules until a match is found. ■ For TCP packets, a state is set only for packets with a SYN flag. However, rules that do not specify a protocol (service ANY), can match TCP packets with any combination of flags. ■ For UDP packets, 5-tuple details are extracted from the packet. If a state does not exist in the state table, a new state is created using the extracted 5-tuple details. Subsequently received packets are matched against the state that was just created. ■ For ICMP packets, ICMP type, code, and packet direction are used to create a state. VMW_ORCT_CSCO_2212

No.	'111 Patent Claim 17	VMware NSX
		10 Point to the **Action** cell of the new rule and click ⊞. Make appropriate selections as described in the table below and click **OK**.

Action	Results in
Allow	Allows traffic from or to the specified source(s), destination(s), and service(s).
Block	Blocks traffic from or to the specified source(s), destination(s), and service(s).
Reject	Sends reject message for unaccepted packets. RST packets are sent for TCP connections. ICMP messages with administratively prohibited code are sent for UDP, ICMP, and other IP connections.
Log	Logs all sessions matching this rule. Enabling logging can affect performance.
Do not log	Does not log sessions.

Under at least the apparent claim scope alleged by Orckit's Infringement Disclosures, 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 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.")

No.	'111 Patent Claim 17	VMware NSX
		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.") 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.") 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.") 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.") 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

No.	'111 Patent Claim 17	VMware NSX
		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.") 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.") 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.") 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.") As another example, Uchida discloses the TCP (Transmission Control Protocol) FIN flag, RST flag, and SYN flag. 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

No.	'111 Patent Claim 17	VMware NSX
		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.") 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.") 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.") 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.") 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.") 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

No.	'111 Patent Claim 17	VMware NSX
		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.") 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.") 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).") 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

No.	'111 Patent Claim 17	VMware NSX
		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.") 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).") 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).") 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.") 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.

No.	'111 Patent Claim 17	VMware NSX
		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.") 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).") 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).") 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.") 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

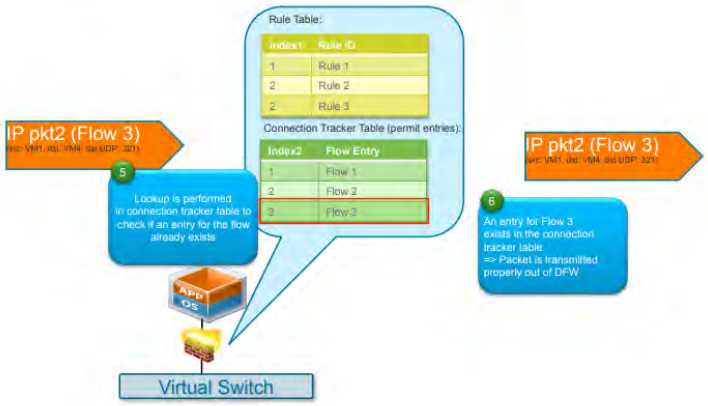
No.	'111 Patent Claim 17	VMware NSX
		storage unit C13, and marks a FIN packet reception flag (steps S204 to S205 in FIG. 8). This process corresponds to the internal state update process in step S115 in FIG. 3.") Uchida at [0134] ("Referring back to the state transition diagram of TCP connection in 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.") 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.") 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.") 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).") 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

No.	'111 Patent Claim 17	VMware NSX
		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).") 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.") 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.") 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.") 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.") 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).")

No.	'111 Patent Claim 17	VMware NSX
		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).") 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.")

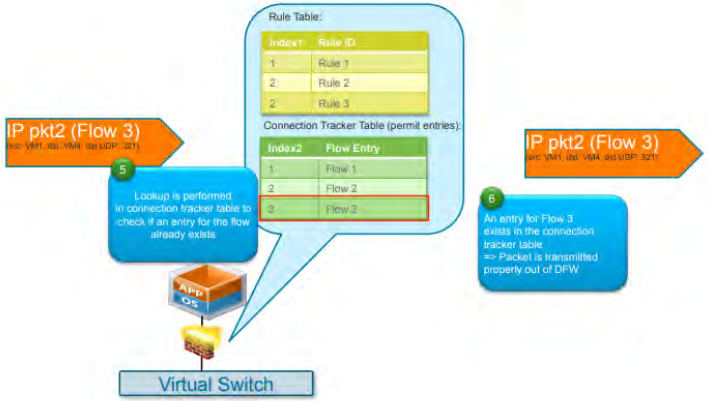
No.	'111 Patent Claim 18	VMware NSX
18[a]	The method according to claim 1, wherein the packet comprises distinct header and payload fields,	VMware NSX System discloses the method according to claim 1, wherein the packet comprises distinct header and payload fields. *See supra* at Claim 1, 15[a].
18[b]	the header comprises at least the first and second entities addresses in the packet network, and	VMware NSX System discloses the header comprises at least the first and second entities addresses in the packet network. For example, VMware NSX System discloses source and destination address in the packet header. NSX Design Guide at 20

No.	'111 Patent Claim 18	VMware NSX
		• The source and destination IP addresses used in the external IP header uniquely identify the ESXi hosts originating and terminating the VXLAN encapsulation of frames. Those are usually referred to as VXLAN Tunnel Endpoints (VTEPs). NSX Design Guide at 29 Traffic redirection to third party vendor provides the capability to steer a particular type of traffic to a selected partner services VM. For instance, web traffic from the Internet to a set of Apache or Tomcat servers can be redirected to an L4-L7 deep packet inspection firewall for advanced protection. Traffic redirection is defined under Service Composer/Security Policy for NSX version 6.0 or under Partner Security Services tab of the DFW menu in NSX version 6.1. The Partner Security Services tab provides a powerful and easy to use user interface to define what type of traffic needs to be redirected to which partner services. It follows the same policy definition construct as DFW, providing the same options for source field, destination field and services field. The only difference is in the action field; instead of Block/Allow/Reject, a user can select between redirect/no redirect followed by a partner list. Any partner that has been registered with NSX can be successfully deployed on the platform. Additionally, log options can be enabled for this traffic redirection rule. The DFW instance on an ESXi host contains 2 separate tables. The rule table is used to store all policy rules, while the connection tracker table caches flow entries for rules with permit actions. One DFW instance is permitted per VM vNIC. A specific flow is identified by the 5-tuple information consisting source IP address, destination IP address, protocols, L4 source port, and L4 destination port fields. By default, DFW does not perform a lookup on L4 source port, but it can be configured to do so by defining a specific policy rule.

No.	'111 Patent Claim 18	VMware NSX
		NSX Design Guide at 30-31 An IP packet identified as pkt1 that matches rule number 2. The order of operation is the following: 30 1. A lookup is performed in the connection tracker table to determine if an entry for the flow already exists. 2. As flow 3 is not present in the connection tracker table, a lookup is performed in the rule table to identify which rule is applicable to flow 3. The first rule that matches the flow will be enforced. 3. Rule 2 matches for flow 3. The action is set to 'Allow'. 4. Because the action is set to 'Allow' for flow 3, a new entry will be created inside the connection tracker table. The packet is then transmitted out of DFW.  Figure 23 – DFW Policy Rule Lookup and Packet – Subsequent Packets.

No.	'111 Patent Claim 18	VMware NSX
		VMW_ORCT_CSCO_ 2754 Location Header in the response body contains the resource url for the newly created rule resource. This URL can be used to identify this resource.
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.	VMware NSX 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. For example, VMware NSX System discloses checking if the packet matches the rule by checking the source or destination addresses of the packet. NSX Design Guide at 29

No.	'111 Patent Claim 18	VMware NSX
		Traffic redirection to third party vendor provides the capability to steer a particular type of traffic to a selected partner services VM. For instance, web traffic from the Internet to a set of Apache or Tomcat servers can be redirected to an L4-L7 deep packet inspection firewall for advanced protection. Traffic redirection is defined under Service Composer/Security Policy for NSX version 6.0 or under Partner Security Services tab of the DFW menu in NSX version 6.1. The Partner Security Services tab provides a powerful and easy to use user interface to define what type of traffic needs to be redirected to which partner services. It follows the same policy definition construct as DFW, providing the same options for source field, destination field and services field. The only difference is in the action field; instead of Block/Allow/Reject, a user can select between redirect/no redirect followed by a partner list. Any partner that has been registered with NSX can be successfully deployed on the platform. Additionally, log options can be enabled for this traffic redirection rule. The DFW instance on an ESXi host contains 2 separate tables. The rule table is used to store all policy rules, while the connection tracker table caches flow entries for rules with permit actions. One DFW instance is permitted per VM vNIC. A specific flow is identified by the 5-tuple information consisting source IP address, destination IP address, protocols, L4 source port, and L4 destination port fields. By default, DFW does not perform a lookup on L4 source port, but it can be configured to do so by defining a specific policy rule. NSX Design Guide at 30-31

No.	'111 Patent Claim 18	VMware NSX
		An IP packet identified as pkt1 that matches rule number 2. The order of operation is the following: 30 1. A lookup is performed in the connection tracker table to determine if an entry for the flow already exists. 2. As flow 3 is not present in the connection tracker table, a lookup is performed in the rule table to identify which rule is applicable to flow 3. The first rule that matches the flow will be enforced. 3. Rule 2 matches for flow 3. The action is set to 'Allow'. 4. Because the action is set to 'Allow' for flow 3, a new entry will be created inside the connection tracker table. The packet is then transmitted out of DFW.  Figure 23 – DFW Policy Rule Lookup and Packet – Subsequent Packets. VMW_ORCT_CSCO_ 1121

No.	'111 Patent Claim 18	VMware NSX

		Update source address match rule. <ip-address> is a IP address or IP address/mask where mask is an integer 1-32.
		`set network acl <rule-type> <rule-number> destination-address <equals|not-equals> <ip-address>`
		Update destination address match rule. <ip-address> is a IP address or IP address/mask where mask is an integer 1-32.
		`set network acl <rule-type> <rule-number> interface <equals|not-equals> <interface>`
		Update interface match rule. If interface ends with +, a starts-with match is performed.
		`set network acl <rule-type> <rule-number> rule_action <action>`
		Update rule action. <action> is one of: accept, drop, null.

No.	'111 Patent Claim 19	VMware NSX
19	The method according to claim 18, wherein the addresses are Internet Protocol (IP) addresses.	VMware NSX System discloses the method according to claim 18, wherein the addresses are Internet Protocol (IP) addresses. For example, VMware NSX System discloses source and destination IP addresses, wherein the addresses are Internet Protocol (IP) addresses. *See supra* at Claim 18. NSX Design Guide at 30

DFW rules are enforced as follows:

- Rules are processed in top-to-bottom ordering.
- Each packet is checked against the top rule in the rule table before moving down the subsequent rules in the table.
- The first rule in the table that matches the traffic parameters is enforced. No subsequent rules can be enforced as the search is then terminated for that packet.

Because of this behavior, it is always recommended to put the most granular policies at the top of the rule table. This will ensure they will be enforced before more specific rules.

The DFW default policy rule, located at the bottom of the rule table, is a "catch-all" rule; packets not matching any other rules will be enforced by the default rule. After the host preparation operation, the DFW default rule is set to 'allow' action. This ensures that VM-to-VM communication is not broken during staging or migration phases. It is a best practice to then change this default rule to 'block' action and enforce access control through a positive control model (i.e., only traffic defined in the firewall policy is allowed onto the network).

Figure 22 steps through policy rule lookup and packet flow:

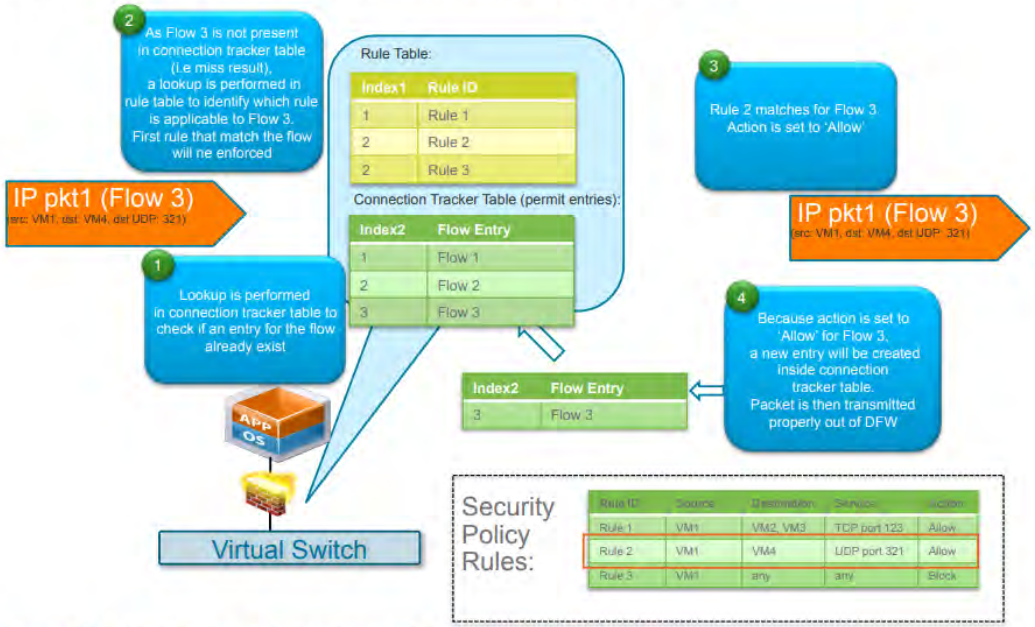
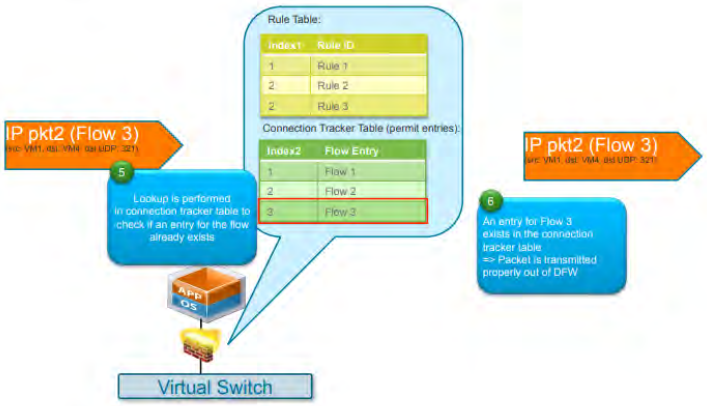
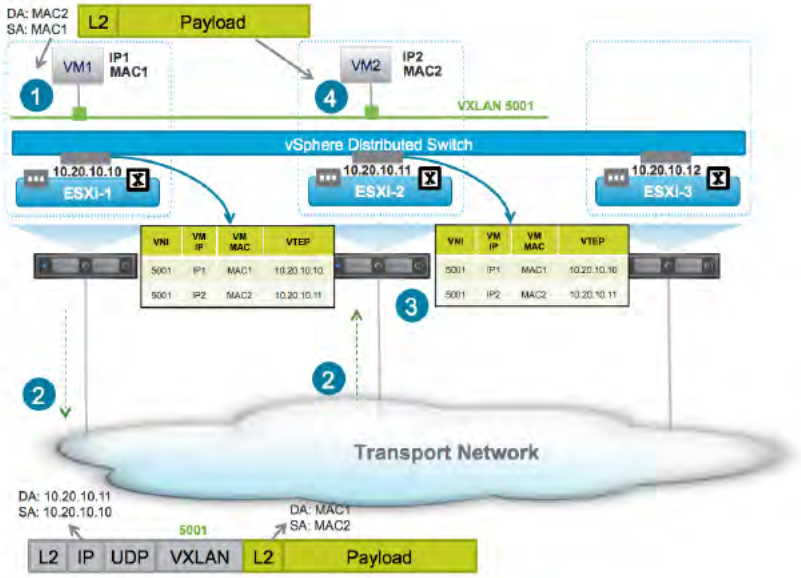


Figure 22 – DFW Policy Rule Lookup and Packet – First Packet

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 2053 of 2137

No.	'111 Patent Claim 19	VMware NSX
		NSX Design Guide at 30-31 An IP packet identified as pkt1 that matches rule number 2. The order of operation is the following: 30 1. A lookup is performed in the connection tracker table to determine if an entry for the flow already exists. 2. As flow 3 is not present in the connection tracker table, a lookup is performed in the rule table to identify which rule is applicable to flow 3. The first rule that matches the flow will be enforced. 3. Rule 2 matches for flow 3. The action is set to 'Allow'. 4. Because the action is set to 'Allow' for flow 3, a new entry will be created inside the connection tracker table. The packet is then transmitted out of DFW.  Figure 23 – DFW Policy Rule Lookup and Packet – Subsequent Packets.

No.	'111 Patent Claim 19	VMware NSX
		NSX Design Guide at 45  Figure 36 - Intra Logical Switch Unicast Communication 1. VM1 generates a data packet directed to VM2. 2. ESXi-1 learned about the location of VM2 from the ARP report received from the controller through control plane learning. It encapsulates the packet originated by VM1 in a VXLAN packet destined to the VTEP of ESXi2 – 10.20.10.11. 3. ESXi-2 receives the packet and leverages the information in the external IP header to learn about the location of VM1, associating VM1 MAC and IP addresses to the VTEP of ESXi-1. 4. The frame is delivered to VM2. VMW_ORCT_CSCO_2776

No.	'111 Patent Claim 19	VMware NSX
		Working with SpoofGuard After synchronizing with the vCenter Server, NSX Manager collects the IP addresses of all vCenter guest virtual machines from VMware Tools on each virtual machine. If a virtual machine has been compromised, the IP address can be spoofed and malicious transmissions can bypass firewall policies. You create a SpoofGuard policy for specific networks that allows you to authorize the IP addresses reported by VMware Tools and alter them if necessary to prevent spoofing. SpoofGuard inherently trusts the MAC addresses of virtual machines collected from the VMX files and vSphere SDK. Operating separately from Firewall rules, you can use SpoofGuard to block traffic determined to be spoofed. VMW_ORCT_CSCO_2780

No.	'111 Patent Claim 19	VMware NSX
		Approve IP Addresses Approves specified IP addresses. **Example 10-46.** Approve IP addresses Request: POST https://<vsm ip>/api/4.0/services/spoofguard/<policyId>?action=approve Request Body: <spoofguardList> <spoofguard> <id>5009aa18-ab89-ab9d-9386-c7f0da8773aa.000</id> <vnicUuid>50204903-f1c9-0e97-e222-4b96f87ec7fe.000</vnicUuid> <approvedIpAddress> <ipAddress>10.24.123.129</ipAddress> <ipAddress>10.24.123.130</ipAddress> <ipAddress>10::129</ipAddress> </approvedIpAddress> <approvedMacAddress>00:50:56:be:00:06</approvedMacAddress> <approvedBy>system_user</approvedBy> <approvedOn>2011-10-28 16:12:20.0</approvedOn> <publishedIpAddress> <ipAddress>10.24.123.129</ipAddress> <ipAddress>10::129</ipAddress> </publishedIpAddress> <publishedMacAddress>00:50:56:be:00:06</publishedMacAddress> <publishedBy>system_user</publishedBy> <publishedOn>2011-10-28 16:12:20.0</publishedOn> </spoofguard> <spoofguard> </spoofguard> </spoofguardList> VMW_ORCT_CSCO_ 2825

No.	'111 Patent Claim 19	VMware NSX

		Activity Monitoring provides visibility into your virtual network to ensure that security policies at your organization are being enforced correctly.
		
		A Security policy may mandate who is allowed access to what applications. The Cloud administrator can generate Activity Monitoring reports to see if the IP based firewall rule that they set is doing the intended work. By providing user and application level detail, Activity Monitoring translates high level security policies to low level IP address and network based implementation.
		
		Once you enable data collection for Activity Monitoring, you can run reports to view inbound traffic (such as virtual machines being accessed by users) as well as outbound traffic (resource utilization, interaction between inventory containers, and AD groups that accessed a server).
		
		VMW_ORCT_CSCO_ 1121
		
		Update source address match rule. <ip-address> is a IP address or IP address/mask where mask is an integer 1-32.
		`set network acl <rule-type> <rule-number> destination-address <equals|not-equals> <ip-address>`
		Update destination address match rule. <ip-address> is a IP address or IP address/mask where mask is an integer 1-32.
		`set network acl <rule-type> <rule-number> interface <equals|not-equals> <interface>`
		Update interface match rule. If interface ends with +, a starts-with match is performed.
		`set network acl <rule-type> <rule-number> rule_action <action>`
		Update rule action. <action> is one of: accept, drop, null.

No.	'111 Patent Claim 20	VMware NSX
20[a]	The method according to claim 1, wherein the packet is an Transmission Control Protocol (TCP) packet that	VMware NSX System 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. *See supra* at Claim 1, 17[a].

No.	'111 Patent Claim 20	VMware NSX
	comprises source and destination TCP ports, a TCP sequence number, and a TCP sequence mask fields, and	
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.	VMware NSX System 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. For example, VMware NSX System discloses checking if the packet header has specific TCP ports fields that match to the rule table . NSX Design Guide at 25

No.	'111 Patent Claim 20	VMware NSX
		3.1.7 NSX Distributed Firewall (DFW) The NSX DFW provides L2-L4 stateful firewall services to any workload in the NSX environment. DFW runs in the kernel space and provides near line rate network traffic protection. DFW performance and throughput scale linearly through addition of new ESXi hosts. DFW is activated as soon as the host preparation process is completed. If a VM does not require DFW service, it can be added in the exclusion list functionality. By default, NSX Manager, NSX Controllers, and Edge services gateways are automatically excluded from DFW function. One DFW instance is created per VM vNIC; for example, if a new VM with 3 vNICs is created, 3 instances of DFW will be allocated to this VM. Configuration of these DFW instances can be identical or different based on the "apply to" setting. When a DFW rule is created, the user can select a Point of Enforcement (PEP) for this rule, with options varying from Logical Switch to vNIC. By default, "apply to" is not selected, so DFW rules are propagated down to all the ESXi hosts which are part of the NSX domain that have been prepared for enforcement and applied to all the connected virtual machines. DFW policy rules can be written in 2 ways, using L2 rules (Ethernet) or L3/L4 rules. • L2 rules are mapped to L2 in the OSI model; only MAC addresses can be used in the source and destination fields and only L2 protocols can be used in the service fields (e.g., ARP). • L3/L4 rules are mapped to L3/L4 in the OSI model; policy rules can be written using IP addresses and TCP/UDP ports. NSX Design Guide at 27 When vCenter containers are used in the DFW policy rules, VMtools must be installed on the guest VMs. VMtools has visibility of the IP address of the VM – whether dynamically provided through DHCP or statically configured on the VM by the administrator – and provides this information down to the DFW engine that operates based on MAC, IP, and TCP/UDP port fields. If the DFW policy rules use only IP information (e.g., host IP, subnet IP, or IP sets) then the presence of VMtools on guest VM is not required. **Note:** With NSX release 6.2, VMtools is not mandatory. See the "Deploying Distributed Firewall" section for further details.

No.	'111 Patent Claim 20	VMware NSX
		NSX Design Guide at 30

DFW rules are enforced as follows:

- Rules are processed in top-to-bottom ordering.
- Each packet is checked against the top rule in the rule table before moving down the subsequent rules in the table.
- The first rule in the table that matches the traffic parameters is enforced. No subsequent rules can be enforced as the search is then terminated for that packet.

Because of this behavior, it is always recommended to put the most granular policies at the top of the rule table. This will ensure they will be enforced before more specific rules.

The DFW default policy rule, located at the bottom of the rule table, is a "catch-all" rule; packets not matching any other rules will be enforced by the default rule. After the host preparation operation, the DFW default rule is set to 'allow' action. This ensures that VM-to-VM communication is not broken during staging or migration phases. It is a best practice to then change this default rule to 'block' action and enforce access control through a positive control model (i.e., only traffic defined in the firewall policy is allowed onto the network).

Figure 22 steps through policy rule lookup and packet flow:

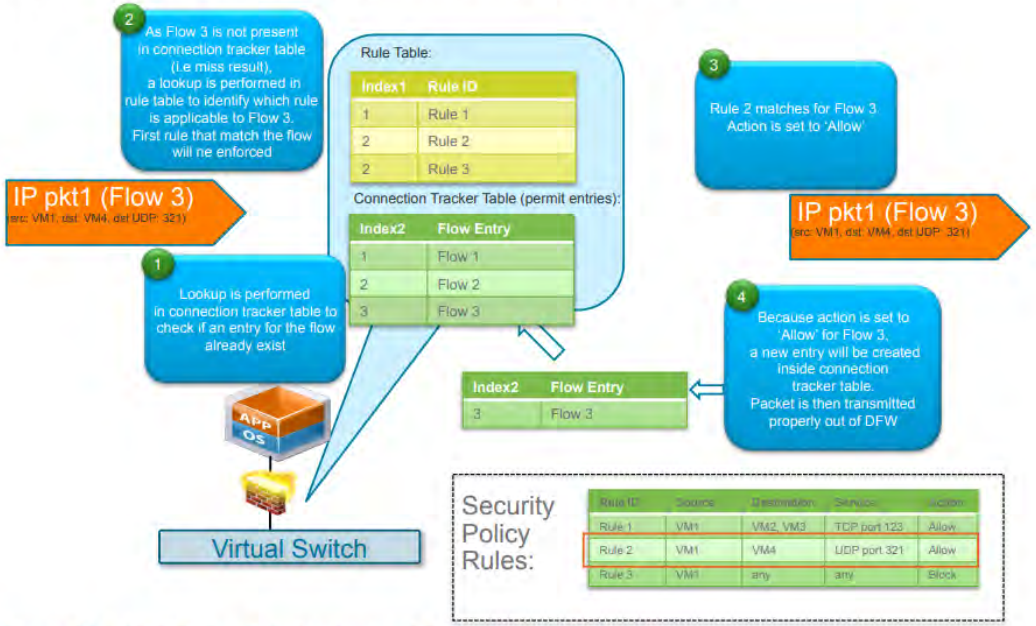


Figure 22 – DFW Policy Rule Lookup and Packet – First Packet

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 2062 of 2137

No.	'111 Patent Claim 20	VMware NSX
		VMW_ORCT_CSCO_ 2205-2206 **Distributed Firewall** Distributed firewall is a hypervisor kernel-embedded firewall that provides visibility and control for virtualized workloads and networks. You can create access control policies based on VMware vCenter objects like datacenters and clusters and virtual machine names; network constructs like IP or IPSet addresses, VLAN (DVS port-groups), VXLAN (logical switches), security groups, as well as user group identity from Active Directory. Firewall rules are enforced at the vNIC level of each virtual machine to provide consistent access control even when the virtual machine gets vMotioned. The hypervisor-embedde nature of the firewall delivers close to line rate throughput to enable higher workload consolidation on physical servers. The distributed nature of the firewall provides a scale-out architecture that automatically extends firewall capacity when additional hosts are added to a datacenter. For L2 packets, distributed firewall creates a cache for performance boost. L3 packets are processed in the following sequence: 1 All packets are checked for an existing state. This is done for SYNs too so that bogus or retransmitted SYNs for existing sessions can be detected. 2 If a state match is found, the packets are processed. 3 If a state match is not found, the packet is processed through the rules until a match is found. ■ For TCP packets, a state is set only for packets with a SYN flag. However, rules that do not specify a protocol (service ANY), can match TCP packets with any combination of flags. ■ For UDP packets, 5-tuple details are extracted from the packet. If a state does not exist in the state table, a new state is created using the extracted 5-tuple details. Subsequently received packets are matched against the state that was just created. ■ For ICMP packets, ICMP type, code, and packet direction are used to create a state. VMW_ORCT_CSCO_ 2212

No.	'111 Patent Claim 20	VMware NSX
		10 Point to the **Action** cell of the new rule and click ⊞. Make appropriate selections as described in the table below and click **OK**. **Action** — **Results in** **Allow** — Allows traffic from or to the specified source(s), destination(s), and service(s). **Block** — Blocks traffic from or to the specified source(s), destination(s), and service(s). **Reject** — Sends reject message for unaccepted packets. RST packets are sent for TCP connections. ICMP messages with administratively prohibited code are sent for UDP, ICMP, and other IP connections. **Log** — Logs all sessions matching this rule. Enabling logging can affect performance. **Do not log** — Does not log sessions. VMW_ORCT_CSCO_ 2231

No.	'111 Patent Claim 20	VMware NSX
		Add a Private Network Add the network that you want the remote user to be able to access. **Procedure** 1 In the **SSL Vpn-Plus** tab, select **Private Networks** from the left panel. 2 Click the **Add** ( ) icon 3 Type the private network IP address. 4 Type the netmask of the private network. 5 (Optional) Type a description for the network. 6 Specify whether you want to send private network and internet traffic over the SSL VPN-Plus enabled NSX Edge or directly to the private server by bypassing the NSX Edge. 7 If you selected **Send traffic over the tunnel**, select **Enable TCP Optimization** to optimize the internet speed. Conventional full-access SSL VPNs tunnel sends TCP/IP data in a second TCP/IP stack for encryption over the internet. This results in application layer data being encapsulated twice in two separate TCP streams. When packet loss occurs (which happens even under optimal internet conditions), a performance degradation effect called TCP-over-TCP meltdown occurs. In essence, two TCP instruments are correcting a single packet of IP data, undermining network throughput and causing connection timeouts. TCP Optimization eliminates this TCP-over-TCP problem, ensuring optimal performance. VMW_ORCT_CSCO_2261

No.	'111 Patent Claim 20	VMware NSX
		The NSX Edge load balancer enables network traffic to follow multiple paths to a specific destination. It distributes incoming service requests evenly among multiple servers in such a way that the load distribution is transparent to users. Load balancing thus helps in achieving optimal resource utilization, maximizing throughput, minimizing response time, and avoiding overload. NSX Edge provides load balancing up to Layer 7. You map an external, or public, IP address to a set of internal servers for load balancing. The load balancer accepts TCP, HTTP, or HTTPS requests on the external IP address and decides which internal server to use. Port 8090 is the default listening port for TCP, port 80 is the default port for HTTP, and port 443 is the default port for HTTPs. You must have a working NSX Edge instance before you can load balancing. For information on setting up NSX Edge, see "NSX Edge Operations," on page 188. For information on configuring an NSX Edge certificate, see "Working with Certificates," on page 188. VMW_ORCT_CSCO_ 158 *Example 1: Distributed Internet Firewall* One common use case for Security Profiles is implementing a "distributed firewall" that protects VMs connected to a shared network. Consider a tenant that runs a set of web server VMs that must be connected to the public Internet. The tenant wants these VMs to be reachable only on TCP ports 80 (HTTP) and 443 (HTTPS). The tenant also wants these VMs to be reachable on TCP port 22 (SSH), but only for traffic originating from the tenant's own customer premises, which uses IP block 8.8.8.0/24. To create these restrictions, the tenant can define a Security Profile called *Webservers* with the following rules: • allow port TCP 80 from source * • allow port TCP 443 from source * • allow port TCP 22 from source 8.8.8.0/24 When the tenant creates a new web server VM, he will associate the VM's public logical switch port with the Security Profile *WebServers* and the machines would automatically be protected.

No.	'111 Patent Claim 21	VMware NSX
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.	VMware NSX System 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. For example, VMware NSX System discloses a network that may be a WAN, Internet, intranet, or virtualized network as part of the data center. *See supra* at Claim 1. NSX Design Guide at 49

4.3 Logical Routing

The logical routing capability in the NSX platform provides the ability to interconnect both virtual and physical endpoints deployed in different logical L2 networks. This is possible due to the decoupling of network infrastructure from logical networks provided by the deployment of network virtualization.

Figure 40 shows both the logical and corresponding physical view of a routed topology interconnecting two logical switches.

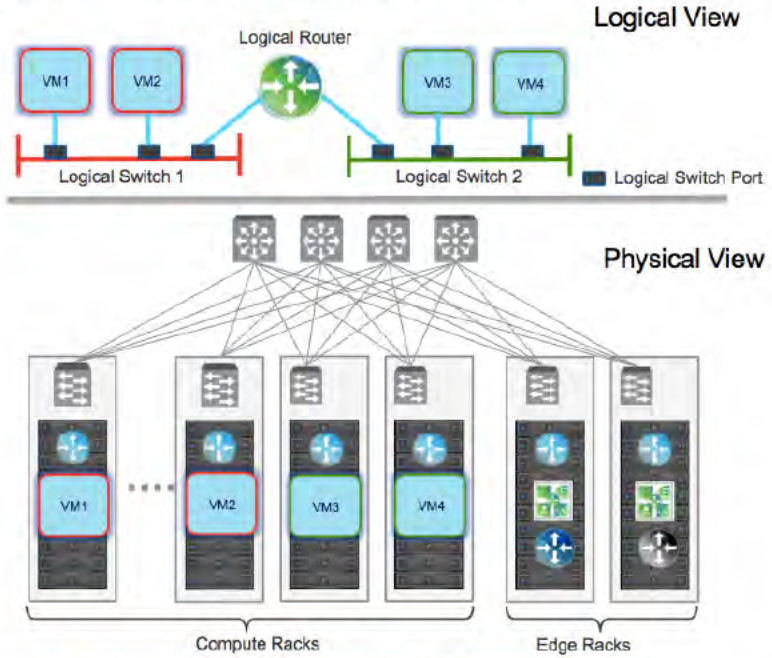


Figure 40 - Logical Routing (Logical and Physical Network Views)

The deployment of logical routing can serve two purposes; interconnecting endpoints – logical or physical – belonging to separate logical L2 domains or interconnecting endpoints belonging to logical L2 domains with devices deployed in the external L3 physical infrastructure. The first type of communication, often confined inside a data center, is referred to as east-west communication. The second type is north-south communication, which provides connectivity into the data center from the external physical world (e.g., WAN, Internet, or intranet).

No.	'111 Patent Claim 21	VMware NSX
		NSX Design Guide at 57 In typical deployments the routing protocol choice is already decided. The corporate routing team typically handles the backbone and core WAN/Internet domains with its administrative control extending to core/aggregation datacenter routers. The data center team usually handles the routing protocol decision below the core/aggregation boundary. In most data centers OSPF is the prevalent choice to maintain a consistency with core routing protocol; however, in modern, large scale data centers, BGP is the de facto protocol of choice – specifically those with spine-leaf topology. The NSX routing protocol selection and its interaction with existing network routing services should be evaluated with following critical understanding: • The NSX routing domain connects to existing network as an edge network. The NSX domain does not act as transport network, thus simplifying the connectivity and features required in a typical routing protocol to support a stub topology. • NSX does not dictate the routing protocol choice in itself, however the design requirements and connectivity options may restrict the choice to a protocol. VMW_ORCT_CSCO_. 571

Logical Routers

Create or Edit a Logical Router

A logical router provides layer-3 routed networking in a logical network and can be configured to offer additional services such as NAT and layer-3 routed connections to the external, physical network.

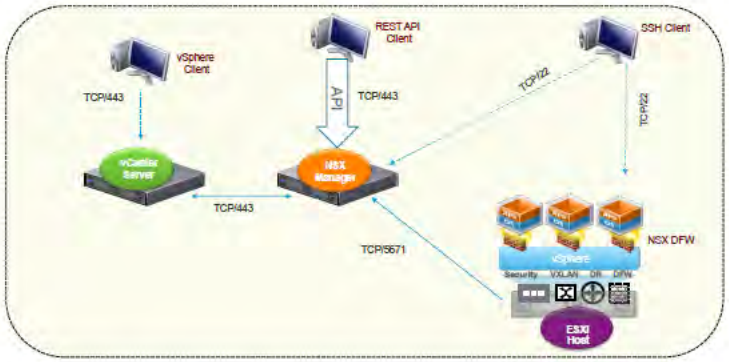
To create a logical router, follow these steps:

1. In NSX Manager, click on **Network Components: Logical Router**, and do one of:
   - *create a router* by clicking **Add** to launch the Create Logical Router wizard; or
   - *edit a router* by clicking the name of a logical router in the list and clicking it's **Edit** button.
2. Give the router a **Display Name** (optional). This name can be used to identify the router when you connect switches to it. If desired, click **Add Tag** to add a metadata tag to this logical router, to help identify it when managing your logical networks.
3. Click **Next**.
4. In the **Routing Type** dropdown box, choose one of:
   - *Routing Table*: Choose this option to allow yourself and other administrators to define static routes on this logical router. (After you have saved the logical router, you must set up the routes as shown in Static routes on logical routers (page 106).)
   - *Single Default Route*: Choose this option to have this router use a single default route for all traffic. This assumes all traffic will be routable via the NSX Gateway that connects this router to the datacenter network. You must specify a **Default Gateway IP Address** if this router will act as an L3 Gateway to the datacenter network. This IP address is the physical gateway exit or, in other words, the direct uplink to the physical router. For example, this might be an HSRP or VRRP segment address used by the physical routers that serve the NSX environment. Many logical routers can share a single physical Gateway IP address.
5. If this logical router will perform network address translation and you wish to have NAT connections survive Gateway failover, tick the **Enable NAT Synchronization** checkbox. If left unticked, in the event of a Gateway failover, your NAT rules will be evaluated again after the failover to re-establish any needed NAT connections.
6. The **Replication Mode** determines which transport nodes handle replication of broadcast and multicast traffic for ports on this logical router. Choose one of the following:
   - **Service Nodes** mode: This is the default and recommended mode. All broadcast and multicast traffic traverses a Service Node.
   - **Source Nodes** mode: (Not available on *distributed* logical routers.) Broadcast and multicast traffic replication is handled directly on each packet's source hypervisor instead of using the Service Nodes. Note: This feature is intended primarily to support future hardware integrations and as such does not provide significant performance improvements in NVP 3.1 installations.
7. Click **Next**.

Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 2070 of 2137

No.	'111 Patent Claim 21	VMware NSX
		VMW_ORCT_CSCO_ 1413 **Add a Private Network** Add the network that you want the remote user to be able to access. **Procedure** 1 In the **SSL Vpn-Plus** tab, select **Private Networks** from the left panel. 2 Click the **Add** ( ) icon 3 Type the private network IP address. 4 Type the netmask of the private network. 5 (Optional) Type a description for the network. 6 Specify whether you want to send private network and internet traffic over the SSL VPN-Plus enabled NSX Edge or directly to the private server by bypassing the NSX Edge. 7 If you selected **Send traffic over the tunnel**, select **Enable TCP Optimization** to optimize the internet speed. Conventional full-access SSL VPNs tunnel sends TCP/IP data in a second TCP/IP stack for encryption over the internet. This results in application layer data being encapsulated twice in two separate TCP streams. When packet loss occurs (which happens even under optimal internet conditions), a performance degradation effect called TCP-over-TCP meltdown occurs. In essence, two TCP instruments are correcting a single packet of IP data, undermining network throughput and causing connection timeouts. TCP Optimization eliminates this TCP-over-TCP problem, ensuring optimal performance. 8 Type the port numbers that you want to open for the remote user to access the corporate internal servers/machines like 3389 for RDP, 20/21 for FTP, and 80 for http. If you want to give unrestricted access to the user, you can leave the **Ports** field blank.

No.	'111 Patent Claim 22	VMware NSX
22	The method according to claim 1, wherein the first entity is a server device and the second entity is a	VMware NSX System 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. For example, VMware NSX System discloses transmitting packets between client devices and servers.

No.	'111 Patent Claim 22	VMware NSX
	client device, or wherein the first entity is a client device and the second entity is a server device..	*See supra* at Claim 1. NSX Design Guide at 27 The SSH client depicted in Figure 19 can access the ESXi host CLI to perform specific DFW related troubleshooting.  Figure 19 – NSX DFW System Architecture and Communications Channels When vCenter containers are used in the DFW policy rules, VMtools must be installed on the guest VMs. VMtools has visibility of the IP address of the VM – whether dynamically provided through DHCP or statically configured on the VM by the administrator – and provides this information down to the DFW engine that operates based on MAC, IP, and TCP/UDP port fields. If the DFW policy rules use only IP information (e.g., host IP, subnet IP, or IP sets) then the presence of VMtools on guest VM is not required. **Note:** With NSX release 6.2, VMtools is not mandatory. See the "Deploying Distributed Firewall" section for further details. NSX Design Guide at 95

Proxy Mode (One-Arm)

Proxy mode configuration is highlighted in Figure 90 and consists of deploying an NSX Edge directly connected to the logical network where load-balancing services are required.

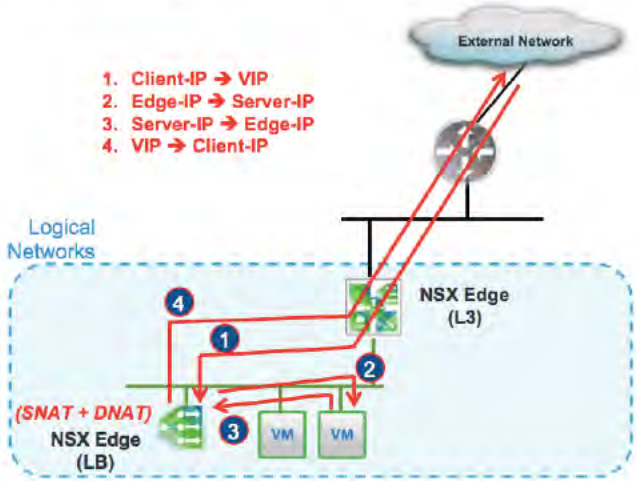


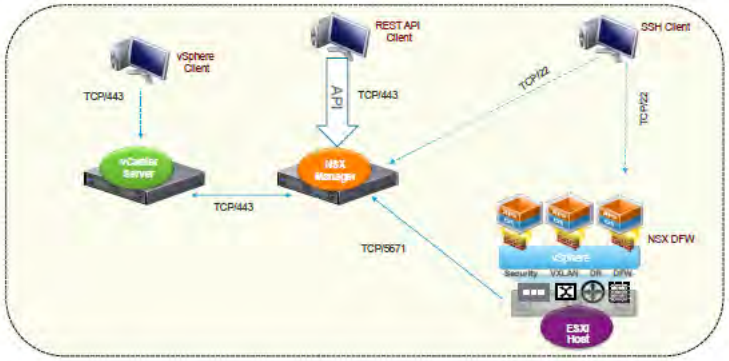
Figure 90 - One-Arm Mode Load Balancing Services

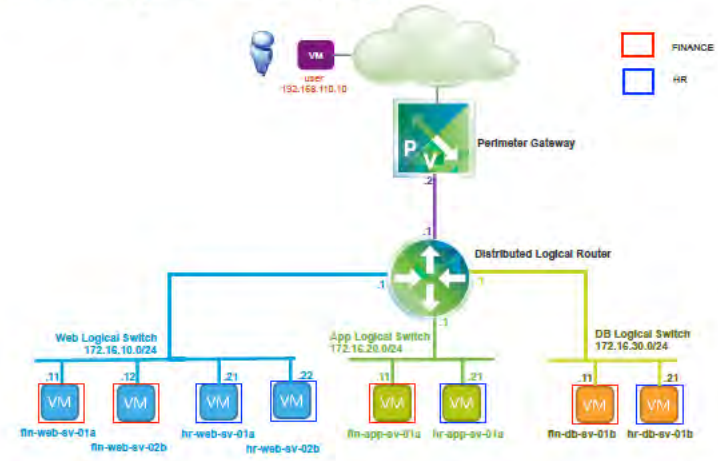
In this configuration mode:

1. The external client sends traffic to the Virtual IP address (VIP) exposed by the load balancer.
2. The load balancer performs two address translations on the original packets received from the client: destination NAT (D-NAT) to replace the VIP with the IP address of one of the servers deployed in the server farm, and source NAT (S-NAT) to replace the client IP address with the IP address identifying the load-balancer itself. S-NAT is required to force through the load balancer the return traffic from the server farm to the client.
3. The server in the server farm replies by sending the traffic to the load balancer per S-NAT functionality.
4. The load balancer again performs a source and destination NAT service to send traffic to the external client, leveraging its VIP as source IP address.

No.	'111 Patent Claim 22	VMware NSX
		NSX Design Guide at 100 **4.6.2 L3 VPN** Finally, L3 VPN services are used to provide secure L3 connectivity into the datacenter network from remote locations.   Figure 95 - NSX L3 VPN Services As highlighted in Figure 95, the L3 VPN services can be used by remote clients leveraging SSL tunnels to securely connect to private networks behind a NSX Edge gateway acting as L3 VPN Server in the datacenter. This service is usually referred to as SSL VPN-Plus. Alternatively, the NSX Edge can be deployed to use standard IPSec protocol settings to interoperate with all major physical VPN vendors' equipment and establish site-so-site secure L3 connections. It is possible to connect multiple remote IP subnets to the internal network behind the NSX Edge. Differently from the previously described L2 VPN service, connectivity in this case is routed since the remote and local subnets are part of different address spaces. In the current implementation, unicast-only communication is supported across the IPSec connection; thus dynamic routing protocols cannot be used and a static routing configuration is required. VMW_ORCT_CSCO_2939

No.	'111 Patent Claim 22	VMware NSX
		show service l2vpn (on server) Shows the L2 VPN server status and tunnel information along with the encryption algorithm that is being used in the communication. **Synopsis** show service l2vpn **CLI Mode** Basic, Privileged **Example** vShield-edge-1-0> show service l2vpn L2 VPN is running --- L2 VPN type: Server Tunnel information: 1 ABC na 1 1402561453 AES128-SHA **show service l2vpn (on client)** Shows the L2 VPN client status. **Synopsis** show service l2vpn **CLI Mode** Basic, Privileged **Example** vShield-edge-1-0> show service l2vpn L2 VPN is running --- L2 VPN type: Client Tunnel status: up Total bytes sent: 582 Total bytes received: 408

No.	'111 Patent Claim 23	VMware NSX
23[a]	The method according to claim 22, wherein the server device comprises a web server, and	VMware NSX System discloses the method according to claim 22, wherein the server device comprises a web server. For example, VMware NSX System discloses a server that communicates with a client over the Internet or web. *See supra* at Claim 22. NSX Design Guide at 27 The SSH client depicted in Figure 19 can access the ESXi host CLI to perform specific DFW related troubleshooting.  Figure 19 – NSX DFW System Architecture and Communications Channels When vCenter containers are used in the DFW policy rules, VMtools must be installed on the guest VMs. VMtools has visibility of the IP address of the VM – whether dynamically provided through DHCP or statically configured on the VM by the administrator – and provides this information down to the DFW engine that operates based on MAC, IP, and TCP/UDP port fields. If the DFW policy rules use only IP information (e.g., host IP, subnet IP, or IP sets) then the presence of VMtools on guest VM is not required. **Note:** With NSX release 6.2, VMtools is not mandatory. See the "Deploying Distributed Firewall" section for further details.

No.	'111 Patent Claim 23	VMware NSX
		NSX Design Guide at 89-90 **4.4.7 Micro-Segmentation with NSX DFW and Implementation** DFW can be leveraged to implement micro-segmentation for a 3-tier application (e.g., web/app/database) where multiple organizations (e.g., finance, HR) share the same logical network topology.  Figure 80 – Micro-Segmentation Use Case This network design contains 3 logical switches (web LS, app LS, and db LS) interconnected through a Distributed Logical Router. IP addresses defined on DLR are the default gateway for VM connected to any of the Logical Switches. DLR is connected to an Edge services gateway that provides connectivity to the physical world. An end user connected to the external network can reach the web Logical Switch subnet where a dynamic routing protocol enabled on the DLR and the ESG to announce web LS subnet. In this example, the finance and HR organization each have 2 web servers, 1 app server and 1 database server.

No.	'111 Patent Claim 23	VMware NSX
		In this micro-segmentation design, servers of same role are connected to same Logical Switch, irrespective of the organization they belong to. HR and Finance are used as examples; it is possible to find many more organizations to host on this logical network infrastructure. Using DFW, it is possible to achieve the same level of security if Finance and HR workloads are connected to different logical networks. The power of DFW resides in the fact that the network topology is no longer a barrier to security enforcement; the same level of traffic access control can be achieved for any type of topology. To group VMs of same role and of same organization together, leverage the Service Composer functionality. One property within Service Composer is the Security Group (SG) construct. SG allows for dynamically or statically including objects in a container. This container will then be used as the source or destination field of our DFW security policy rule. NSX Design Guide at 95

Proxy Mode (One-Arm)

Proxy mode configuration is highlighted in Figure 90 and consists of deploying an NSX Edge directly connected to the logical network where load-balancing services are required.

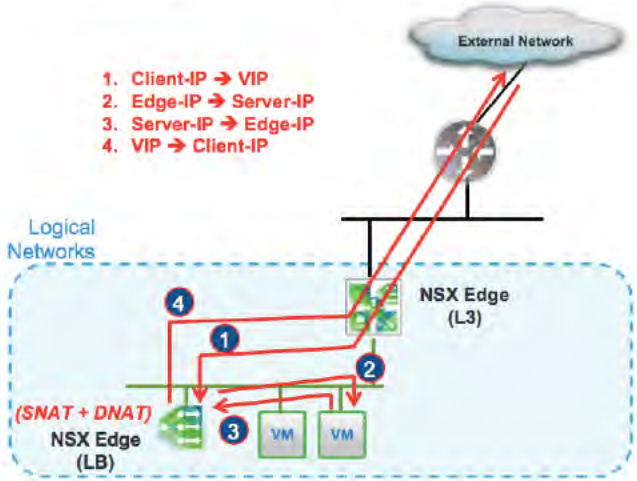
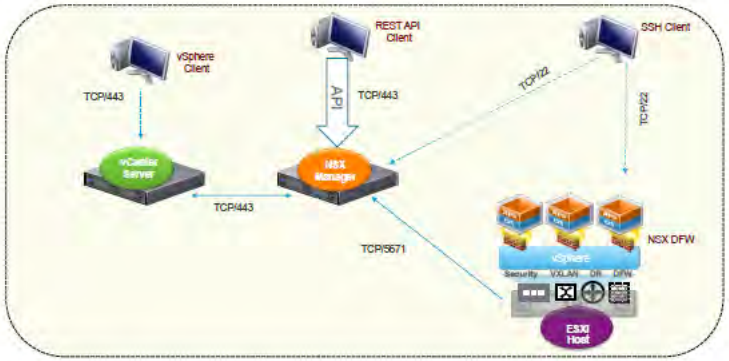


Figure 90 - One-Arm Mode Load Balancing Services

In this configuration mode:

1. The external client sends traffic to the Virtual IP address (VIP) exposed by the load balancer.
2. The load balancer performs two address translations on the original packets received from the client: destination NAT (D-NAT) to replace the VIP with the IP address of one of the servers deployed in the server farm, and source NAT (S-NAT) to replace the client IP address with the IP address identifying the load-balancer itself. S-NAT is required to force through the load balancer the return traffic from the server farm to the client.
3. The server in the server farm replies by sending the traffic to the load balancer per S-NAT functionality.
4. The load balancer again performs a source and destination NAT service to send traffic to the external client, leveraging its VIP as source IP address.

No.	'111 Patent Claim 23	VMware NSX
		NSX Design Guide at 100 **4.6.2 L3 VPN** Finally, L3 VPN services are used to provide secure L3 connectivity into the datacenter network from remote locations.   Figure 95 - NSX L3 VPN Services As highlighted in Figure 95, the L3 VPN services can be used by remote clients leveraging SSL tunnels to securely connect to private networks behind a NSX Edge gateway acting as L3 VPN Server in the datacenter. This service is usually referred to as SSL VPN-Plus. Alternatively, the NSX Edge can be deployed to use standard IPSec protocol settings to interoperate with all major physical VPN vendors' equipment and establish site-so-site secure L3 connections. It is possible to connect multiple remote IP subnets to the internal network behind the NSX Edge. Differently from the previously described L2 VPN service, connectivity in this case is routed since the remote and local subnets are part of different address spaces. In the current implementation, unicast-only communication is supported across the IPSec connection; thus dynamic routing protocols cannot be used and a static routing configuration is required.

No.	'111 Patent Claim 23	VMware NSX
23[b]	wherein the client device comprises a smartphone, a tablet computer, a personal computer, a laptop computer, or a wearable computing device.	VMware NSX System discloses wherein the client device comprises a smartphone, a tablet computer, a personal computer, a laptop computer, or a wearable computing device. For example, VMware NSX System discloses client devices that may be computers, laptops, or other network devices. NSX Design Guide at 27 The SSH client depicted in Figure 19 can access the ESXi host CLI to perform specific DFW related troubleshooting.  Figure 19 – NSX DFW System Architecture and Communications Channels When vCenter containers are used in the DFW policy rules, VMtools must be installed on the guest VMs. VMtools has visibility of the IP address of the VM – whether dynamically provided through DHCP or statically configured on the VM by the administrator – and provides this information down to the DFW engine that operates based on MAC, IP, and TCP/UDP port fields. If the DFW policy rules use only IP information (e.g., host IP, subnet IP, or IP sets) then the presence of VMtools on guest VM is not required. **Note:** With NSX release 6.2, VMtools is not mandatory. See the "Deploying Distributed Firewall" section for further details. NSX Design Guide at 95

Proxy Mode (One-Arm)

Proxy mode configuration is highlighted in Figure 90 and consists of deploying an NSX Edge directly connected to the logical network where load-balancing services are required.

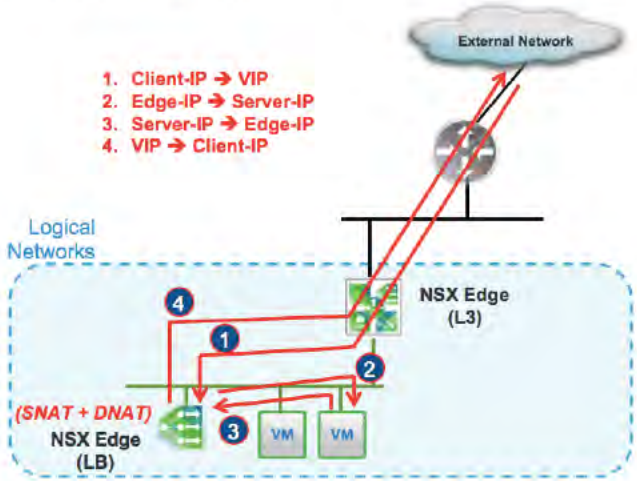


Figure 90 - One-Arm Mode Load Balancing Services

In this configuration mode:

1. The external client sends traffic to the Virtual IP address (VIP) exposed by the load balancer.
2. The load balancer performs two address translations on the original packets received from the client: destination NAT (D-NAT) to replace the VIP with the IP address of one of the servers deployed in the server farm, and source NAT (S-NAT) to replace the client IP address with the IP address identifying the load-balancer itself. S-NAT is required to force through the load balancer the return traffic from the server farm to the client.
3. The server in the server farm replies by sending the traffic to the load balancer per S-NAT functionality.
4. The load balancer again performs a source and destination NAT service to send traffic to the external client, leveraging its VIP as source IP address.

No.	'111 Patent Claim 23	VMware NSX
		NSX Design Guide at 100 **4.6.2 L3 VPN** Finally, L3 VPN services are used to provide secure L3 connectivity into the datacenter network from remote locations.   Figure 95 - NSX L3 VPN Services As highlighted in Figure 95, the L3 VPN services can be used by remote clients leveraging SSL tunnels to securely connect to private networks behind a NSX Edge gateway acting as L3 VPN Server in the datacenter. This service is usually referred to as SSL VPN-Plus. Alternatively, the NSX Edge can be deployed to use standard IPSec protocol settings to interoperate with all major physical VPN vendors' equipment and establish site-so-site secure L3 connections. It is possible to connect multiple remote IP subnets to the internal network behind the NSX Edge. Differently from the previously described L2 VPN service, connectivity in this case is routed since the remote and local subnets are part of different address spaces. In the current implementation, unicast-only communication is supported across the IPSec connection; thus dynamic routing protocols cannot be used and a static routing configuration is required. VMW_ORCT_CSCO_ 2034

No.	'111 Patent Claim 23	VMware NSX
		Distributed firewall can help in creating identity-based rules as well. Administrators can enforce access control based on the user's group membership as defined in the enterprise Active Directory. Here are some scenarios where identity-based firewall rules can be used: ■ User accessing virtual applications using a laptop/mobile device where AD is used for user authentication ■ User accessing virtual applications using VDI infrastructure where the virtual machines are Microsoft Windows based VMW_ORCT_CSCO_ 3052 **What to do next** 1 Add an SNAT rule to translate the IP address of the NSX Edge appliance to the VPN Edge IP address. 2 Using a web browser, navigate to the IP address of the NSX Edge interface by typing `https//NSXEdgeIPAddress`. 3 Login using the user name and password that you created in the "Add a User," on page 78 section and download the installation package. 4 Enable port forwarding on your router for the port number used in "Add SSL VPN-Plus Server Settings," on page 72. 5 Launch the VN client, select your VPN server, and login. You can now navigate to the services on your network. SSL VPN-Plus gateway logs are sent to the syslog server configured on the NSX Edge appliance. SSL VPN-Plus client logs are stored in the following directory on the remote user's computer: `%PROGRAMFILES%/VMWARE/SSLVPN Client/`. VMW_ORCT_CSCO_. 3053 **SSL VPN-Plus Logs** SSL VPN-Plus gateway logs are sent to the syslog server configured on the NSX Edge appliance. SSL VPN-Plus client logs are stored in the following directory on the remote user's computer: `%PROGRAMFILES %/VMWARE/SSL VPN Client/`.

No.	'111 Patent Claim 24	VMware NSX
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.	VMware NSX System 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. For example, VMware NSX System discloses using standard protocols for forwarding rules, entries, and tables from the NSX controller to the vSwitches, including OpenFlow. *See supra* at Claim 22. Switching at 1 Layer-2 forwarding VMware NSX supports traditional layer-2 segments with proper flooding of BUM (Broadcast, Unknown unicast, Multicast) frames. NSX controller downloads forwarding entries to individual virtual switches, either through OpenFlow (NSX for multiple hypervisors) or a proprietary protocol (NSX for vSphere). The forwarding entries map destination VM MAC addresses into destination hypevisor (or gateway) IP addresses. On top of static forwarding entries downloaded from the controller, virtual switches perform dynamic MAC learning for MAC addresses reachable through layer-2 gateways. NSX Design Guide at 57

No.	'111 Patent Claim 24	VMware NSX
		In typical deployments the routing protocol choice is already decided. The corporate routing team typically handles the backbone and core WAN/Internet domains with its administrative control extending to core/aggregation datacenter routers. The data center team usually handles the routing protocol decision below the core/aggregation boundary. In most data centers OSPF is the prevalent choice to maintain a consistency with core routing protocol; however, in modern, large scale data centers, BGP is the de facto protocol of choice – specifically those with spine-leaf topology. The NSX routing protocol selection and its interaction with existing network routing services should be evaluated with following critical understanding: • The NSX routing domain connects to existing network as an edge network. The NSX domain does not act as transport network, thus simplifying the connectivity and features required in a typical routing protocol to support a stub topology. • NSX does not dictate the routing protocol choice in itself, however the design requirements and connectivity options may restrict the choice to a protocol. NSX Design Guide at 100

No.	'111 Patent Claim 24	VMware NSX
		4.6.2 L3 VPN Finally, L3 VPN services are used to provide secure L3 connectivity into the datacenter network from remote locations.   Figure 95 - NSX L3 VPN Services As highlighted in Figure 95, the L3 VPN services can be used by remote clients leveraging SSL tunnels to securely connect to private networks behind a NSX Edge gateway acting as L3 VPN Server in the datacenter. This service is usually referred to as SSL VPN-Plus. Alternatively, the NSX Edge can be deployed to use standard IPSec protocol settings to interoperate with all major physical VPN vendors' equipment and establish site-so-site secure L3 connections. It is possible to connect multiple remote IP subnets to the internal network behind the NSX Edge. Differently from the previously described L2 VPN service, connectivity in this case is routed since the remote and local subnets are part of different address spaces. In the current implementation, unicast-only communication is supported across the IPSec connection; thus dynamic routing protocols cannot be used and a static routing configuration is required. VMW_ORCT_CSCO_ 57

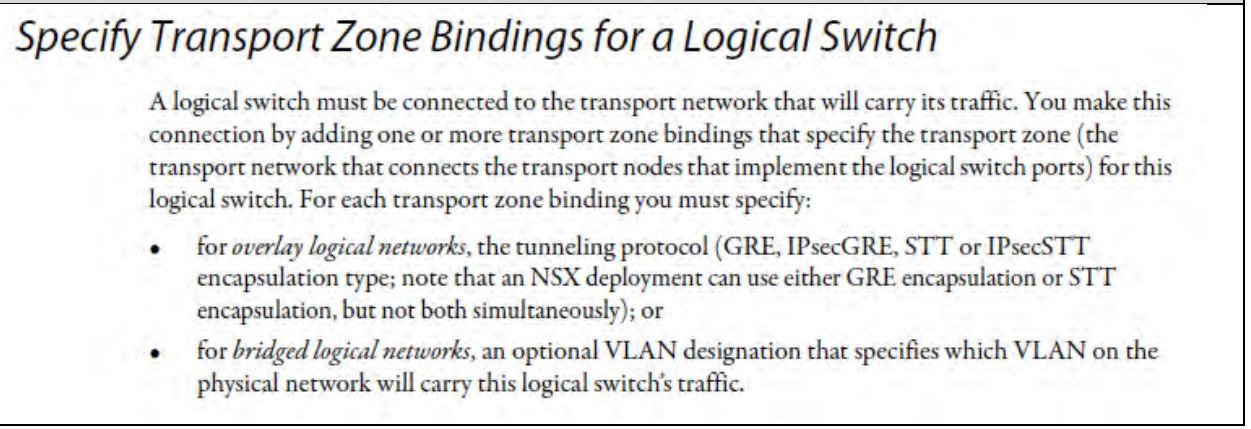
No.	'111 Patent Claim 24	VMware NSX
		NSX Control Plane The NSX Controller Cluster accepts logical network configuration instructions from administrators (through the NVP API and its clients), calculates the required network flow entries, and inserts these network flow entries into Open vSwitch (OVS) instances running on the transport nodes (hypervisor switches and NSX appliances) via OpenFlow. In each transport node, the flow entries give OVS the routing information it needs to direct packets to the right hypervisor(s) or network gateway. When OVS on a transport node encounters a packet that it does not know how to route, it asks the Controller Cluster to provide a new flow entry for that packet. In addition to logical network set-up capabilities, the NVP API allows administrators to configure and manage the forwarding state of transport nodes. The Controller Cluster is a distributed system that runs on a set of x86 servers. Clustering provides: • **High-availability**—The ability to survive failures of individual Controller Nodes. • **Scale-out**—The ability to increase the number of endpoints managed by the control plane by adding more servers to the Controller Cluster. VMW_ORCT_CSCO_ 93 **Controller Cluster** The Controller Cluster is both an OpenFlow controller that manages transport nodes and a logical network manager that enables administrators to build and maintain the logical networks carried by the transport nodes. (Each transport node, such as a hypervisor, Service Node, or Gateway, runs Open vSwitch.) Through the NVP API and its clients, the NSX Controller Cluster accepts administrators' logical network configuration instructions, calculates the required network flows, and inserts these network flows into the appropriate transport nodes via OpenFlow. The Controller Cluster is a distributed system that runs on a set of x86 servers. VMW_ORCT_CSCO_ 208

No.	'111 Patent Claim 24	VMware NSX
		Manage CPU usage in OVS transport nodes Open vSwitch package handling is multi-threaded. You can adjust how CPU is allotted by changing the number of threads an OVS transport nodes uses. There are three types of threads in the connection set up and packet handling process. • The main thread handles OVS configuration and OpenFlow protocol processing, manages all other threads, and performs other switch functions. You will always have only one main thread. • The dispatcher thread sends packets from kernel to the miss_handler threads. You will always have only one dispatch thread. • The miss_handler threads get packets from the dispatcher, figure out how to route the packets, and forward the packets. By default, there is one miss_handler thread, but you can edit this setting. The more threads there are, the more CPU the connection set up process uses. To dedicate more CPU to other uses, such as virtual machines, lower the number of miss_handler threads. Set this in the OVS configuration for each NSX transport node using the n-handler-threads setting. For example, the following command results in 3 threads: the default threads and one miss_handler thread: `ovs-vsctl set Open_vSwitch . other_config:n-handler-threads=1` The default value for n-handler-threads is the number of cores minus 2. On a dual-core device there will be one miss_handler thread; on a quad-core device there will be 2 miss-handler threads by default. VMW_ORCT_CSCO_ 213

Controller Cluster Node Connectivity Requirements

Port/Protocol	Service	Listen interface	Comments
22/TCP	SSH based CLI Access	All interfaces	
123/UDP	NTP	All interfaces	Network Time Protocol uses this port.
161/UDP	SNMP	All interfaces	This port is Optional. This port is only required when SNMP services are used.
443/TCP	API Provider over HTTPS	ethX	Access to this port is authenticated via NSX administrator credentials.
2878/TCP	Distributed Persistence Server	All interfaces	This port is accessed without any additional authentication and traffic is passed unencrypted. This port also maintains active connection to peer Controller nodes.
2888/TCP	Distributed Persistence Server – Client	ethX	This port is accessed without any additional authentication and traffic is passed unencrypted. This port also maintains active connection to peer Controller nodes.
3888/TCP	Distributed Persistence Server – Synchronizer	All interfaces	This port is accessed without any additional authentication and traffic is passed unencrypted. This port also maintains active connection to peer Controller nodes.
6632/TCP	Open vSwitch Management Service	ethX	All traffic is encrypted using pre-provided certificate. This port also maintains active connection to each of the Service, Gateway and hypervisor nodes.
6633/TCP	OpenFlow Service	ethX	All traffic is encrypted using pre-provided certificate. This port also maintains active connection to each of the Service, Gateway and hypervisor nodes.
7000/TCP	Distributed Hash Table Server	ethX	This port is accessed without any additional authentication and traffic is passed unencrypted. This port also maintains active connection to peer Controller nodes.

No.	'111 Patent Claim 24	VMware NSX
		VMW_ORCT_CSCO_ 70 *Transport Connectors* Transport nodes communicate with each other to implement logical networks. Transport connectors define the encapsulation mechanism and identifiers used for that communication. For example, a hypervisor that implements overlay logical networks using an L2-in-L3 tunneling protocol such as GRE would have a *GREConnector* that indicates the IP address it will use for tunneled traffic of that type. Similarly, a hypervisor that implements bridged logical networks would have a BridgeConnector identifying the bridge-ID that will be used to transmit unencapsulated traffic. The supported transport connectors include: • **GREConnector / STTConnector**—uses unencrypted GRE or STT tunneling to create logical networks that overlay on top of the physical network topology. • **IPsecGREConnector / IPsecSTTConnector**—uses GRE or STT tunneling with IPsec to create a secure logical network overlay on top of the physical network topology. • **BridgeConnector**—no encapsulation is used.The logical network is mapped directly onto the transport network topology (or optionally, a VLAN in the transport network). VMW_ORCT_CSCO_ 71 Since Gateways and Service Nodes are only used in overlay logical networks, they only use connectors that provide tunneling capability via protocols like GRE or STT. A Gateway or Service Node has a connector for each IP address that it uses to tunnel packets. For a Gateway to establish a tunnel with a Service Node, it must have a tunnel connector of the same type as the Service Node on a shared transport zone. VMW_ORCT_CSCO_ 84 L2 Gateway Services also participate in 802.1D spanning tree protocol (STP) and monitor STP BPDU packets on the network to detect physical network topology changes that might lead to a loop. L2 Gateway Services bridge-ids are configured so that L2 Gateway Services are never elected as STP root bridges. VMW_ORCT_CSCO_ 999

No.	'111 Patent Claim 24	VMware NSX
		

No.	'111 Patent Claim 27	VMware NSX
27	The method according to claim 1, wherein the network node comprises a router, a switch, or a bridge.	VMware NSX System discloses the method according to claim 1, wherein the network node comprises a router, a switch, or a bridge. For example, VMware NSX System discloses a virtual switch or vSwitch that implements the NSX data plane. *See supra* at Claim 1. NSX Design Guide at 8-9

No.	'111 Patent Claim 27	VMware NSX
		2.1.1 Data Plane The NSX data plane is implemented by the NSX vSwitch. The vSwitch in NSX for vSphere is based on the VDS with additional components added to enable rich services. The add-on NSX components include kernel modules distributed as VMware installation bundles (VIBs). These modules run within the hypervisor kernel, providing services including distributed routing, distributed firewall, and VXLAN to VLAN bridging. The NSX VDS abstracts the physical network, providing access-level switching in the hypervisor. This is central to network virtualization as it enables logical networks that are independent of physical constructs (e.g., VLANs). The NSX vSwitch enables support for overlay networking with the use of the VXLAN protocol and centralized network configuration. Overlay networking with NSX enables the following capabilities: • Creation of a flexible logical layer 2 (L2) overlay over existing IP networks on existing physical infrastructure. • Agile provisioning of communication – both east–west and north–south – while maintaining isolation between tenants. • Application workloads and VMs that are agnostic of the overlay network, operating as if they were connected to a physical network. • Massive scalability of hypervisors. Operational features – port mirroring, NetFlow/IPFIX, Traceflow, configuration backup and restore, Network Health Check, QoS, and LACP – provide a comprehensive toolkit for traffic management, monitoring, and troubleshooting within a virtual network. The data plane also consists of gateway devices that can provide communication from the logical networking space to the physical network (e.g., VXLAN to VLAN). This functionality can happen at either L2 (NSX bridging) or at L3 (NSX routing). NSX Design Guide at 30

DFW rules are enforced as follows:

- Rules are processed in top-to-bottom ordering.
- Each packet is checked against the top rule in the rule table before moving down the subsequent rules in the table.
- The first rule in the table that matches the traffic parameters is enforced. No subsequent rules can be enforced as the search is then terminated for that packet.

Because of this behavior, it is always recommended to put the most granular policies at the top of the rule table. This will ensure they will be enforced before more specific rules.

The DFW default policy rule, located at the bottom of the rule table, is a "catch-all" rule; packets not matching any other rules will be enforced by the default rule. After the host preparation operation, the DFW default rule is set to 'allow' action. This ensures that VM-to-VM communication is not broken during staging or migration phases. It is a best practice to then change this default rule to 'block' action and enforce access control through a positive control model (i.e., only traffic defined in the firewall policy is allowed onto the network).

Figure 22 steps through policy rule lookup and packet flow:

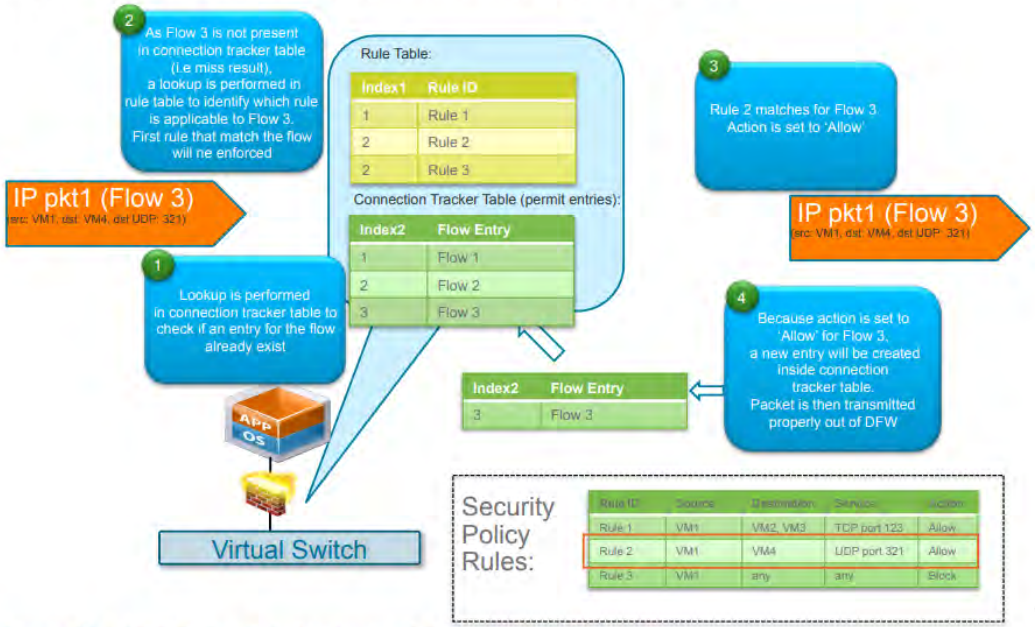
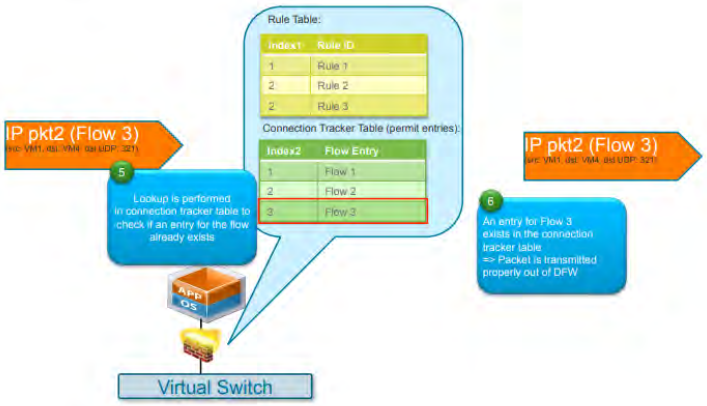


Figure 22 – DFW Policy Rule Lookup and Packet – First Packet

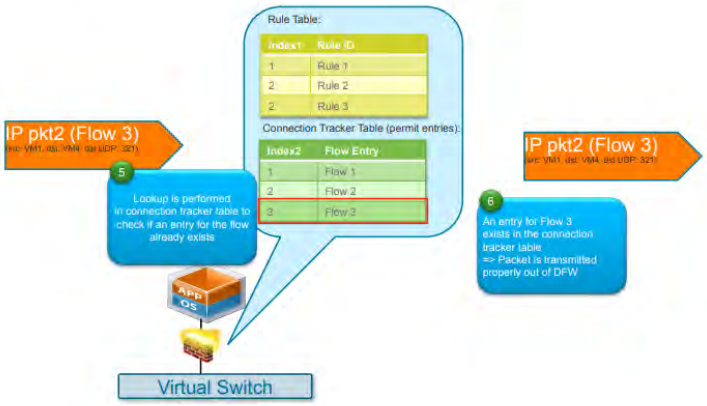
Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 2094 of 2137

No.	'111 Patent Claim 27	VMware NSX
		NSX Design Guide at 30-31 An IP packet identified as pkt1 that matches rule number 2. The order of operation is the following: 30 1. A lookup is performed in the connection tracker table to determine if an entry for the flow already exists. 2. As flow 3 is not present in the connection tracker table, a lookup is performed in the rule table to identify which rule is applicable to flow 3. The first rule that matches the flow will be enforced. 3. Rule 2 matches for flow 3. The action is set to 'Allow'. 4. Because the action is set to 'Allow' for flow 3, a new entry will be created inside the connection tracker table. The packet is then transmitted out of DFW.  Figure 23 – DFW Policy Rule Lookup and Packet – Subsequent Packets.

No.	'111 Patent Claim 27	VMware NSX
		NSX Design Guide at 52 Distributed routing is provided by a logical element called Distributed Logical Router (DLR). The DLR is essentially a router with directly connected interfaces to all hosts where VM connectivity is required. The supervisory function (i.e., control plane) to control the forwarding is imported from a control VM. The DLR consists of two primary components: • **Control Plane:** The control plane is provided by the DLR Control VM and NSX controller. The control plane supports dynamic routing protocols (e.g., BGP, OSPF), exchanges routing updates with the next layer 3 hop device (e.g., NSX Edge), and communicates with the NSX manager and the controller cluster. High availability for the DLR control VM is supported through an active/standby configuration. • **Data Plane:** DLR kernel modules (e.g., VIBs) are installed on the ESXi hosts part of the NSX domain. The kernel modules are similar to the line cards in a modular chassis supporting layer 3 routing. These kernel modules have a routing information base (RIB) that is pushed through the controller cluster. The traditional data plane functionality of route and ARP lookups is performed by the kernel modules. The kernel modules are equipped with logical interfaces (LIFs) connecting to the different logical switches. Each LIF has an IP address representing the default IP gateway for its logical L2 segment as well as a vMAC address. The IP address is unique per LIF and remains same where the logical switch exists. The vMAC associated with each LIF remains the consistent in each hypervisor as well and thus during the vMotion, the default gateway and MAC remains the same. Switching at 1

No.	'111 Patent Claim 27	VMware NSX
		Layer-2 forwarding VMware NSX supports traditional layer-2 segments with proper flooding of BUM (Broadcast, Unknown unicast, Multicast) frames. NSX controller downloads forwarding entries to individual virtual switches, either through OpenFlow (NSX for multiple hypervisors) or a proprietary protocol (NSX for vSphere). The forwarding entries map destination VM MAC addresses into destination hypevisor (or gateway) IP addresses. On top of static forwarding entries downloaded from the controller, virtual switches perform dynamic MAC learning for MAC addresses reachable through layer-2 gateways.

No.	'111 Patent Claim 28	VMware NSX
28	The method according to claim 1, wherein the packet network is an Internet Protocol (IP) network, and the packet is an IP packet.	VMware NSX System discloses the method according to claim 1, wherein the packet network is an Internet Protocol (IP) network, and the packet is an IP packet. For example, VMware NSX System discloses an IP network in which the packets transmitted are IP packets. *See supra* at Claim 1. NSX Design Guide at 27 When vCenter containers are used in the DFW policy rules, VMtools must be installed on the guest VMs. VMtools has visibility of the IP address of the VM – whether dynamically provided through DHCP or statically configured on the VM by the administrator – and provides this information down to the DFW engine that operates based on MAC, IP, and TCP/UDP port fields. If the DFW policy rules use only IP information (e.g., host IP, subnet IP, or IP sets) then the presence of VMtools on guest VM is not required. **Note:** With NSX release 6.2, VMtools is not mandatory. See the "Deploying Distributed Firewall" section for further details.

No.	'111 Patent Claim 28	VMware NSX
		NSX Design Guide at 30-31 An IP packet identified as pkt1 that matches rule number 2. The order of operation is the following: 30 1. A lookup is performed in the connection tracker table to determine if an entry for the flow already exists. 2. As flow 3 is not present in the connection tracker table, a lookup is performed in the rule table to identify which rule is applicable to flow 3. The first rule that matches the flow will be enforced. 3. Rule 2 matches for flow 3. The action is set to 'Allow'. 4. Because the action is set to 'Allow' for flow 3, a new entry will be created inside the connection tracker table. The packet is then transmitted out of DFW.  Figure 23 – DFW Policy Rule Lookup and Packet – Subsequent Packets.

No.	'111 Patent Claim 28	VMware NSX
		VMW_ORCT_CSCO_ 1386 **Configure OSPF Protocol** NSX Edge supports OSPF, an interior gateway protocol that routes IP packets only within a single routing domain. It gathers link state information from available routers and constructs a topology map of the network. The topology determines the routing table presented to the Internet Layer, which makes routing decisions based on the destination IP address found in IP packets. OSPF routing policies provide a dynamic process of traffic load balancing between routes of equal cost. An OSPF network is divided into routing areas to optimize traffic. An area is a logical collection of OSPF networks, routers, and links that have the same area identification. Areas are identified by an Area ID. VMW_ORCT_CSCO_ 1902 **Manage OSPF Routes for NSX Edge** NSX Edge supports OSPF, an interior gateway protocol that routes IP packets only within a single routing domain. It gathers link state information from available routers and constructs a topology map of the network. The topology determines the routing table presented to the Internet Layer, which makes routing decisions based on the destination IP address found in IP packets. OSPF routing policies provide a dynamic process of traffic load balancing between routes of equal cost. An OSPF network is divided into routing areas to optimize traffic. An area is a logical collection of OSPF networks, routers, and links that have the same area identification. Areas are identified by an Area ID.

No.	'111 Patent Claim 29	VMware NSX
29	The method according to claim 28, wherein the packet network is an Transmission Control Protocol (TCP) network, and the	VMware NSX System 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. For example, VMware NSX System discloses packets with TCP fields, that may be transmitted in a TCP network. *See supra* at Claim 28.

No.	'111 Patent Claim 29	VMware NSX
	packet is an TCP packet.	NSX Design Guide at 27 When vCenter containers are used in the DFW policy rules, VMtools must be installed on the guest VMs. VMtools has visibility of the IP address of the VM – whether dynamically provided through DHCP or statically configured on the VM by the administrator – and provides this information down to the DFW engine that operates based on MAC, IP, and TCP/UDP port fields. If the DFW policy rules use only IP information (e.g., host IP, subnet IP, or IP sets) then the presence of VMtools on guest VM is not required. **Note:** With NSX release 6.2, VMtools is not mandatory. See the "Deploying Distributed Firewall" section for further details.

No.	'111 Patent Claim 30	VMware NSX
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;	VMware NSX System 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. For example, VMware NSX System discloses transmitting additional packets of a flow between VMs in which the vSwitch receives additional packets and checks to see if there is a match in the rule table. *See supra at* Claim 1, 1[c]. NSX Design Guide at 31

No.	'111 Patent Claim 30	VMware NSX
		Subsequent packets are processed in this order: 5. A lookup is performed in the connection tracker table to check if an entry for the flow already exists. 6. An entry for flow 3 exists in the connection tracker table. The packet is transmitted out of DFW As DFW fully supports vMotion – either automatic vMotion with DRS or manual vMotion - the rule table and the connection tracker table always follow the VM during vMotion operation. This ensures that there is no traffic disruption during workload moves while connections initiated before vMotion remain intact after the vMotion is completed. DFW brings VM movement freedom while ensuring continuous network traffic protection. This functionality is not dependent on the availability of controllers or NSX manager. NSX DFW offers a paradigm shift that was not previously possible. Security services are no longer dependent on the network topology as DFW security enforcement is completely decoupled from logical network topology. VMW_ORCT_CSCO_ 88

MDI Transport Data Forwarding

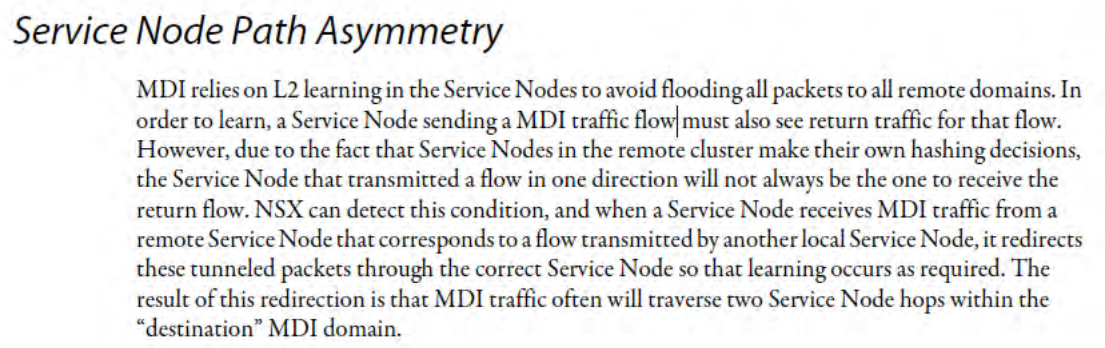
Data plane communication between logical switches managed by different NSX Controller Clusters occurs via overlay (i.e., STT or GRE) tunnels between Service Nodes in the respective domains. MDI data traffic flows from the source OVS edge switch (hypervisor or NSX Gateway), through a Service Node in the source cluster, to a Service Node in the second cluster, then to the destination OVS edge switch.

The Service Nodes in the source domain perform standard L2 learning to identify which remote domain a destination MAC address is located in. If a packet is broadcast, multicast, or the destination is unknown, the Service Node will flood the packet to each of the remote domains (subject to the loop prevention mechanisms described below). Each Service Node participating in MDI performs its own L2 learning, independent of other Service Nodes in the same domain.

Per-flow hashing is used to spread MDI data traffic across all available Service Nodes, for both load sharing and high-availability with respect to Service Node failures. This hashing takes place at two points in the MDI forwarding process: first when the source OVS edge switch (hypervisor or Gateway) chooses a Service Node in the source domain to handle a flow, then again when the Service Node in the source domain chooses a Service Node in the remote domain as the tunnel destination. In both cases, Service Node failures are detected using the standard OVS tunnel monitoring probes, allowing fast-failover in the case of a hardware event on a Service Node.

A single STT or GRE tunnel between a pair of Service Nodes in different domains can carry traffic from multiple sets of interconnected logical switches, as all packets traversing such a link have an *interconnect context-ID* value stored in the tunnel header. The context-ID is a shared identifier that identifies a pair of switches connected over MDI. Each NSX domain uses the context-ID to dispatch MDI traffic to the correct logical switch: a Service Node in the source domain tags each MDI packet with a context-ID based on the source logical switch, and the Service Node in the destination domain uses the context-ID to map an incoming MDI packet to the correct destination logical switch. The context-ID also scopes the L2 learning (described above), so that different logical switches can use overlapping MAC addresses as long as those switches are not interconnected with MDI. How the context-ID is chosen and associated with the logical switch is described in the following section on MDI API Entities.

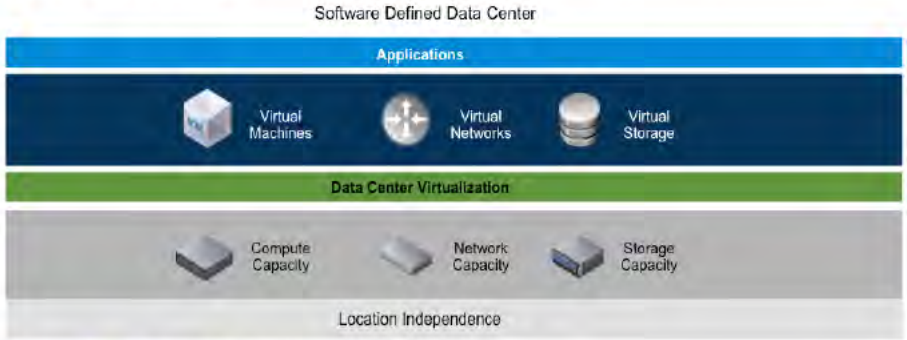
MDI requires meshed connectivity in the logical layer, meaning that any interconnection of three or more logical switches creates a loop in the logical L2 topology. To prevent broadcast, multicast, and destination unknown packets from looping between logical switches in different domains, MDI employs conservative "single-hop" forwarding logic such that packets received from a Service Node in a different domain are forwarded only to local OVS edge switches, but never to Service Nodes in other domains. This avoid loops, but means that each NSX domain must have L3 tunnel connectivity to each other domain participating in MDI (i.e., the tunnel connectivity must also be a mesh).

No.	'111 Patent Claim 30	VMware NSX
		VMW_ORCT_CSCO_ 92  *Service Node Path Asymmetry* MDI relies on L2 learning in the Service Nodes to avoid flooding all packets to all remote domains. In order to learn, a Service Node sending a MDI traffic flow must also see return traffic for that flow. However, due to the fact that Service Nodes in the remote cluster make their own hashing decisions, the Service Node that transmitted a flow in one direction will not always be the one to receive the return flow. NSX can detect this condition, and when a Service Node receives MDI traffic from a remote Service Node that corresponds to a flow transmitted by another local Service Node, it redirects these tunneled packets through the correct Service Node so that learning occurs as required. The result of this redirection is that MDI traffic often will traverse two Service Node hops within the "destination" MDI domain.
30[b]	checking, by the network node, if any one of the one or more additional packets satisfies the criterion;	VMware NSX System discloses checking, by the network node, if any one of the one or more additional packets satisfies the criterion. *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	VMware NSX System 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. *See supra at* Claim 1[e], 30[a].

No.	'111 Patent Claim 30	VMware NSX
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.	VMware NSX System 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. *See supra at* Claim 1[f], 30[a].

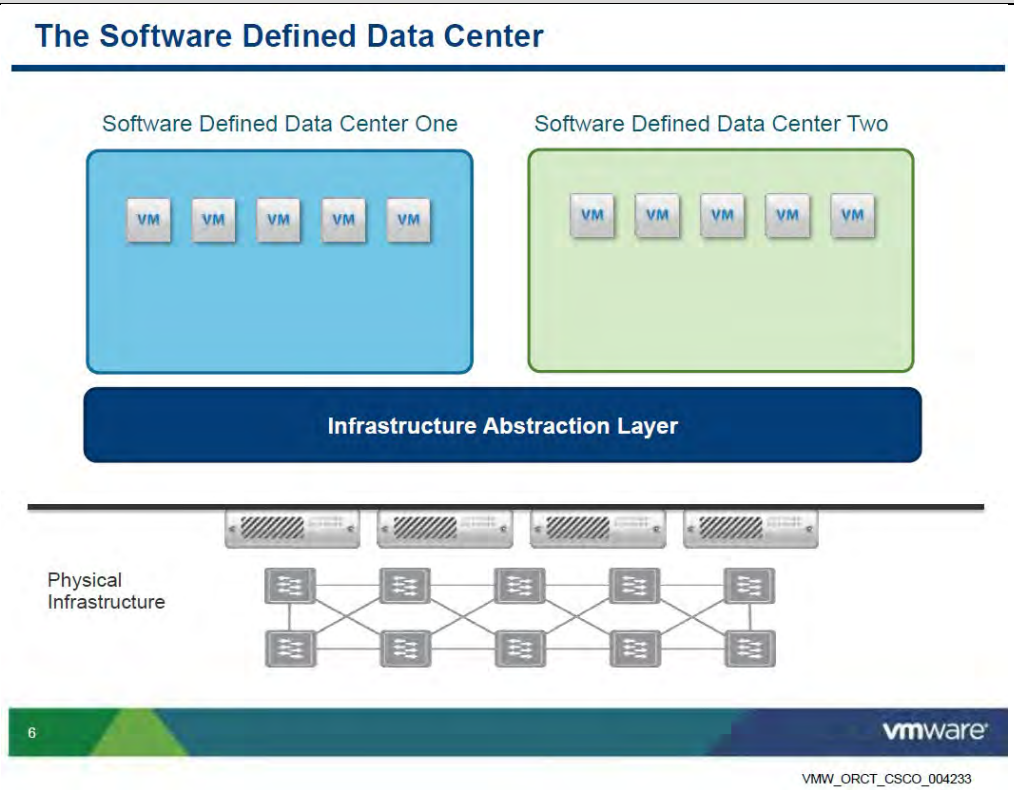
No.	'111 Patent Claim 31	VMware NSX
31[a]	The method according to claim 1, wherein the packet network is a Software Defined Network (SDN),	VMware NSX System discloses the method according to claim 1, wherein the packet network is a Software Defined Network (SDN). For example, VMware NSX System discloses a network topology deployed by a software defined data center architecture. *See supra* at Claim 1. NSX Design Guide at 4

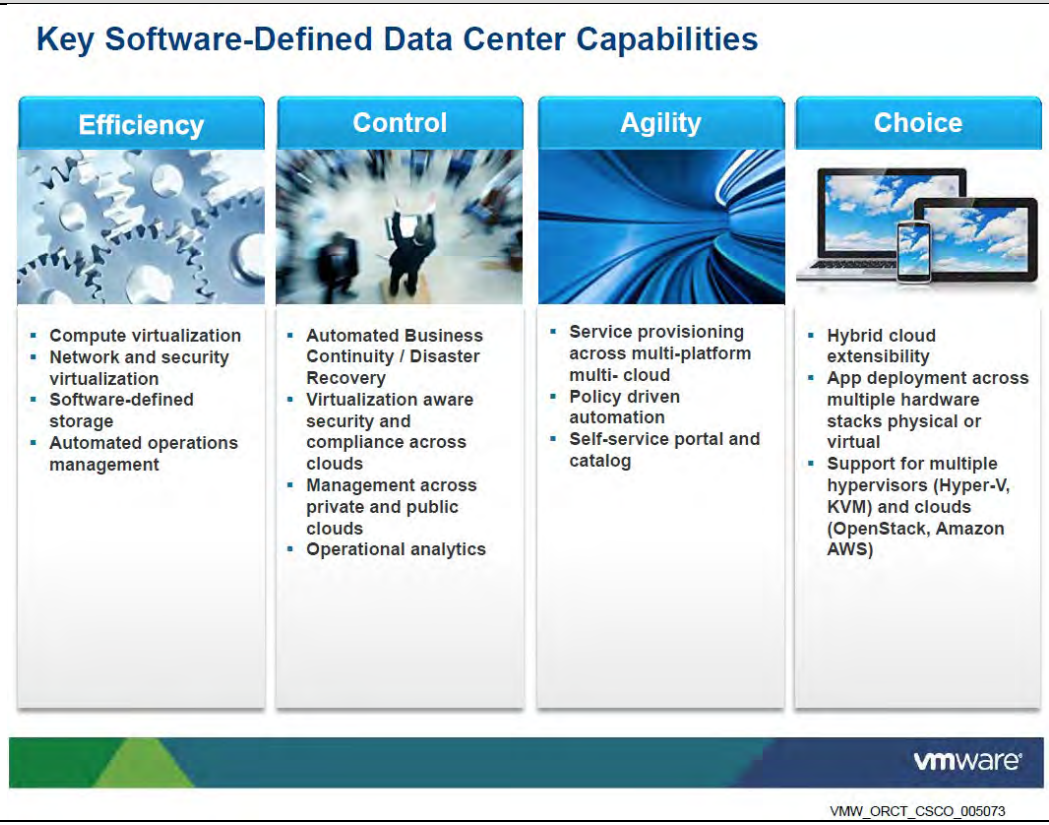
No.	'111 Patent Claim 31	VMware NSX
		1 Overview IT organizations have gained significant benefits as a direct result of server virtualization. Tangible advantages of server consolidation include reduced physical complexity, increased operational efficiency, and simplified dynamic re-purposing of underlying resources. These technology solutions have delivered on their promise of helping IT to quickly and optimally meet the needs of increasingly dynamic business applications. VMware's Software Defined Data Center (SDDC) architecture moves beyond the server, extending virtualization technologies across the entire physical data center infrastructure. VMware NSX, the network virtualization platform, is a key product in the SDDC architecture. With VMware NSX, virtualization now delivers for networking what it has already delivered for compute. Traditional server virtualization programmatically creates, snapshots, deletes, and restores virtual machines (VMs); similarly, network virtualization with VMware NSX programmatically creates, snapshots, deletes, and restores software-based virtual networks. The result is a completely transformative approach to networking, enabling orders of magnitude better agility and economics while also vastly simplifying the operational model for the underlying physical network. NSX is a completely non-disruptive solution which can be deployed on any IP network from any vendor – both existing traditional networking models and next generation fabric architectures. The physical network infrastructure already in place is all that is required to deploy a software-defined data center with NSX. NSX Design Guide at 80

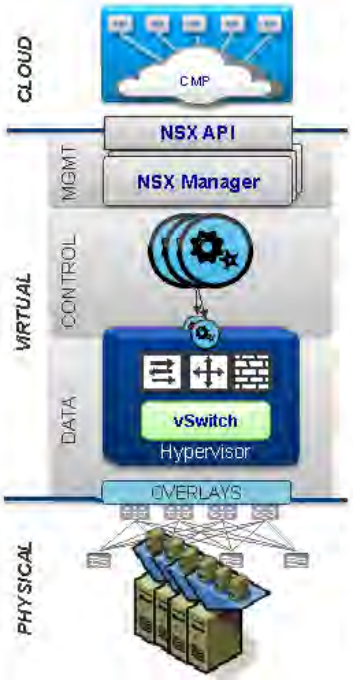
No.	'111 Patent Claim 31	VMware NSX
		NSX introduces a mechanism for deploying security services independent of the underlying topology. Traditional services like firewall or advanced services like agentless AV, L7 firewall, IPS, and traffic monitoring can be deployed independent of the underlying physical or logical networking topologies. This enables a significant shift in planning and deploying services in the datacenter. Services no longer need to be tied down to networking topology. NSX provides a framework called Service Composer to enable deployment of security services for the datacenter.  Figure 67 – Software Defined Data Center Switching at 1 Layer-2 forwarding VMware NSX supports traditional layer-2 segments with proper flooding of BUM (Broadcast, Unknown unicast, Multicast) frames. NSX controller downloads forwarding entries to individual virtual switches, either through OpenFlow (NSX for multiple hypervisors) or a proprietary protocol (NSX for vSphere). The forwarding entries map destination VM MAC addresses into destination hypevisor (or gateway) IP addresses. On top of static forwarding entries downloaded from the controller, virtual switches perform dynamic MAC learning for MAC addresses reachable through layer-2 gateways.

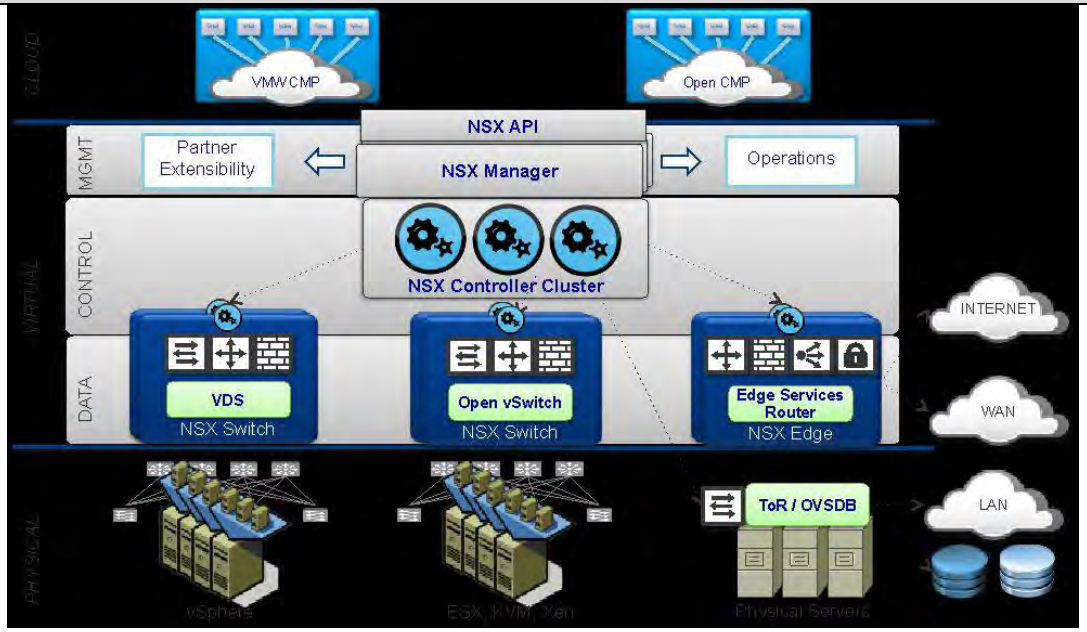
No.	'111 Patent Claim 31	VMware NSX
		VMW_ORCT_CSCO_ 485-486 *What is Network Virtualization?* Virtualization reaches its full potential when all data center resources -- including networking -- are virtualized. Without network virtualization, each VM's addressing, connectivity, and associated network services (for example NAT configuration, security settings, and QoS policies) are defined in the network hardware, tying that VM to its current location in the data center. Without network virtualization, each VM's IP address and the set of L2-adjacent VMs depend on the physical location of the hypervisor and the physical network to which it is attached. Network virtualization overcomes this limitation by allowing VMs to connect to *logical networks* rather than attaching directly to physical networking hardware, doing away with the in-hardware network configurations that tie a VM to its physical location in the network. Network virtualization provides: • **Forwarding-plane virtualization**—With NSX network virtualization, each tenant in the data center occupies its own logical network. The logical network appears to each VM as a dedicated switching device and network, providing full L2 and L3 service models. Because the logical network is defined in the data center's virtualization software (hence "software defined networking" or "SDN") rather than on its networking hardware, it provides address isolation from other tenants' logical networks that share the physical network. • **Management-plane virtualization**—Each VM's network settings (for example, NAT policies, QoS policies, and port statistics) automatically follow the VM wherever it is provisioned or migrated, and network management is done through a single management plane, the NVP API. The NVP API enables automated provisioning and monitoring of all logical networks and their underlying transport nodes. Cloud providers can use the NVP API to build tenant-facing management dashboards that give each tenant the ability to create and manage his or her logical network. VMW_ORCT_CSCO_ 1349

No.	'111 Patent Claim 31	VMware NSX
		VMware NSX® is a software networking and security virtualization platform that delivers the operational model of a virtual machine for the network. Virtual networks reproduce the Layer2 - Layer7 network model in software, allowing complex multi-tier network topologies to be created and provisioned programmatically in seconds. NSX also provides a new model for network security. Security profiles are distributed to and enforced by virtual ports and move with virtual machines. NSX supports VMware's software-defined data center strategy. By extending the virtualization capabilities of abstraction, pooling and automation across all data center resources and services, the software-defined data center architecture simplifies and speeds the provisioning and management of compute, storage and networking resources through policy-driven automation. By virtualizing the network, NSX delivers a new operational model for networking that breaks through current physical network barriers and enables data center operators to achieve better speed and agility with reduced costs. NSX includes a library of logical networking services - logical switches, logical routers, logical firewalls, logical load balancers, logical VPN, and distributed security. You can create custom combinations of these services in isolated software-based virtual networks that support existing applications without modification, or deliver unique requirements for new application workloads. Virtual networks are programmatically provisioned and managed independent of networking hardware. This decoupling from hardware introduces agility, speed, and operational efficiency that can transform datacenter operations. VMW_ORCT_CSCO_4233

No.	'111 Patent Claim 31	VMware NSX
		 VMW_ORCT_CSCO_5073

No.	'111 Patent Claim 31	VMware NSX
		
31[b]	the packet is routed as part of a data plane and	VMware NSX System discloses the packet is routed as part of a data plane. For example, VMware NSX routing packets via the vSwitch, which implements the data plane. Introducing VMware NSX at 3-4

No.	'111 Patent Claim 31	VMware NSX
		VMware NSX Architecture and Design As depicted in the diagram below, the VMware NSX solution resides in the virtualization layer, providing L2-L7 network services to the cloud consumption layer above, and mapping these services onto the physical infrastructure below. Independence between the cloud and virtualization layers is achieved by providing a REST API exposing the network services to any upstream provisioning platform. Likewise, vSwitch overlays such as VXLAN provide independence to deploy NSX on any physical IP network infrastructure. This brings us to the network virtualization layer. For each capability, be it switching, routing or firewall, the services are provided via NSX APIs, and realized using a three-tiered design pattern encompassing the management plane, control plane and data plane. The NSX Manager internally maps the APIs onto the control plane. **The controller cluster is the work horse of the system**, handling real time mapping between the system's desired state and the running state, which it communicates to the control plane agent(s) present per hypervisor. The local information is now used to set up the appropriate switching, routing, or firewall tables and contexts. The appropriate data plane function now proceeds with high-performance, in a scale out fashion across the virtual plane.  **NSX: architectural context & design pattern**: Cloud Management Platform consumes services The REST **API** abstracts underlying services The **Manager** cluster maps services to the controllers The control plane consists of a master **controller** cluster delegating tasks to control plane agents in each hypervisor (or controlled device) The agent performs the local task of activating the **data plane** action in the hypervisor e.g. switching, routing or firewalls **Overlays** de-couple the virtual plane from the underlying physical network fabric The network fabric provides connectivity to physical servers, hosting multiple VMs connected to a programmable vSwitch. IP connectivity is the only requirement of the physical network fabric

No.	'111 Patent Claim 31	VMware NSX
		 NSX Design Guide at 8-9

2.1.1 Data Plane

The NSX data plane is implemented by the NSX vSwitch. The vSwitch in NSX for vSphere is based on the VDS with additional components added to enable rich services. The add-on NSX components include kernel modules distributed as VMware installation bundles (VIBs). These modules run within the hypervisor kernel, providing services including distributed routing, distributed firewall, and VXLAN to VLAN bridging.

The NSX VDS abstracts the physical network, providing access-level switching in the hypervisor. This is central to network virtualization as it enables logical networks that are independent of physical constructs (e.g., VLANs).

The NSX vSwitch enables support for overlay networking with the use of the VXLAN protocol and centralized network configuration. Overlay networking with NSX enables the following capabilities:

- Creation of a flexible logical layer 2 (L2) overlay over existing IP networks on existing physical infrastructure.
- Agile provisioning of communication – both east–west and north–south – while maintaining isolation between tenants.
- Application workloads and VMs that are agnostic of the overlay network, operating as if they were connected to a physical network.
- Massive scalability of hypervisors.

Operational features – port mirroring, NetFlow/IPFIX, Traceflow, configuration backup and restore, Network Health Check, QoS, and LACP – provide a comprehensive toolkit for traffic management, monitoring, and troubleshooting within a virtual network.

The data plane also consists of gateway devices that can provide communication from the logical networking space to the physical network (e.g., VXLAN to VLAN). This functionality can happen at either L2 (NSX bridging) or at L3 (NSX routing).

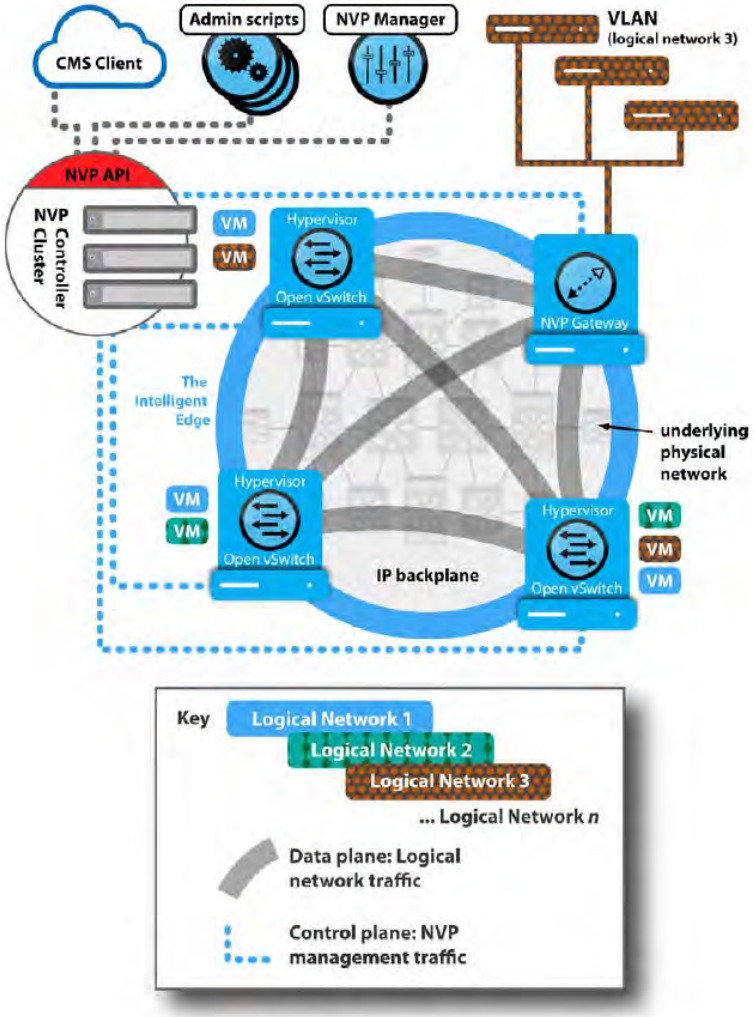
Orckit Exhibit 2018
Cisco Systems v. Orckit Corp.
IPR2023-00401, Page 2113 of 2137

No.	'111 Patent Claim 31	VMware NSX
		NSX Design Guide at 52 Distributed routing is provided by a logical element called Distributed Logical Router (DLR). The DLR is essentially a router with directly connected interfaces to all hosts where VM connectivity is required. The supervisory function (i.e., control plane) to control the forwarding is imported from a control VM. The DLR consists of two primary components: • **Control Plane:** The control plane is provided by the DLR Control VM and NSX controller. The control plane supports dynamic routing protocols (e.g., BGP, OSPF), exchanges routing updates with the next layer 3 hop device (e.g., NSX Edge), and communicates with the NSX manager and the controller cluster. High availability for the DLR control VM is supported through an active/standby configuration. • **Data Plane:** DLR kernel modules (e.g., VIBs) are installed on the ESXi hosts part of the NSX domain. The kernel modules are similar to the line cards in a modular chassis supporting layer 3 routing. These kernel modules have a routing information base (RIB) that is pushed through the controller cluster. The traditional data plane functionality of route and ARP lookups is performed by the kernel modules. The kernel modules are equipped with logical interfaces (LIFs) connecting to the different logical switches. Each LIF has an IP address representing the default IP gateway for its logical L2 segment as well as a vMAC address. The IP address is unique per LIF and remains same where the logical switch exists. The vMAC associated with each LIF remains the consistent in each hypervisor as well and thus during the vMotion, the default gateway and MAC remains the same. VMW_ORCT_CSCO_ 56

NSX Architecture: Implementing Network Virtualization

Administrators use the NVP API and Manager to configure logical networks and transport nodes that forward network traffic. NSX manages transport nodes via its control plane, and VMs send traffic on data plane.

NSX Control Plane and Data Plane



No.	'111 Patent Claim 31	VMware NSX
		VMW_ORCT_CSCO_ 57-58

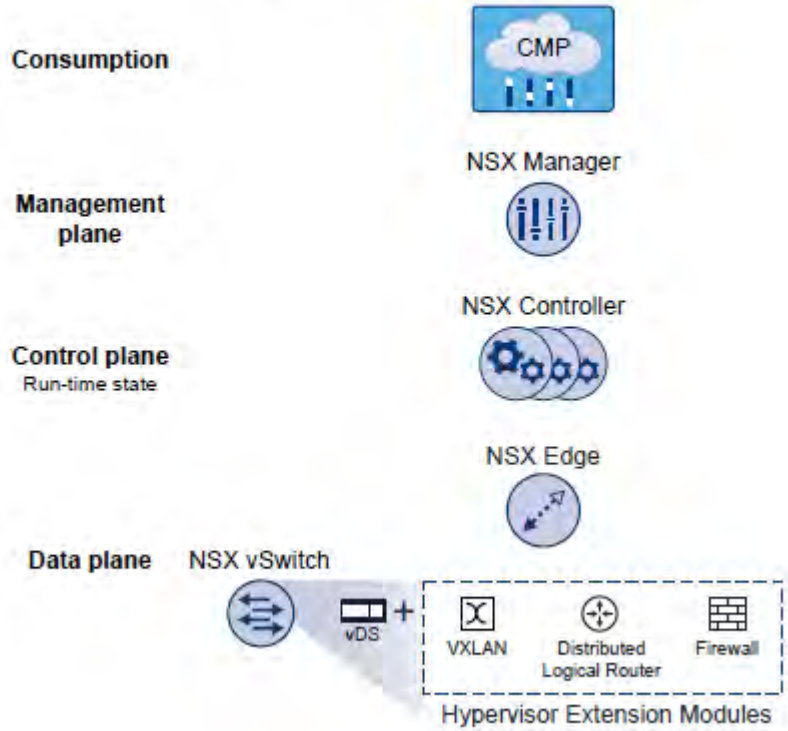
NSX Data Plane

The NSX data plane is implemented by access-layer switching devices (hypervisor vSwitches) and NSX appliances (NSX Service Nodes, NSX Gateways, third-party hardware L2 Gateways) that are managed by the Controller Cluster. We refer to these access-layer devices as "transport nodes." Traditional network hardware (not managed by NSX) provides the underlying bandwidth that connects the transport nodes. NSX creates network overlays that implement the logical network view atop the physical network.

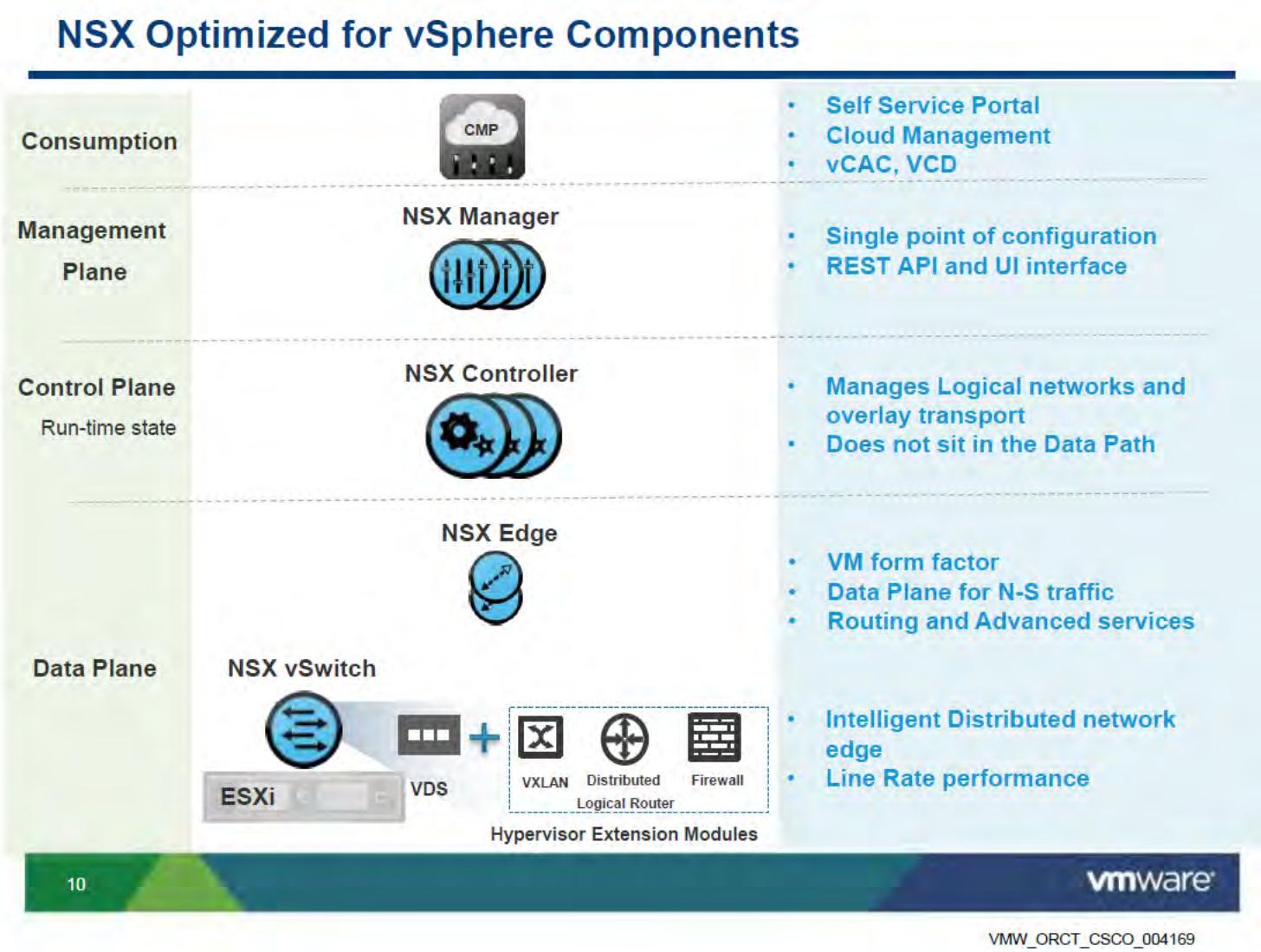
Transport nodes support the OVS management interface, an open protocol that enables fine-grained control of and visibility into a switch's packet forwarding behavior. Endpoints connected to OVS-enabled access switches can be "attached" to a logical network to send and receive traffic. For example:

- **Hypervisor** — An OVS-enabled virtual switch is installed on each hypervisor, allowing VM endpoints running on the hypervisor to be attached to a logical network in NSX.
- **NSX Gateway Service** — An NSX Gateway Service consists of one or more NSX Gateways that attach a logical network to a physical network not managed by NSX. Each Gateway Service can operate as an L2 Gateway Service sending traffic to a physical L2 segment, or as an L3 Gateway Service mapped to a physical router port. For example, a cloud tenant might use an NSX Gateway Service to connect his logical network to his non-cloud, physical hosting environment located elsewhere in the data center or to his on-campus data center. Each Gateway in the service

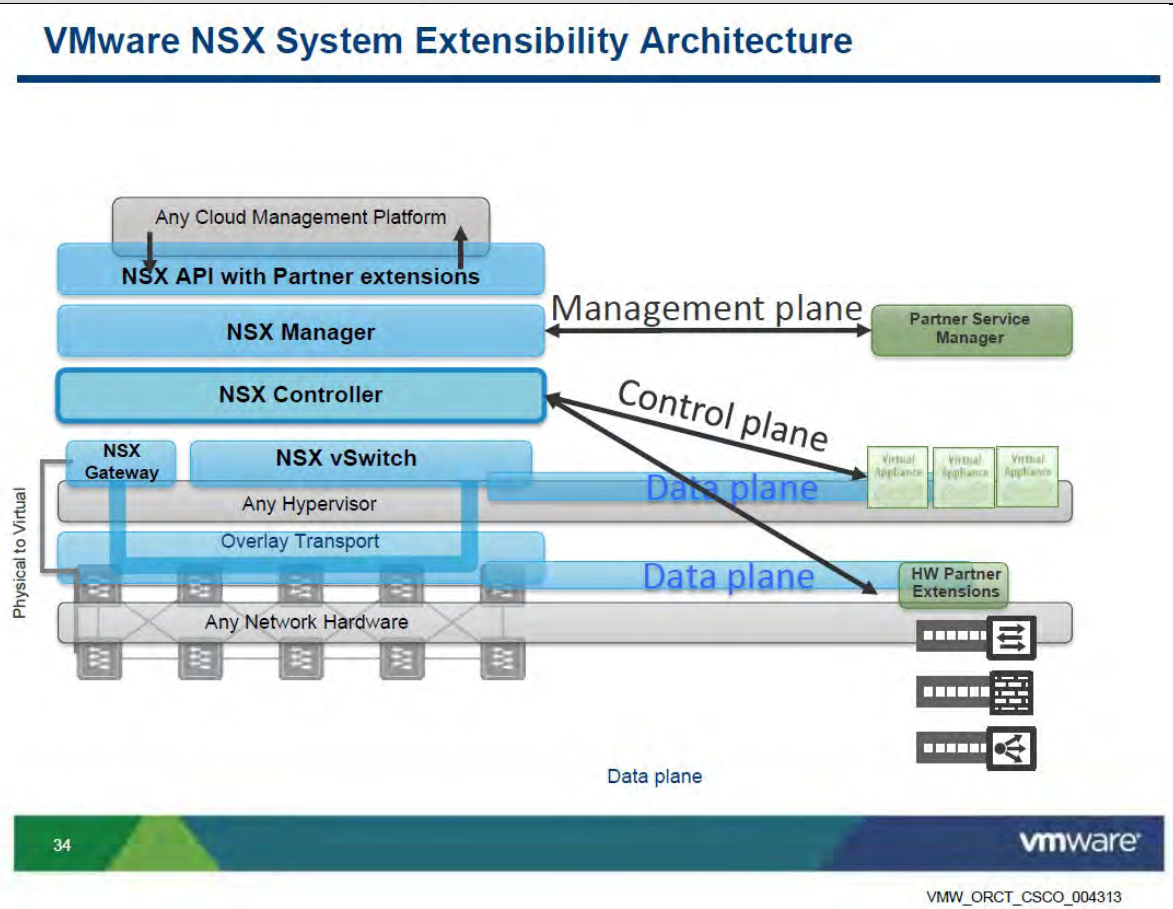
No.	'111 Patent Claim 31	VMware NSX
		is a virtual appliance running OVS, or a physical VTEP-enabled appliance. If the Gateway Service includes more than one Gateway, it provides a high-availability (HA) connection to the physical network. Neither OVS nor OpenFlow support is required on the physical network devices that interconnect the access switches. Any network hardware that provides unicast IP connectivity between access switches is sufficient. In the data plane, logical switches can be implemented using different encapsulation mechanisms, which result in two different types of logical networks: • **Overlay logical networks** are implemented using L2-in-L3 tunneling, meaning that logical network topology can be completely decoupled from transport network topology. • **Bridged logical networks** do not use encapsulation. Instead, VM traffic is sent directly on the physical network, similar to traditional hypervisor networking. **NSX Service Nodes:** For overlay logical networks, NSX employs **NSX Service Nodes**, OVS-enabled x86 appliances that are managed by the Controller Cluster to provide extra packet processing capacity for logical networks. For example, Service Nodes assist with the packet replication required for logical network broadcast/multicast and unknown unicast flooding in overlay logical networks. The Controller Cluster manages all NSX Service Node appliances as a resource pool, automatically spreading traffic across the available Service Nodes and masking individual node failures. VMW_ORCT_CSCO_2160

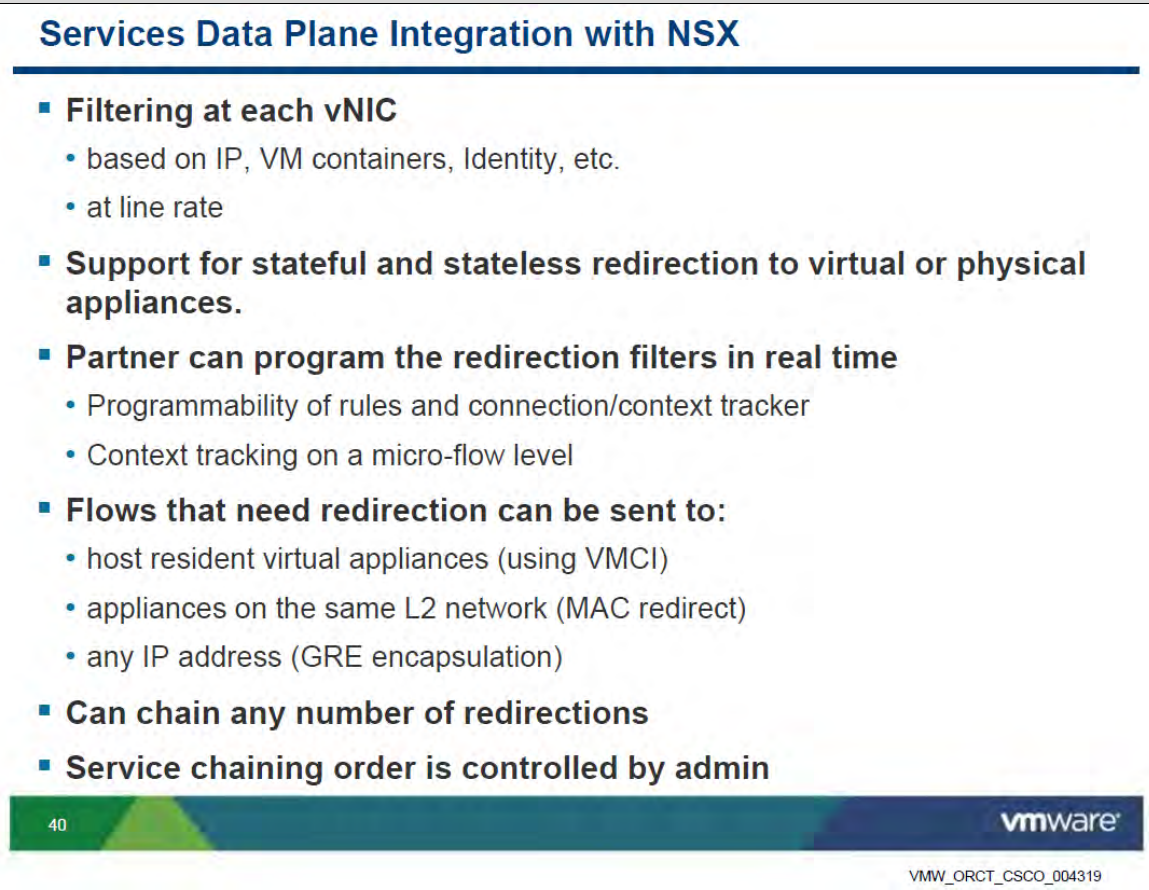
No.	'111 Patent Claim 31	VMware NSX
		NSX Components This section describes the components of the NSX solution.  VMW_ORCT_CSCO_2161

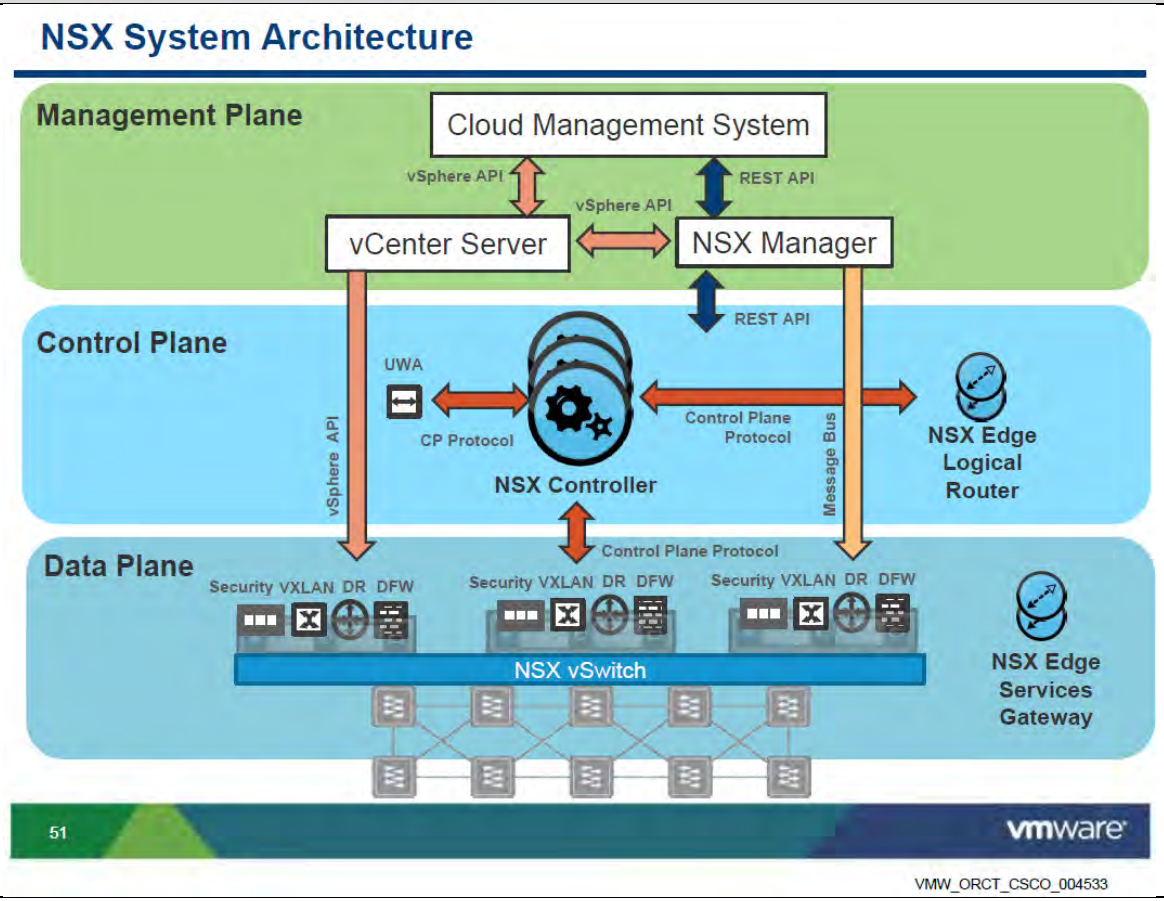
No.	'111 Patent Claim 31	VMware NSX
		Data Plane The NSX Data plane consists of the NSX vSwitch, which is based on the vSphere Distributed Switch (VDS) with additional components to enable services. Kernel modules (VIBs) run within the hypervisor kernel to provide services such as distributed routing and logical firewall and to enable VXLAN bridging capabilities. The NSX vSwitch (vDS-based) abstracts the physical network and provides access-level switching in the hypervisor. It is central to network virtualization because it enables logical networks that are independent of physical constructs, such as VLANs. Some of the benefits of the vSwitch are: ■ Support for overlay networking with protocols (such as VXLAN) and centralized network configuration. Overlay networking enables the following capabilities: ■ Creation of a flexible logical Layer 2 (L2) overlay over existing IP networks on existing physical infrastructure without the need to re-architect any of the data center networks ■ Provision of communication (east–west and north–south), while maintaining isolation between tenants ■ Application workloads and virtual machines that are agnostic of the overlay network and operate as if they were connected to a physical L2 network ■ Facilitates massive scale of hypervisors ■ Multiple features—such as Port Mirroring, NetFlow/IPFIX, Configuration Backup and Restore, Network Health Check, QoS, and LACP—provide a comprehensive toolkit for traffic management, monitoring, and troubleshooting within a virtual network Additionally, the data plane consists of gateway devices that can provide L2 bridging from the logical networking space (VXLAN) to the physical network (VLAN). The gateway device is typically an NSX Edge virtual appliance. NSX Edge offers L2, L3, perimeter firewall, load balancing, and other services such as SSL VPN and DHCP. VMW_ORCT_CSCO_ 4169

No.	'111 Patent Claim 31	VMware NSX
		

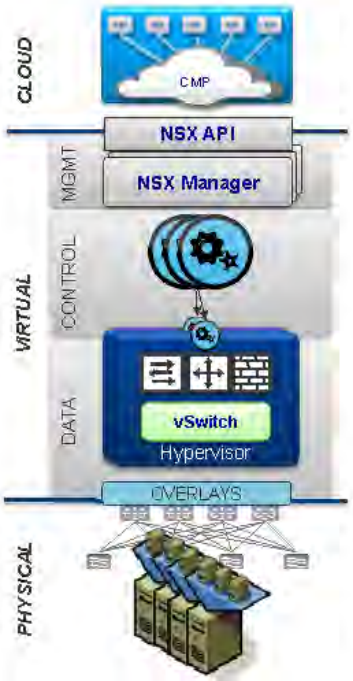
VMW_ORCT_CSCO_4313

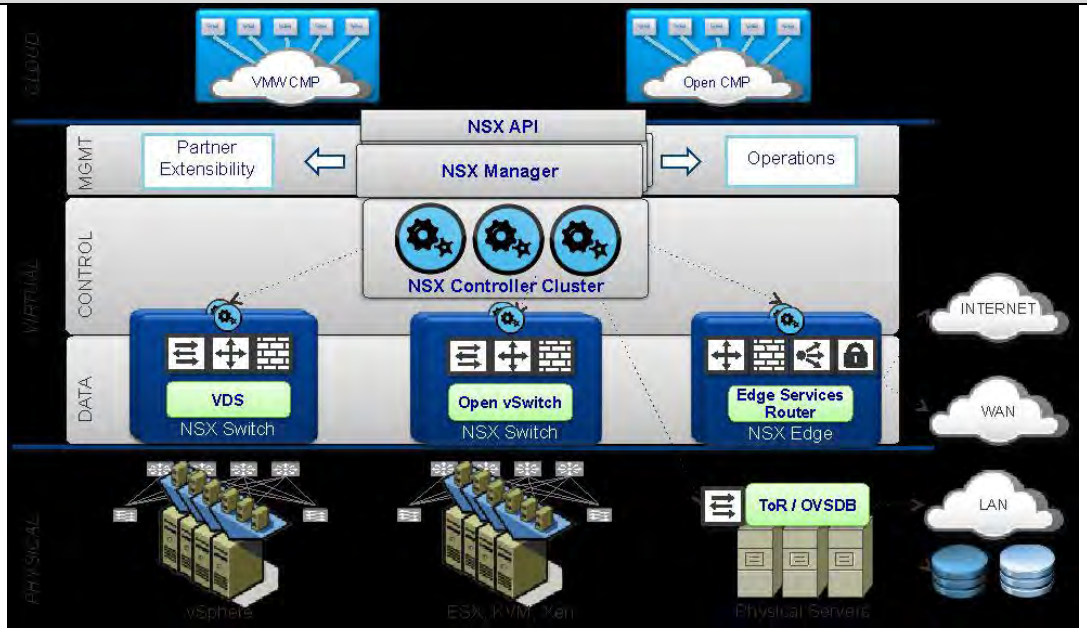
No.	'111 Patent Claim 31	VMware NSX
		 VMW_ORCT_CSCO_ 4319

No.	'111 Patent Claim 31	VMware NSX
		 VMW_ORCT_CSCO_4533

No.	'111 Patent Claim 31	VMware NSX
		

No.	'111 Patent Claim 31	VMware NSX
31[c]	the network node communication with the controller serves as a control plane.	VMware NSX System discloses the network node communication with the controller serves as a control. For example, VMware NSX System discloses the NSX controller as the control plane. Introducing VMware NSX at 3-4 **VMware NSX Architecture and Design** As depicted in the diagram below, the VMware NSX solution resides in the virtualization layer, providing L2-L7 network services to the cloud consumption layer above, and mapping these services onto the physical infrastructure below. Independence between the cloud and virtualization layers is achieved by providing a REST API exposing the network services to any upstream provisioning platform. Likewise, vSwitch overlays such as VXLAN provide independence to deploy NSX on any physical IP network infrastructure. This brings us to the network virtualization layer. For each capability, be it switching, routing or firewall, the services are provided via NSX APIs, and realized using a three-tiered design pattern encompassing the management plane, control plane and data plane. The NSX Manager internally maps the APIs onto the control plane. **The controller cluster is the work horse of the system**, handling real time mapping between the system's desired state and the running state, which it communicates to the control plane agent(s) present per hypervisor. The local information is now used to set up the appropriate switching, routing, or firewall tables and contexts. The appropriate data plane function now proceeds with high-performance, in a scale out fashion across the virtual plane.

No.	'111 Patent Claim 31	VMware NSX
		 **NSX: architectural context & design pattern**: Cloud Management Platform consumes services The REST **API** abstracts underlying services The **Manager** cluster maps services to the controllers The control plane consists of a master **controller** cluster delegating tasks to control plane agents in each hypervisor (or controlled device) The agent performs the local task of activating the **data plane** action in the hypervisor e.g. switching, routing or firewalls **Overlays** de-couple the virtual plane from the underlying physical network fabric The network fabric provides connectivity to physical servers, hosting multiple VMs connected to a programmable vSwitch. IP connectivity is the only requirement of the physical network fabric

No.	'111 Patent Claim 31	VMware NSX
		 NSX Design Guide at 11

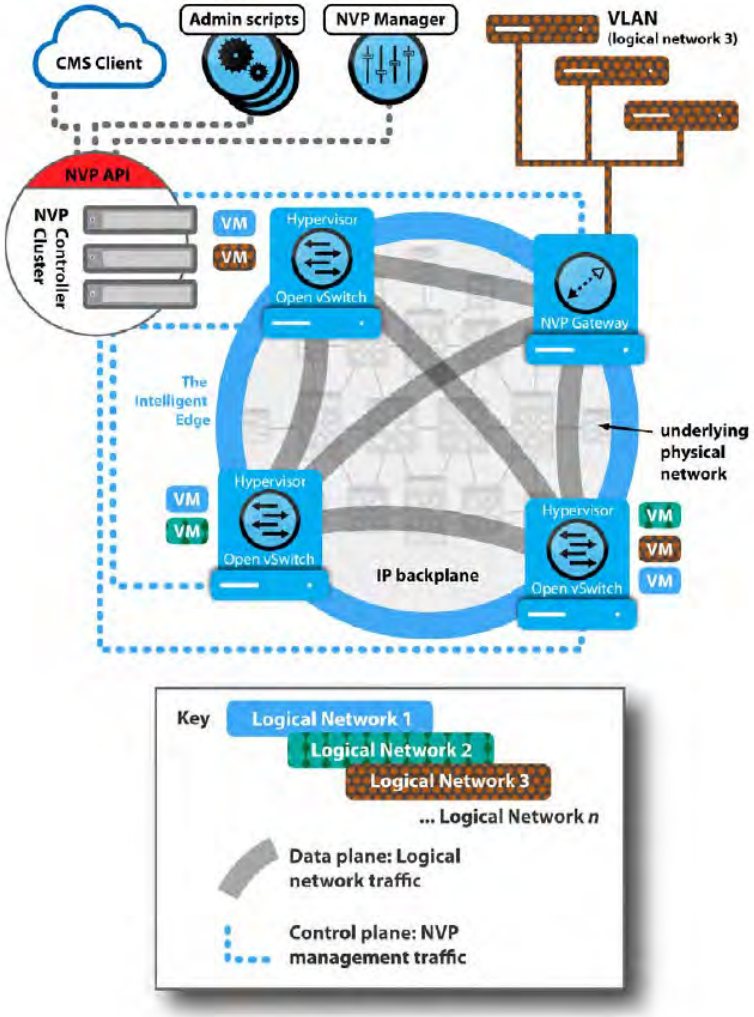
No.	'111 Patent Claim 31	VMware NSX
		2.1.2 Control Plane The NSX controller is a key part of the NSX control plane. In a vSphere environment with the vSphere Distributed Switch (VDS), the controller enables multicast free VXLAN and control plane programming of elements such as the Distributed Logical Routing (DLR). Stability and reliability of data transport are central concerns in networking. The NSX controller is a part of the control plane; it is logically separated from all data plane traffic. To further enhance high availability and scalability, NSX controller nodes are deployed in a cluster of odd number instances. In addition to controller, the control VM, provides the routing control plane that allows the local forwarding in ESXi and allows dynamic routing between ESXI and north-south routing provided by Edge VM. It is critical to understand that data plane traffic never traverses the control plane component. NSX Design Guide at 41 **4.2.2 Populating the Controller Tables** Controller tables handle information essential for L2 unicast communication. Control plane communication between ESXi hosts and the controller cluster is used to populate the VTEP, MAC, and ARP tables on controller nodes. This process is detailed in Figure 32. NSX Design Guide at 52

No.	'111 Patent Claim 31	VMware NSX
		Distributed routing is provided by a logical element called Distributed Logical Router (DLR). The DLR is essentially a router with directly connected interfaces to all hosts where VM connectivity is required. The supervisory function (i.e., control plane) to control the forwarding is imported from a control VM. The DLR consists of two primary components: • **Control Plane:** The control plane is provided by the DLR Control VM and NSX controller. The control plane supports dynamic routing protocols (e.g., BGP, OSPF), exchanges routing updates with the next layer 3 hop device (e.g., NSX Edge), and communicates with the NSX manager and the controller cluster. High availability for the DLR control VM is supported through an active/standby configuration. • **Data Plane:** DLR kernel modules (e.g., VIBs) are installed on the ESXi hosts part of the NSX domain. The kernel modules are similar to the line cards in a modular chassis supporting layer 3 routing. These kernel modules have a routing information base (RIB) that is pushed through the controller cluster. The traditional data plane functionality of route and ARP lookups is performed by the kernel modules. The kernel modules are equipped with logical interfaces (LIFs) connecting to the different logical switches. Each LIF has an IP address representing the default IP gateway for its logical L2 segment as well as a vMAC address. The IP address is unique per LIF and remains same where the logical switch exists. The vMAC associated with each LIF remains the consistent in each hypervisor as well and thus during the vMotion, the default gateway and MAC remains the same. VMW_ORCT_CSCO_ 56

NSX Architecture: Implementing Network Virtualization

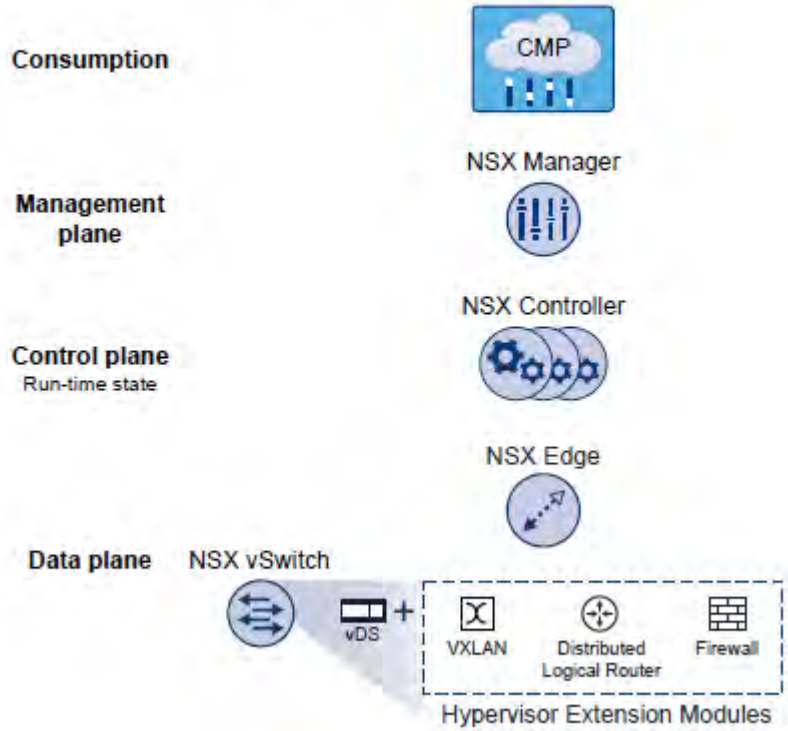
Administrators use the NVP API and Manager to configure logical networks and transport nodes that forward network traffic. NSX manages transport nodes via its control plane, and VMs send traffic on data plane.

NSX Control Plane and Data Plane

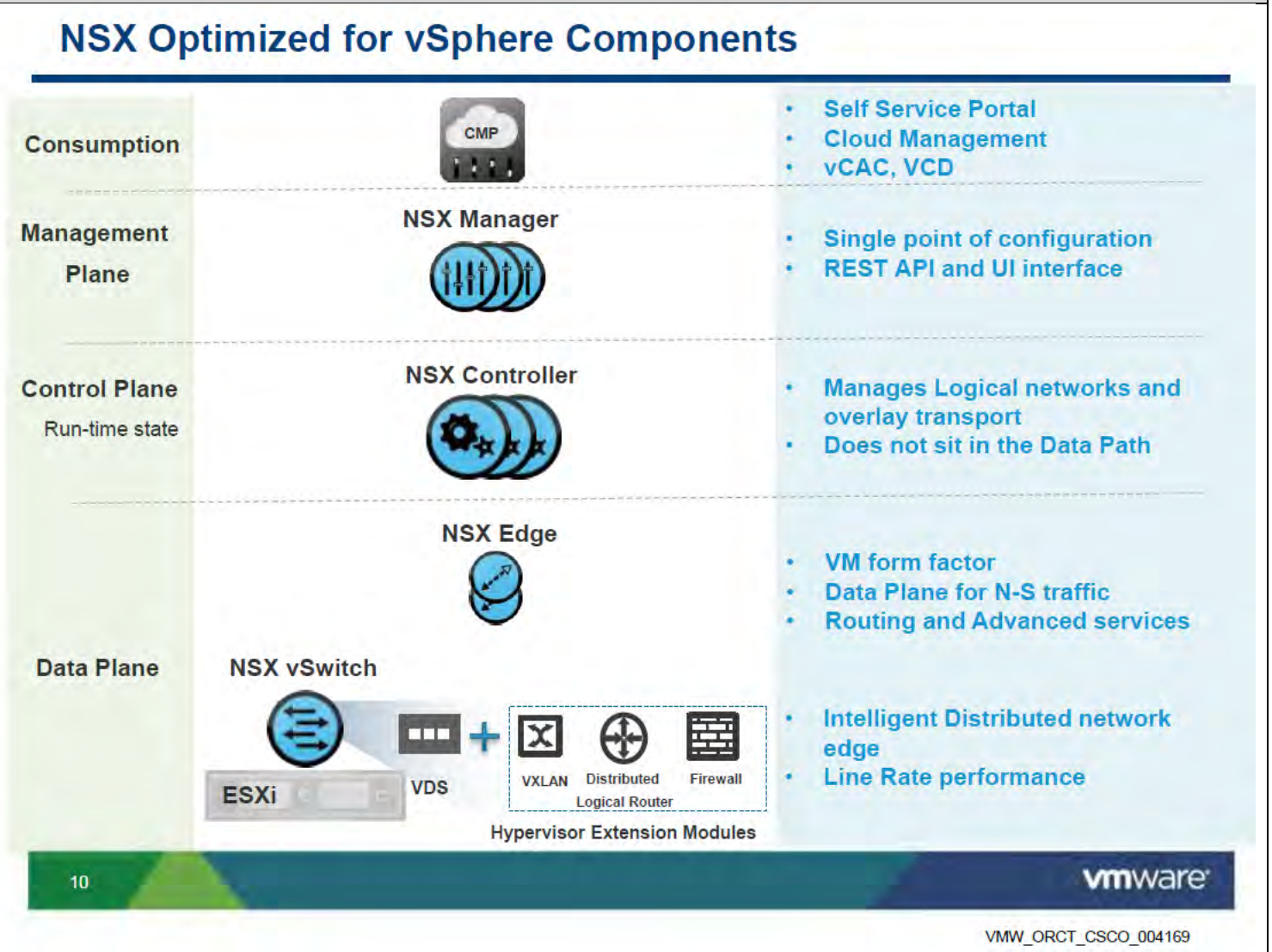


No.	'111 Patent Claim 31	VMware NSX
		VMW_ORCT_CSCO_ 57 *NSX Control Plane* The NSX Controller Cluster accepts logical network configuration instructions from administrators (through the NVP API and its clients), calculates the required network flow entries, and inserts these network flow entries into Open vSwitch (OVS) instances running on the transport nodes (hypervisor switches and NSX appliances) via OpenFlow. In each transport node, the flow entries give OVS the routing information it needs to direct packets to the right hypervisor(s) or network gateway. When OVS on a transport node encounters a packet that it does not know how to route, it asks the Controller Cluster to provide a new flow entry for that packet. In addition to logical network set-up capabilities, the NVP API allows administrators to configure and manage the forwarding state of transport nodes. The Controller Cluster is a distributed system that runs on a set of x86 servers. Clustering provides: • **High-availability**—The ability to survive failures of individual Controller Nodes. • **Scale-out**—The ability to increase the number of endpoints managed by the control plane by adding more servers to the Controller Cluster. VMW_ORCT_CSCO_ 63

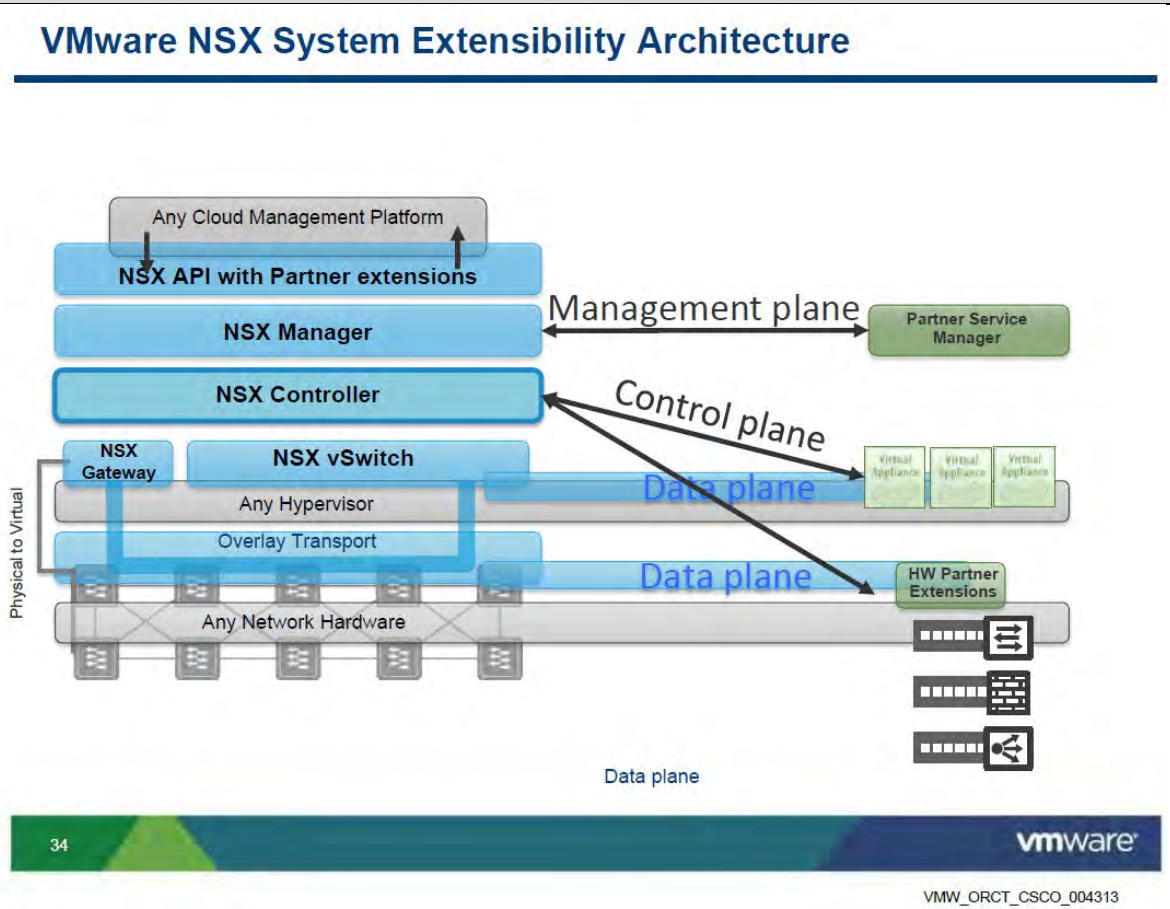
No.	'111 Patent Claim 31	VMware NSX
		## *Overview: The Controller Cluster* NSX uses a cluster of x86 nodes to implement a network control plane that controls the Open vSwitch (OVS) devices that perform packet forwarding. *Controller Cluster nodes* cooperate to manage all OVS devices and enforce consistency between the logical network view (defined via the NVP API) and the transport network view (implemented by OVS-enabled access switches). If any changes occur in the transport view, such as a VM start or migration, the Controller Cluster updates forwarding rules and state in the transport layer so that the connectivity seen by a VM is always consistent with the logical view. Similarly, if an API request changes the configuration of a logical port used by a VM, the Controller Cluster modifies the forwarding state of all relevant OVS devices in the transport view to reflect the change. Control plane processing and associated data storage is spread across all active Controller Cluster nodes, providing a *scale-out* model designed to support: • Cluster capacity growing incrementally as more nodes are added. • *Cluster h*igh-availability (HA) with fast fail-over. Note: the current release of NSX provides cluster fast fail-over but cluster capacity does not increase with additional nodes. This capability will be added in a subsequent release. VMW_ORCT_CSCO_2160

No.	'111 Patent Claim 31	VMware NSX
		NSX Components This section describes the components of the NSX solution.  VMW_ORCT_CSCO_2161

No.	'111 Patent Claim 31	VMware NSX
		Control Plane The NSX control plane runs in the NSX Controller cluster. NSX Controller is an advanced distributed state management system that provides control plane functions for NSX logical switching and routing functions. It is the central control point for all logical switches within a network and maintains information about all hosts, logical switches (VXLANs), and distributed logical routers. The controller cluster is responsible for managing the distributed switching and routing modules in the hypervisors. The controller does not have any dataplane traffic passing through it. Controller nodes are deployed in a cluster of three members to enable high-availability and scale. Any failure of the controller nodes does not impact any data-plane traffic. NSX Controllers work by distributing network information to hosts. To achieve a high level of resiliency the NSX Controller is clustered for scale out and HA. NSX Controllers must be deployed in a three-node cluster. The three virtual appliances provide, maintain, and update the state of all network functioning within the NSX domain. NSX Manager is used to deploy NSX Controller nodes. The three NSX Controller nodes form a control cluster. The controller cluster requires a quorum (also called a majority) in order to avoid a "split-brain scenario." In a split-brain scenario, data inconsistencies originate from the maintenance of two separate data sets that overlap. The inconsistencies can be caused by failure conditions and data synchronization issues. Having three controller nodes ensures data redundancy in case of failure of one NSX Controller node. VMW_ORCT_CSCO_4169

No.	'111 Patent Claim 31	VMware NSX
		

VMW_ORCT_CSCO_4313

No.	'111 Patent Claim 31	VMware NSX
		 VMW_ORCT_CSCO_ 4533

No.	'111 Patent Claim 31	VMware NSX
		