

UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE PATENT TRIAL AND APPEAL BOARD

CISCO SYSTEMS, INC.,

Petitioner,

-VS-

ORCKIT CORPORATION,

Patent Owner

Case No. IPR2023-00554
U.S. Patent No. 10,652,111

PETITIONER'S DEMONSTRATIVES
PTAB ORAL ARGUMENT
July 9, 2024

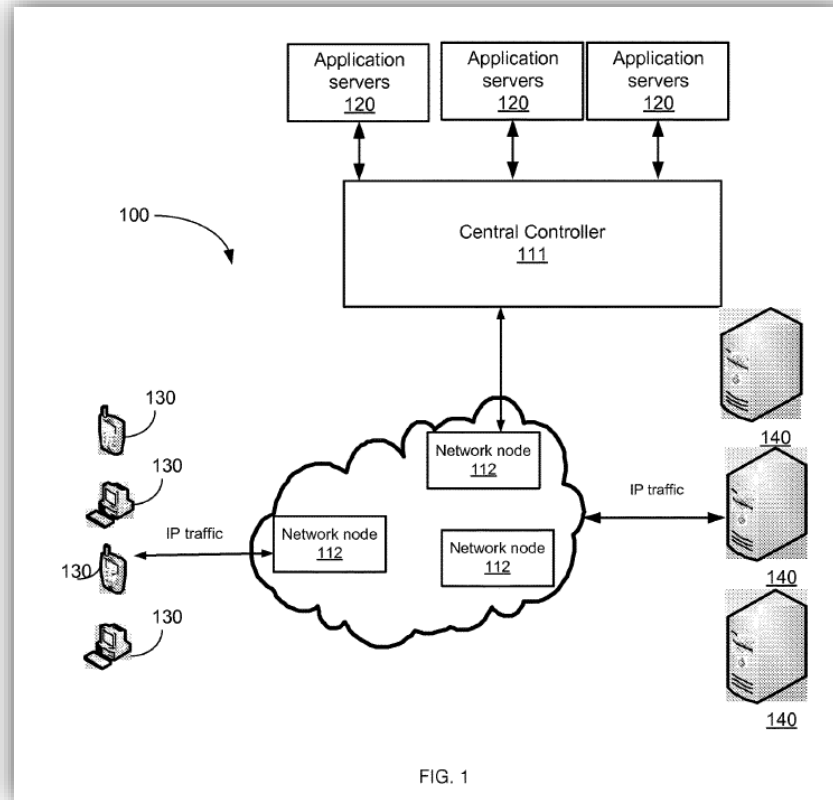
The '111 Patent Relates to Deep Packet Inspection (DPI) in Software Defined Networks (SDN)

TECHNICAL FIELD

This disclosure generally relates to techniques for deep packet inspection (DPI), and particularly for DPI of traffic in cloud-based networks utilizing software defined networks. ¹⁵

EX1001, Col. 1:14-17

Figure 1 of the '111 Patent



EX1001, Figure 1

Figure 2 of the '111 Patent

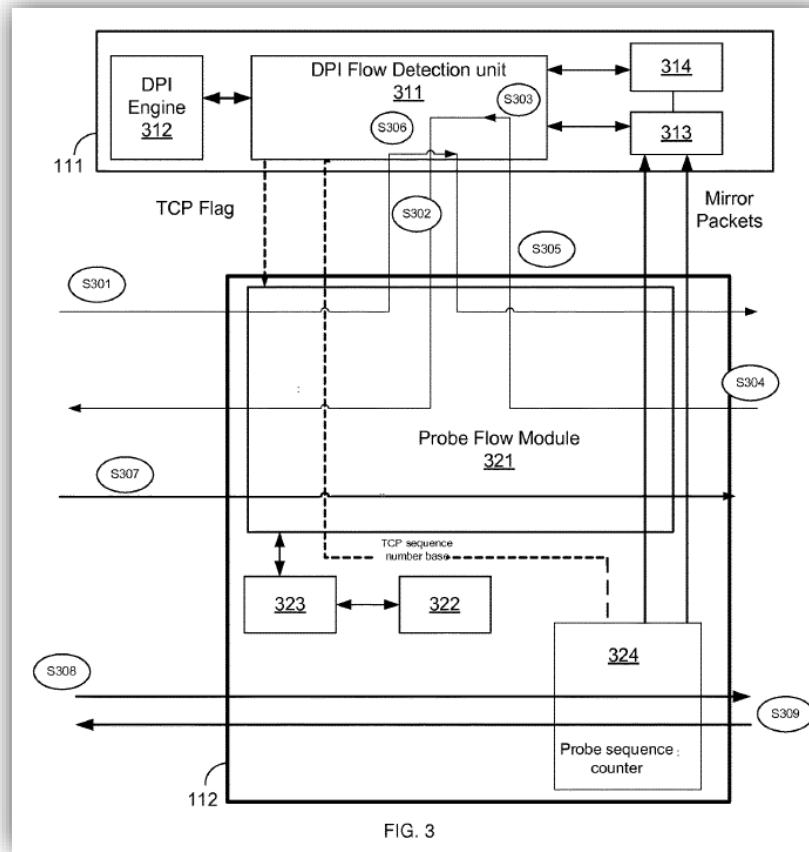
200

KEY <u>210</u>					DATA <u>220</u>									
Client IP address	Server IP address	Client source TCP port	Server destination TCP port	IP protocol number	Flow ID	Client→ Server sequence number M	Server→ Client sequence number N	state	Creation timestamp	Client→ Server Hit counter X [bytes]	Server → Client Hit counter Y [bytes]	Client→ Server data buffer	Server→ Client buffer	Age bit
192.1.1.1	209.1.4.4	15431	21	6	1	0xf46d5c34	0x3c98b9ab	ACK	15:32:13					

FIG. 2

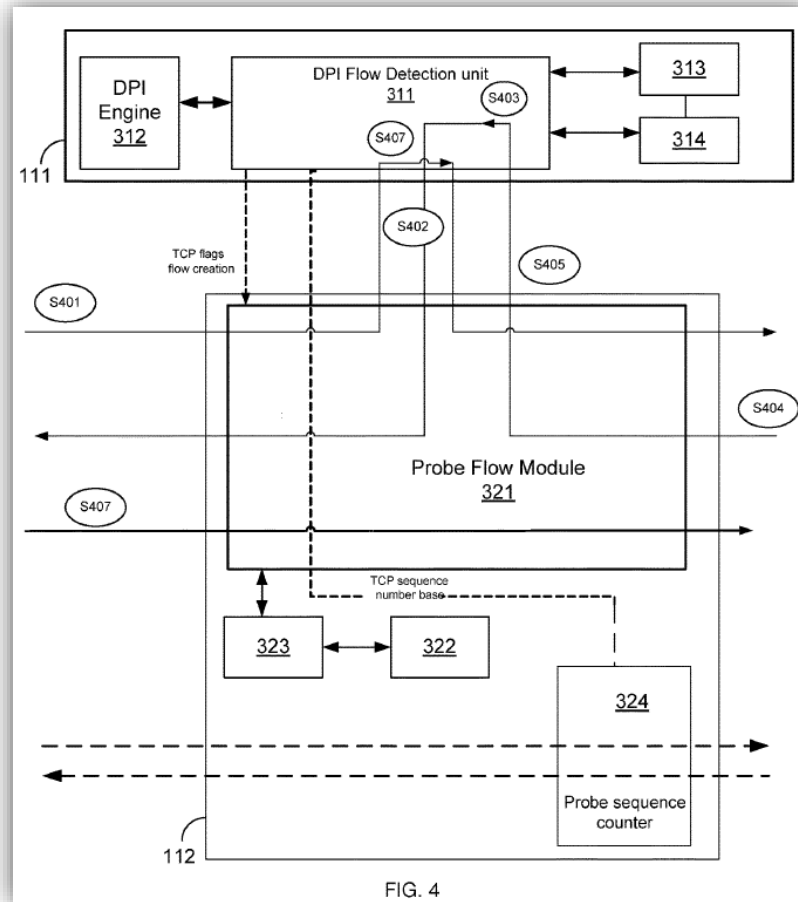
EX1001, Figure 2

Figure 3 of the '111 Patent



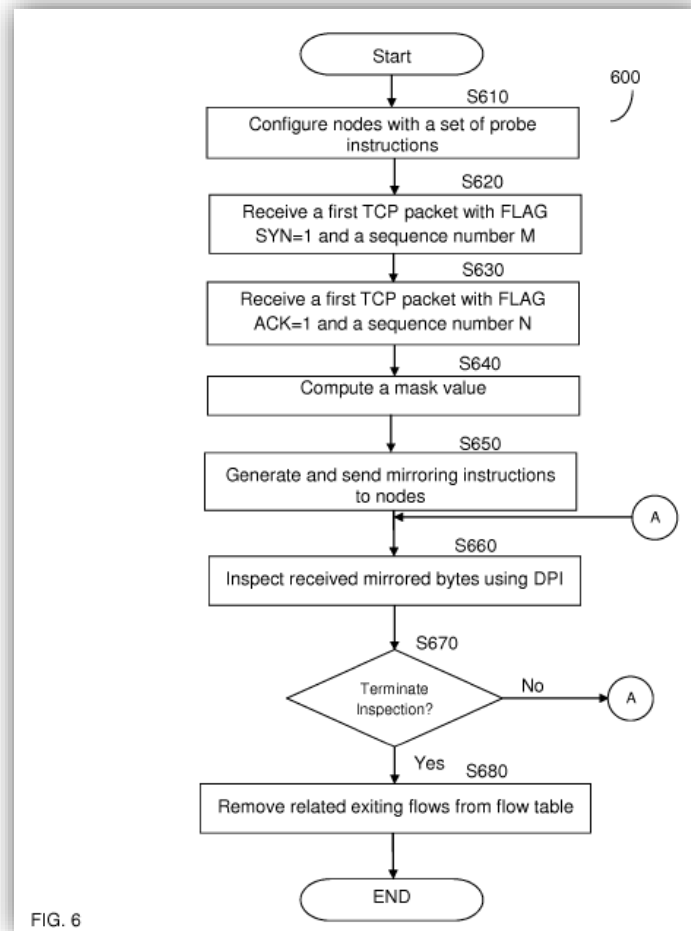
EX1001, Figure 3

Figure 4 of the '111 Patent



EX1001, Figure 4

Figure 6 of the '111 Patent



EX1001, Figure 6

Deep Packet Inspection (DPI) Was Known

BACKGROUND

Deep packet inspection (DPI) technology is a form of network packet scanning technique that allows specific data patterns to be extracted from a data communication channel. Extracted data patterns can then be used by various applications, such as security and data analytics applications. DPI

EX1001, Col. 1:20-25

Software Defined Networking (SDN) Was Known

A software defined networking is a relatively new type of 30
networking architecture that provides centralized manage-
ment of network nodes rather than a distributed architecture
utilized by conventional networks. The SDN is prompted by

EX1001, Col. 1:30-33

Background on SDN and Open Flow Protocol

Specifically, in SDN-based architectures the data forwarding (e.g. data plane) is typically decoupled from control decisions (e.g. control plane), such as routing, resources, and other management functionalities. The decoupling may also allow the data plane and the control plane to operate on different hardware, in different runtime environments, and/or operate using different models. As such, in an SDN network, the network intelligence is logically centralized in the central controller which configures, using Open Flow protocol, network nodes and to control application data traffic flows.

Although, the OpenFlow protocol allows addition of programmability to network nodes for the purpose of packets-processing operations under the control of the central controller, the OpenFlow does not support any mechanism to allow DPI of packets through the various networking layers as defined by the OSI model. Specifically, the current OpenFlow specification defines a mechanism to parse and extract only packet headers, in layer-2 through layer-4, from packets flowing via the network nodes. The OpenFlow specification does not define or suggest any mechanism to extract non-generic, uncommon, and/or arbitrary data patterns contained in layer-4 to layer 7 fields. In addition, the OpenFlow specification does not define or suggest any mechanism to inspect or to extract content from packets belonging to a specific flow or session. This is a major limitation as it would not require inspection of the packet for the purpose of identification of, for example, security threats detection.

EX1001, Col. 1:39-67

Embodiments of the '111 Patent

Certain embodiments disclosed herein include a method for deep packet inspection (DPI) in a software defined network (SDN), wherein the method is performed by a
30 central controller of the SDN. The method comprises: configuring a plurality of network nodes operable in the SDN with at least one probe instruction; receiving from a network node a first packet of a flow, wherein the first packet matches the at least one probe instruction, wherein the first packet
35 includes a first sequence number; receiving from a network node a second packet of the flow, wherein the second packet matches the at least one probe instruction, wherein the second packet includes a second sequence number, wherein the second packet is a response of the first packet; computing
40 a mask value respective of at least the first and second sequence numbers, wherein the mask value indicates which bytes to be mirrored from subsequent packets belonging to the same flow, wherein the mirrored bytes are inspected;
45 generating at least one mirror instruction based on at least the mask value; and configuring the plurality of network nodes with at least one mirror instruction.

EX1001, Col. 2:27-47

The Central Controller Performs Deep Packet Inspection

5 According to some embodiments, the central controller **111** is configured to perform deep packet inspection on designated packets from designated flows or TCP sessions. To this end, the central controller **111** is further configured to instruct each of the network nodes **112** which of the
10 packets and/or sessions should be directed to the controller **111** for packet inspections.

According to some embodiments, each network node **112** is configured to determine if an incoming packet requires inspection or not. The determination is performed based on
15 a set of instructions provided by the controller **111**. A packet that requires inspection is either redirected to the controller **111** or mirrored and a copy thereof is sent to the controller **111**. It should be noted that traffic flows that are inspected are not affected by the operation of the network node **112**. In an

EX1001, Col. 4:5-19

The Operation of the Probe Flow Module

In exemplary implementation, the network node **112** 50 includes a probe flow module **321**, a memory **322**, and a processing unit **323**. The probe flow module **321** is configured to redirect any new TCP connection state initiation packets to the DPI flow detection module **311**, as well as to extract several packets from each detected TCP flow and 55 mirror them to the flow detection module **311**. In an embodiment, probe flow module **321** executes functions and/or implements logic to intercept TCP flags, redirect packets, and count sequence numbers.

EX1001, Col. 5:50-59

Claim 1 of the '111 Patent

1. 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
55 to the network node, the method comprising:
 sending, by the controller to the network node over the packet network, an instruction and a packet-applicable criterion;
 receiving, by the network node from the controller, the
60 instruction and the criterion; receiving, by the network node from the first entity over the packet network, a packet addressed to the second entity;
 checking, by the network node, if the packet satisfies the criterion;
65 responsive to the packet not satisfying the criterion, sending, by the network node over the packet network, the packet to the second entity; and

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.

EX1001, Cols. 10:51-11:4

Lin Discloses SDN

The present invention relates generally to computer security, and more particularly but not exclusively to software defined networking.

EX1005, Col. 1:8-9

Lin Discloses The Flow of Packets Through The SDN Switch

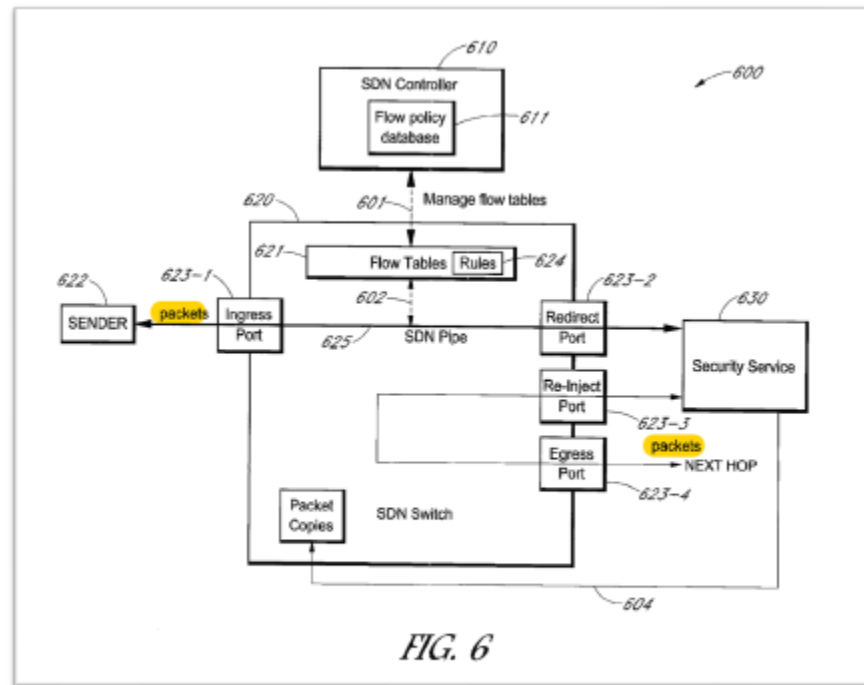
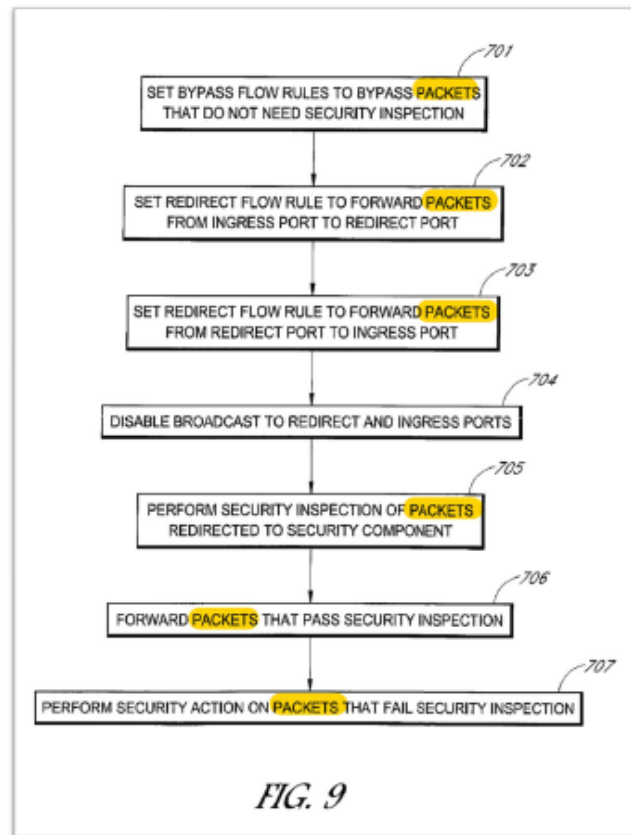


Figure 6 of Lin (Annotated)

EX1004, ¶76

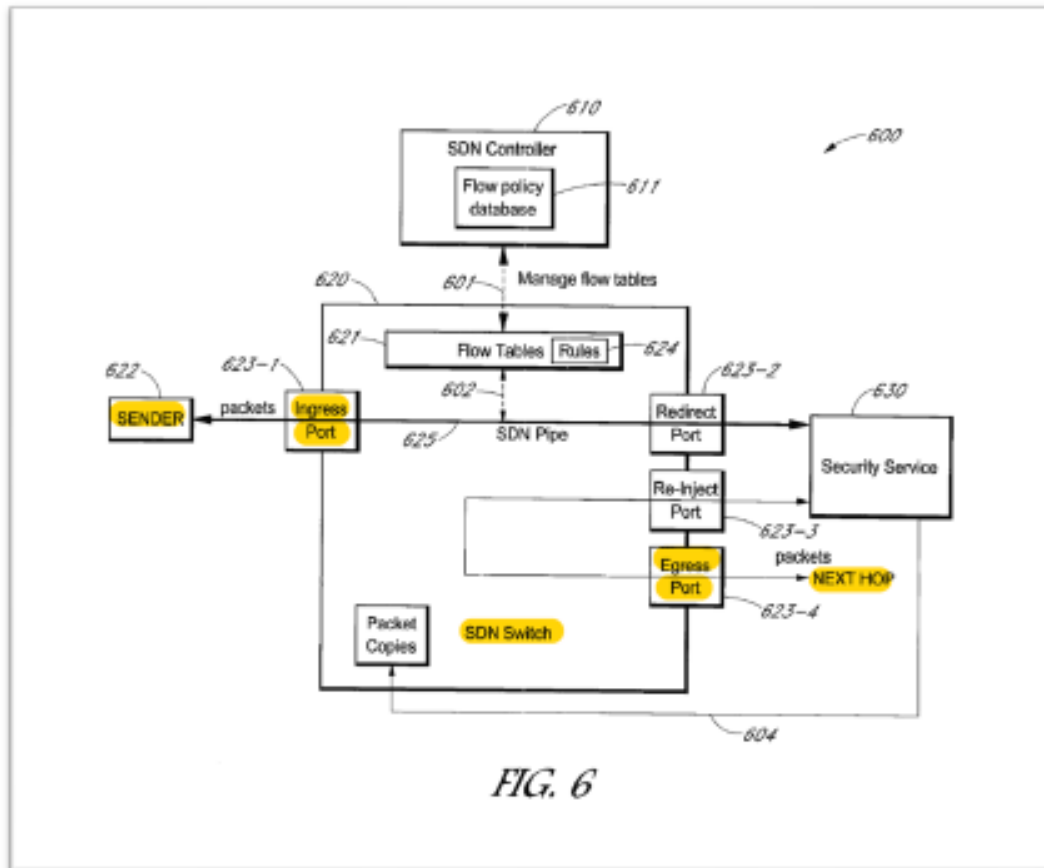
Lin Discloses The Flow of Packets Through The SDN Switch



EX1004, ¶76

Figure 9 of Lin (Annotated)

The Packets Travel From The Sender Through The SDN Switch To The Next Hop



EX1004, ¶102

100

Figure 6 of Lin (Annotated)

Lin Discloses The Claimed Controller

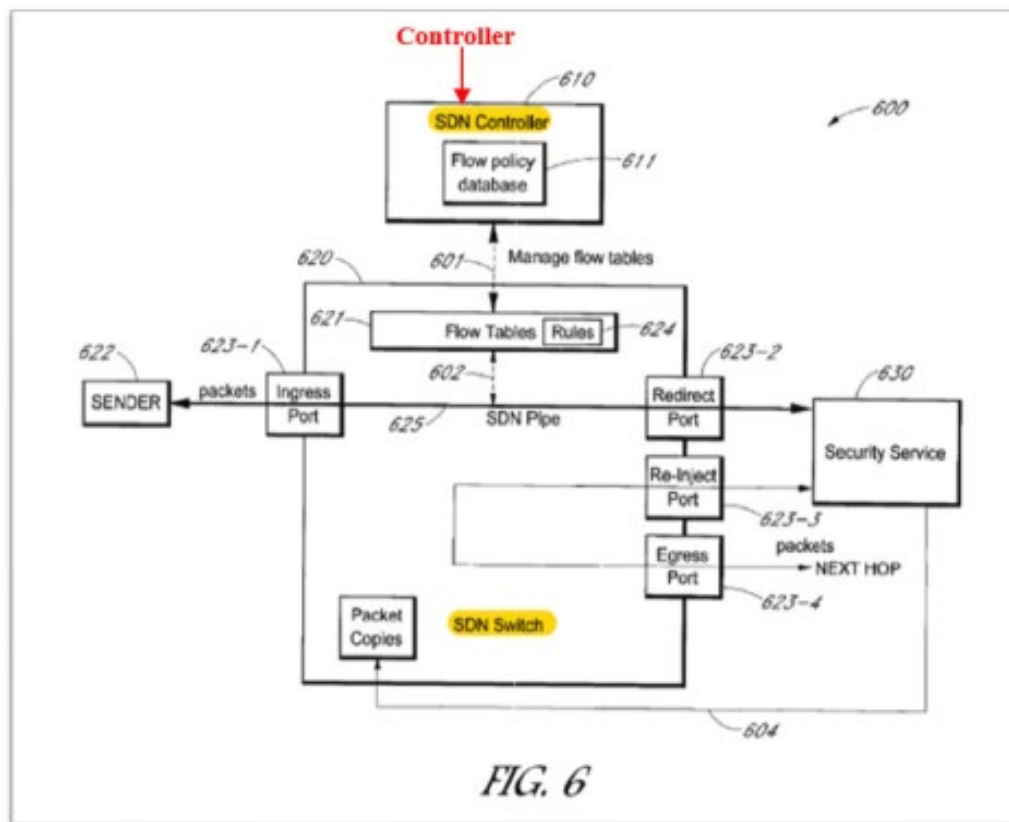


Figure 6 of Lin (Annotated)

EX1004, ¶78

Lin Teaches That SDN “Separates The Control Plane From The Data Plane”

2. Description of the Background Art

10

Software defined networking (SDN) is an emerging architecture for computer networking. Unlike traditional computer network architectures, SDN separates the control plane from the data plane. This provides many advantages, including relatively fast experimentation and optimization of switching and routing policies. SDN is applicable to both physical (i.e., real) and virtual computer networks.

15

EX1005, Col. 1:10-17

Lin Discloses The Use Of The OpenFlow Protocol

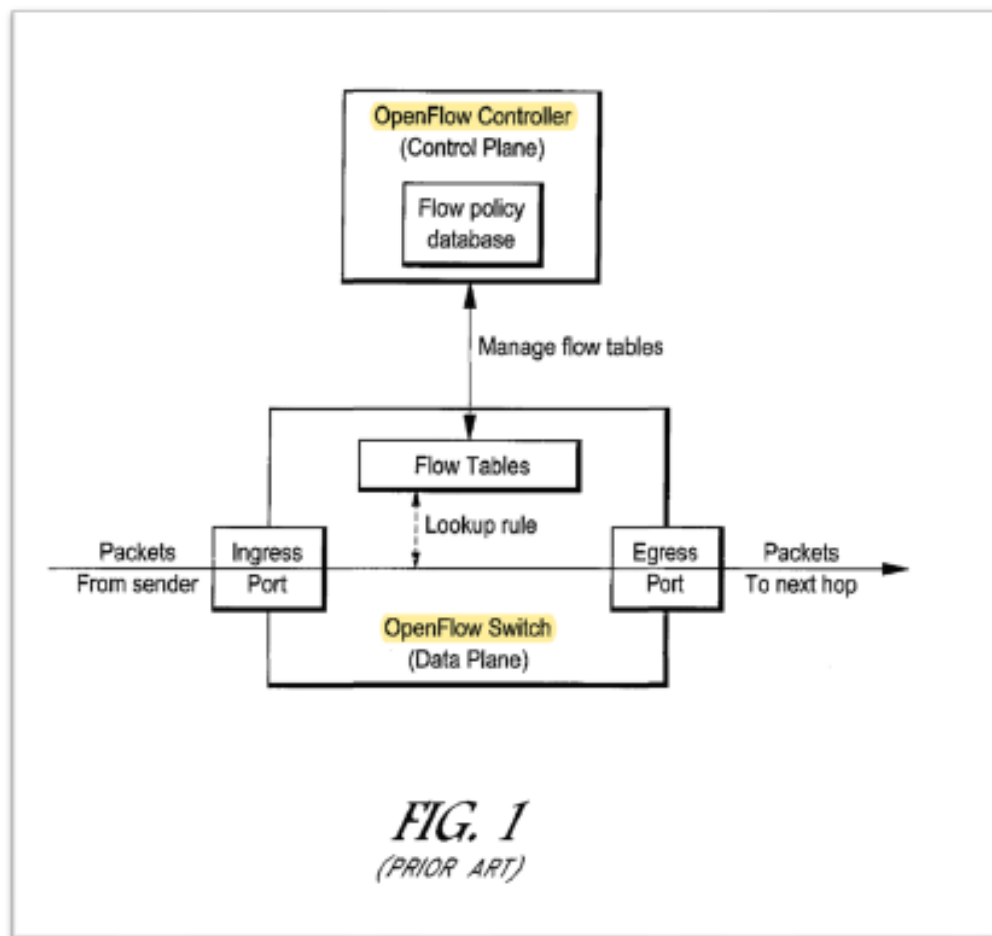
The OpenFlow™ protocol is an open protocol for remotely controlling forwarding tables of network switches that are enabled for SDN. Generally speaking, the OpenFlow protocol allows direct access to and manipulation of the forwarding plane of network devices, such as switches and routers. A control plane of an OpenFlow™ protocol-compliant computer network (also referred to as an “OpenFlow™ controller”) may communicate with OpenFlow™ switches (i.e., network switches that are compliant with the OpenFlow™ protocol) to set flow policies that specify how the switches should manipulate packets of network traffic. Example

20

25

EX1005, Col. 1:18-22

Lin's Controller And Switch Can Utilize The OpenFlow Protocol



X1004, ¶186

Figure 1 of Lin (Annotated)

Lin's Controller Determines Flow Policies That Contain "A Condition And Corresponding Action To Be Performed"

database. The controller determines flow policies in conjunction with network forwarding setting and network topology. 40
The flow policies may contain a condition and corresponding action to be performed when the condition is met. The action may specify how to manipulate a packet.

EX1005, Col. 1:39-43

Lin Utilizes An SDN Controller To Insert Flow Rules In A Flow Table Of The SDN Switch

In one embodiment, a software defined networking (SDN) computer network includes an SDN controller and an SDN switch. The SDN controller inserts flow rules in a flow table of the SDN switch to create an SDN pipe between a sender component and a security component. A broadcast function of the SDN switch to the ports that form the SDN pipe may be disabled. The SDN pipe allows outgoing packets sent by the sender component to be received by the security component. The security component inspects the outgoing packets for compliance with security policies and allows the outgoing

2

packets to be forwarded to their destination when the outgoing packets pass inspection. The SDN controller may also insert a flow rule in the flow table of the SDN switch to bypass inspection of specified packets.

EX1005, Col. 1:57-2:4

Lin Discloses The Use Of DPI

Network security vendors provide network security services, such as firewall or **deep packet inspection (DPI)**. Generally speaking, to provide network security services, packets of network traffic are intercepted for inspection. One way of

EX1005, Col. 3:11-14

Lin Discloses Packets Sent To A Security Service

Another way of intercepting network traffic is to mirror the 25 packets to be inspected on a switch that provides vendor specific mirroring application programming interface (API) as shown in FIG. 4. A user may make an API call such that particular packets that enter the ingress port of the switch are 30 redirected or mirrored to the security service by way of a connection tunnel or a mirror port. The security service may forward the redirected or mirrored packets back to an egress port of the switch after inspection.

EX1005, Col. 3:25-33

Lin's Controller And Switch Can Use The OpenFlow Protocol

Flow™ protocol. Accordingly, in one embodiment, the SDN controller 610 comprises an OpenFlow™ controller and the SDN switch 620 comprises an OpenFlow™ switch. The SDN controller 610 and the SDN switch 620 comprise the control plane and data plane, respectively, of the SDN computer network 600. The SDN computer network 600 may have a plurality of SDN switches 620 but only one is shown for clarity of illustration. The SDN controller 610 and the SDN switch 620 are logically separate components.

EX1005, Col. 3:44-52

Lin's SDN Controller Controls Network Behavior

The SDN controller **610** provides a logically centralized framework for controlling the behavior of the SDN computer
10 network **600**. This is in marked contrast to traditional com-

EX1005, Col. 4:8-10

Lin's Flow Policies Are Enforced Through Flow Rules

work 600. The flow policies may be enforced in terms of flow rules (labeled as 624) that are stored in the flow tables 621 of the SDN switch 620. As a particular example, a flow policy in the flow policy database 611 may indicate inspection of particular packets (e.g., those that meet one or more conditions) by a security service 630. That flow policy may be imple-

EX1005, Col. 4:21-27

Lin's Flow Rules Are Located In Flow Tables

The SDN switch **620** may comprise one or more flow tables **621**. The flow tables **621** may comprise one or more flow rules (labeled as **624**) that indicate how to manipulate or process packets that are passing through the SDN switch **620**. As a particular example, a flow rule may indicate that a packet received in the ingress port **623-1** is to be forwarded to the redirect port **623-2**. Another flow rule may indicate that a packet received in the redirect port **623-2** is to be forwarded to the ingress port **623-1**. The just mentioned pair of flow rules are redirect flow rules that create an SDN pipe between the sender component **622** and the security service **630**, allowing the security service **630** to inspect packets sent by or going to the sender component **622**. Table 1 shows an example flow

EX1005, Col. 4:53-65

The Structure Of Lin's Flow Tables

A flow table may include columns that indicate one or more conditions, a column that indicates an action to take when the conditions are met, and a column for statistics. A row on the 10

EX1005, Col. 5:8-10

Lin Utilizes The IP Source And Destination Addresses

indicate conditions. For example, “IN_PORT”, “MAC src”¹⁵
(media access control (MAC) address of the source of the
packet), “MAC dst” (MAC address of the destination of the
packet), “IP src” (Internet Protocol (IP) address of the source
of the packet), “IP dst” (IP address of the destination of the²⁰
packet), etc. are conditions that identify a particular packet.
When the conditions are met, i.e., the particular packet is
identified, the action indicated in the corresponding “Action”
column is performed on the packet. The asterisks in Table 1
indicate an irrelevant condition.²⁵

EX1005, Col. 5:15-25

Lin Forms An SDN Pipe Between The Sender Component And The Security Service

In the example of Table 1, the first and second rows are redirect flow rules for forming an SDN pipe between the sender component 622 and the security service 630. More specifically, the first row of Table 1 is a flow rule instructing the SDN switch 620 to forward packets received in a port having the Ingress_port_ID (e.g., ingress port 623-1) to the redirect port (e.g., redirect port 623-2). Similarly, the second row of Table 1 is a flow rule instructing the SDN switch 620 to forward packets received in a port having a “Redirect_port_ID” to the ingress port.

EX1005, Col. 5:26-36

Lin's Security Service Inspects Packets

623-3 of the SDN switch 620. The security service 630 may 45
inspect packets for compliance/non-compliance with secu-
rity policies, such as for presence of malicious code, compli-
ance with firewall rules and access control lists, network
intrusion detection, and other computer network security ser-
vices. The security service 630 may employ conventional 50
packet inspection algorithms. The security service 630 may
comprise the Trend Micro Deep Security™ service, for
example. The security service 630 may also comprise a physi-
cal machine, e.g., a server computer, an appliance, a gateway
computer, etc. 55

EX1005, Col. 5:45-55

Lin's Security Service Is Connected To The SDN Switch

The security service 630 may be connected to the SDN switch 620 by a physical link (i.e., using a wire), a virtual link (i.e., in a virtualized environment), or by a software tunnel. As a particular example, instead of using a physical link or a virtual link, the security service 630 may be connected to the SDN switch 620 by a software tunnel using generic routing encapsulation (GRE), stateless transport tunneling (STT), or some other software tunneling protocol supported by the SDN switch 620. In that example, the security service 630 serves as a remote (i.e., not in the same physical or virtual network) service that is only logically connected the SDN switch 620 by way of the software tunnel.

EX1005, Col. 5:56-67

Lin's Flow Rules Allow Redirection Of Packets To The Security Service

5 The SDN controller 610 may insert flow rules in the flow tables 621 (see arrow 601) to create an SDN pipe (labeled as 625) between the sender component 622 and the security service 630. The SDN pipe allows outgoing packets sent by the sender component 622 or incoming packets going to the sender component 622 to be redirected to the security service 630 for inspection before the packets are sent out of the SDN switch 620. In one embodiment, the SDN pipe is created by creating a first flow rule that forwards packets received in the
10 ingress port 623-1 to the redirect port 623-2, and a second flow rule that forwards packets received in the redirect port 623-2 to the ingress port 623-1.

EX1005, Col. 6:1-12

Lin's Security Service May Drop Packets Or Perform Other Security Services

50 with security policies. The security service 630 may drop, or
perform other security response, to packets that do not pass
inspection (e.g., packets that do not meet firewall policies,
packets containing prohibited payload, packets with mali-
cious content, etc.). The security service 630 may forward
55 those packets that pass inspection toward their destination by
re-injecting the packets back into the SDN switch 620 by way
of the re-inject port 623-3. Once back in the SDN switch 620

EX1005, Col. 6:49-57

Lin May Mirror Packets To The Security Service

Re-injecting packets that pass inspection consume bandwidth, as the packets will have to be transmitted by the security service 630 to the re-inject port 623-3. For optimization, the SDN switch 620 may be configured to copy packets that are redirected to the security service 630 for inspection. This way, the security service 630 simply has to inform the SDN switch 620 an action to take on packets based on the result of the inspection (see arrow 604). For example, the security service 630 may send an index identifying the packets and an action on how to manipulate the packets. The action may

EX1005, Col. 7:9-19

Lin Allows Packets To Pass Through The SDN Switch If They Do Not Need Inspection

In one embodiment, bypass flow rules are inserted in the flow tables 621 such that particular packets that do not need to be inspected are not redirected to the security service 630. This embodiment is explained with reference to example flow tables of Tables 2 and 3.

25

EX1005, Col. 7:23-27

Table 2 of Lin

TABLE 2								30
IN_PORT	...	IP src	TCP src port	TCP dst port	...	Action	Count	
Ingress_port_ID	*	*	*	80	*	Egress port	120	35
Egress_port_ID	*	*	80	*	*	Ingress port	120	
Ingress_port_ID	*	*	*	*	*	Redirect port	10	
Redirect_port_ID	*	*	*	*	*	Ingress port	10	

EX1005, Col. 7:30-37

Lin's Bypass Rules

In the example of Table 2, the first two rows are bypass rules for bypassing packets coming from or going to a transport control protocol (TCP) port 80. More specifically, hypertext transfer protocol (HTTP) packets, i.e., port 80 packets, that are received in the ingress port with the Ingress_port_ID (i.e., ingress port 623-1) are forwarded directly to the egress port (i.e., egress port 623-4), instead of being redirected to the redirect port 623-2 for inspection by the security service 630. Similarly, HTTP packets received in the egress port with the Egress_port_ID (i.e., egress port 623-4) are forwarded directly to the ingress port 623-1 without being redirected to the security service 630.

EX1005, Col. 7:39-50

Lin Chooses Which Packets To Redirect To The Security Service

In the example of Table 2, the bottom two rows are redirect flow rules for forming the SDN pipe between the sender component 622 and the security service 630. Because the bypass flow rules are inserted in the flow tables 621 with higher priority than the redirect flow rules, the bypass flow rules are followed by the SDN switch 620 before the redirect flow rules. Accordingly, HTTP packets are not redirected for inspection by the security service 630. Other packets, i.e., non-HTTP packets, are redirected to the security service 630 per the redirect flow rules. Bypass flow rules and redirect flow rules may be set at different priority levels to meet particular packet inspection needs.

EX1005, Col. 7:51-63

Lin Inspects Certain Packets While Allowing Other Packets To Bypass Inspection

The bypass and redirect flow rules also allow for inspection of particular packets, while allowing all other packets to bypass inspection. This is illustrated in the example flow table of Table 3. 65

EX1005, Col. 7:64-67

Table 3 of Lin

TABLE 3

IN_PORT	...	IP src	TCP src port	TCP dst port	...	Action	Count
5	Ingress_port_ID	*	*	*	80	* Redirect port	10
	Redirect_port_ID	*	*	80	*	* Ingress port	10
	Ingress_port_ID	*	*	*	*	* Egress port	130
	Egress_port_ID	*	*	*	*	* Ingress port	130

EX1005, Col. 8:1-8

EX1004, ¶92

The Operation Of Table 3

10 In the example of Table 3, the top two rows are redirect flow
rules for redirecting HTTP packets to the security service 630
for inspection, while the bottom two rows are bypass flow
rules for all packets. Because the redirect flow rules are at
higher priority than the bypass flow rules, HTTP packets are
15 sent through the SDN pipe formed in the SDN switch 620
between the sender component 622 and the security service
630. All other packets bypass the SDN pipe, and are accord-
ingly not inspected by the security service 630.

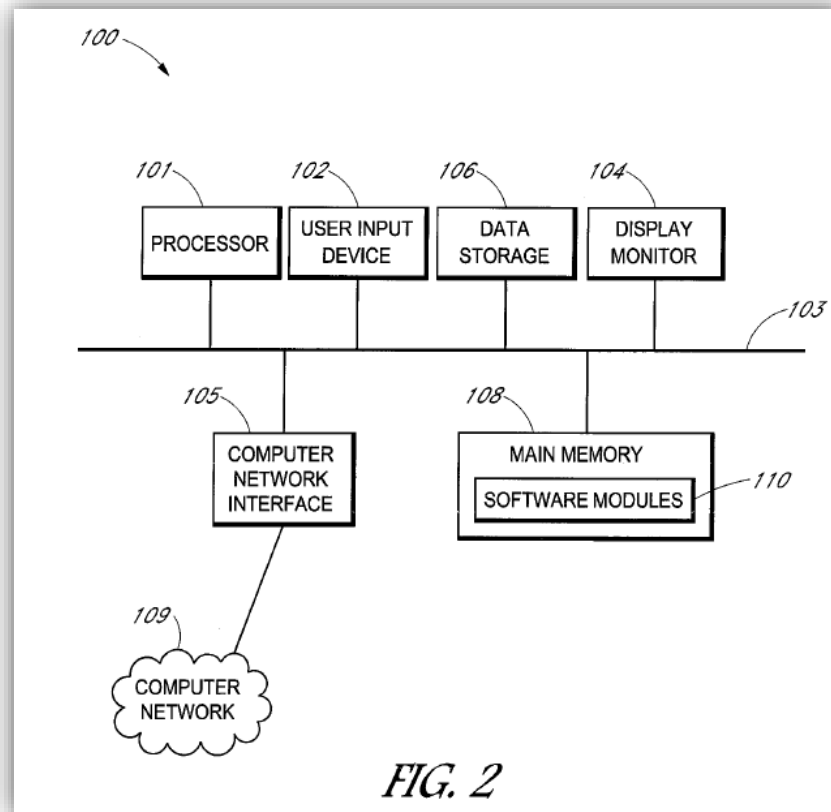
EX1005, Col. 8:10-18

Lin's Bypass Flow Rules

In the method of FIG. 9, the SDN controller inserts one or more bypass flow rules in the flow table of an SDN switch that is controlled by the SDN controller (step 701). The bypass flow rules may instruct the SDN switch to forward particular packets directly from one port to another port of the SDN switch without being redirected to a security component, such as a security service, for inspection. The bypass flow rules may be inserted at higher or lower priority than redirect flow rules for creating an SDN pipe that redirects packets to be inspected to the security service. The security service may comprise a physical or virtual machine that provides a security service by inspecting network traffic.

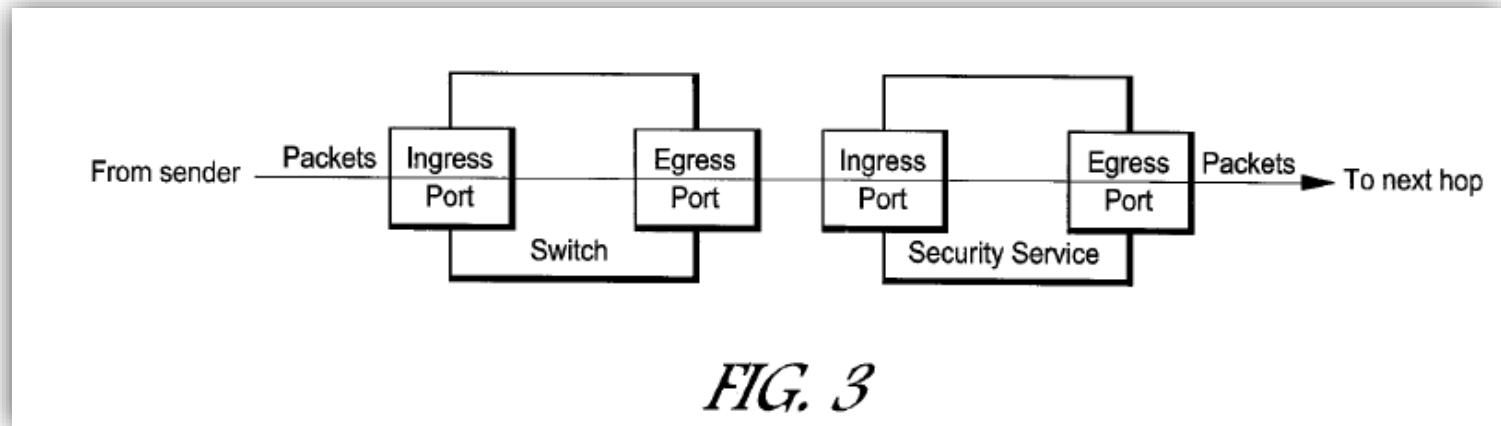
EX1005, Col. 9:10-21

Figure 2 of Lin



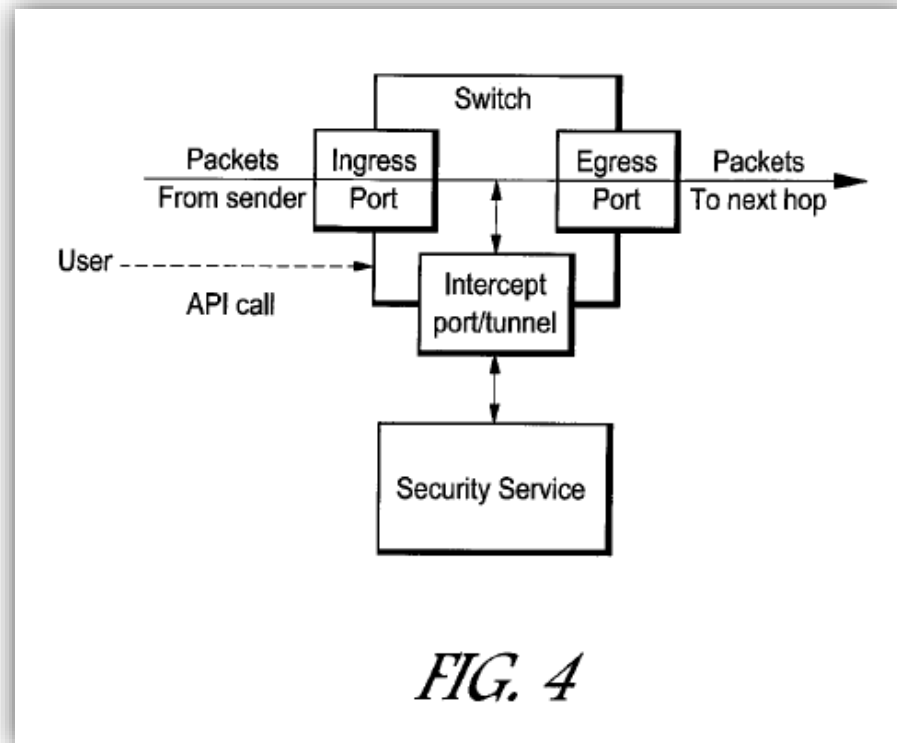
EX1005, Figure 2

Figure 3 of Lin



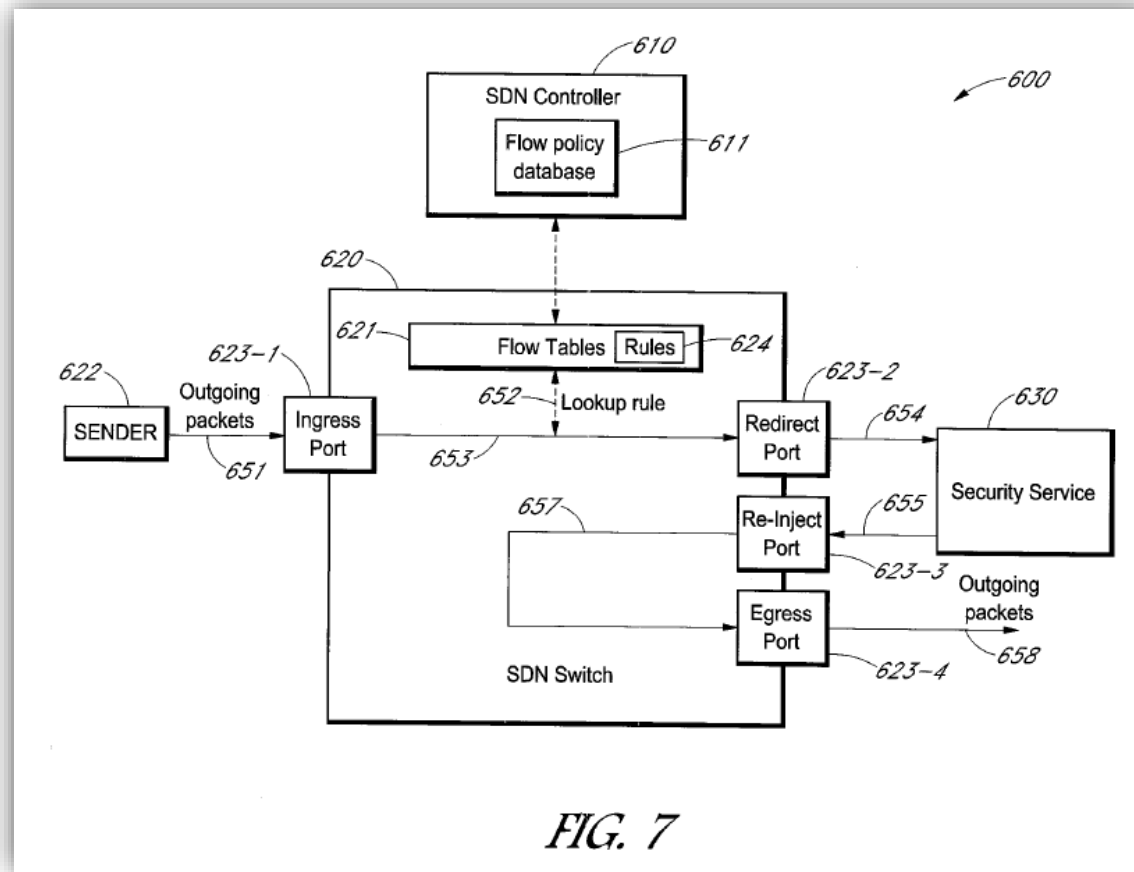
EX1005, Figure 3

Figure 4 of Lin



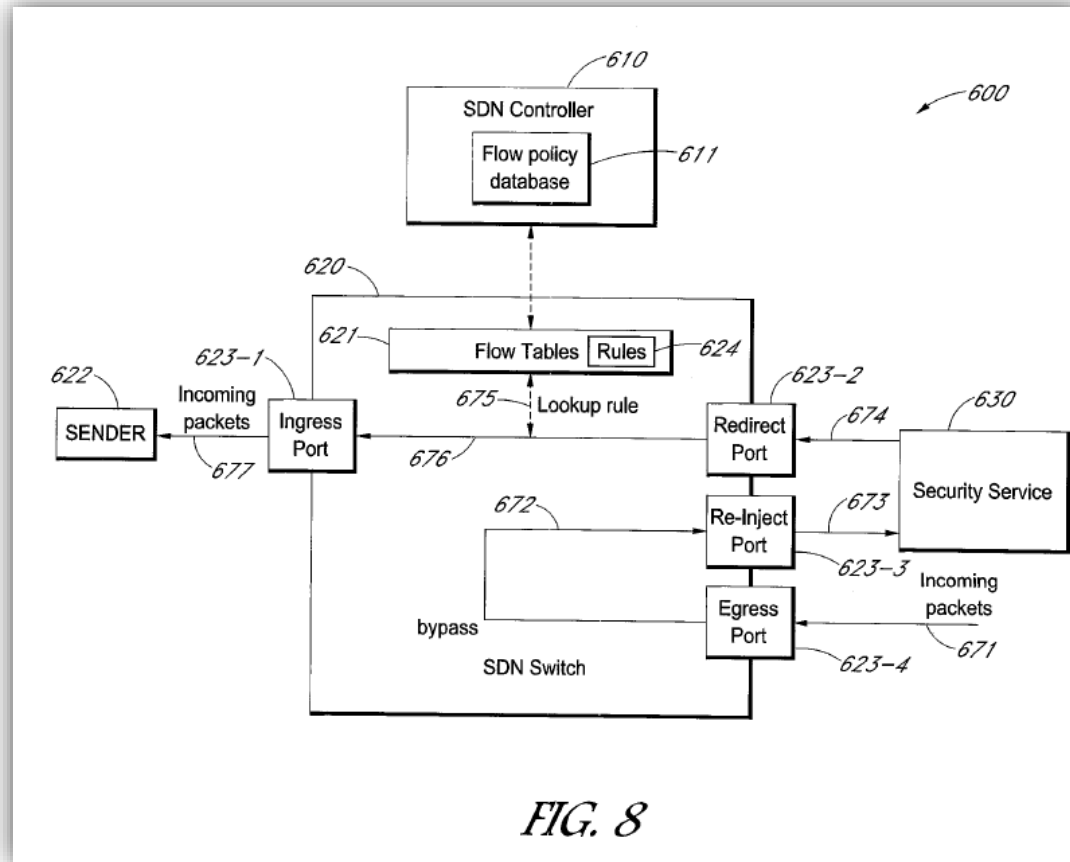
EX1005, Figure 4

Figure 7 of Lin



EX1005, Figure 7

Figure 8 of Lin



EX1005, Figure 8

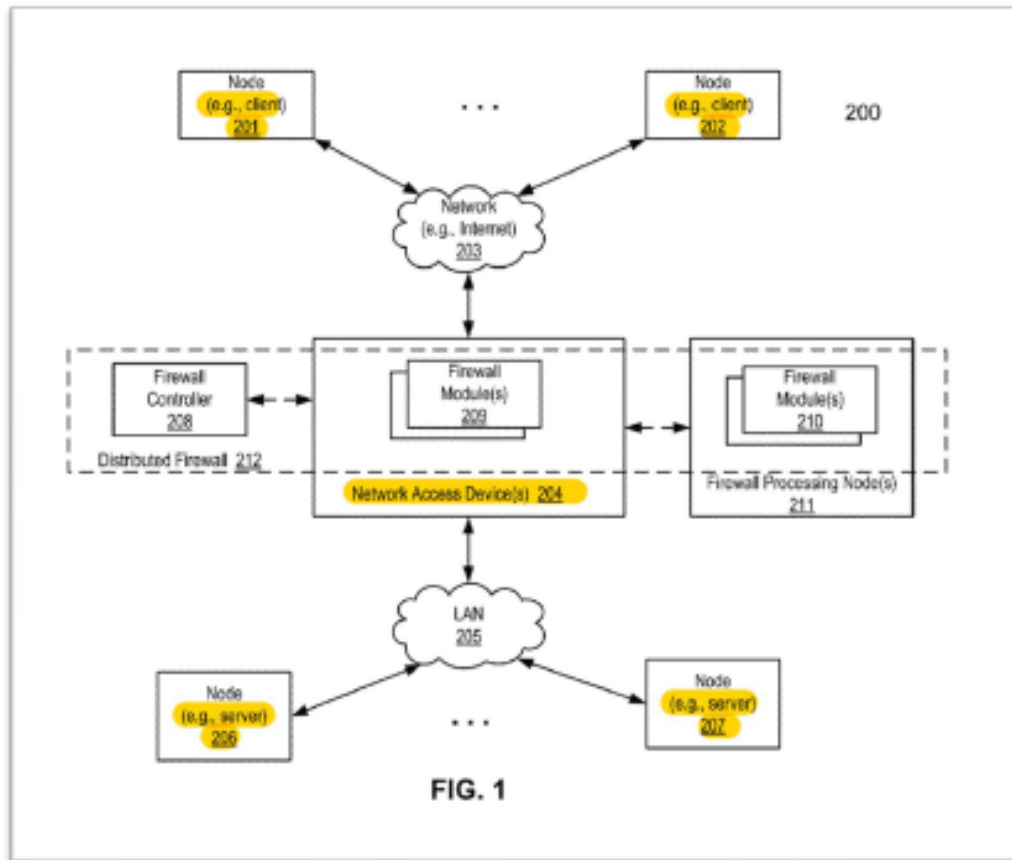
Shieh Relates To Network Security

FIELD OF THE INVENTION

[0002] Embodiments of the present invention relate generally to **network security**. More particularly, embodiments of the invention relate to enabling network security with network equipment.

EX1006, ¶[0002]

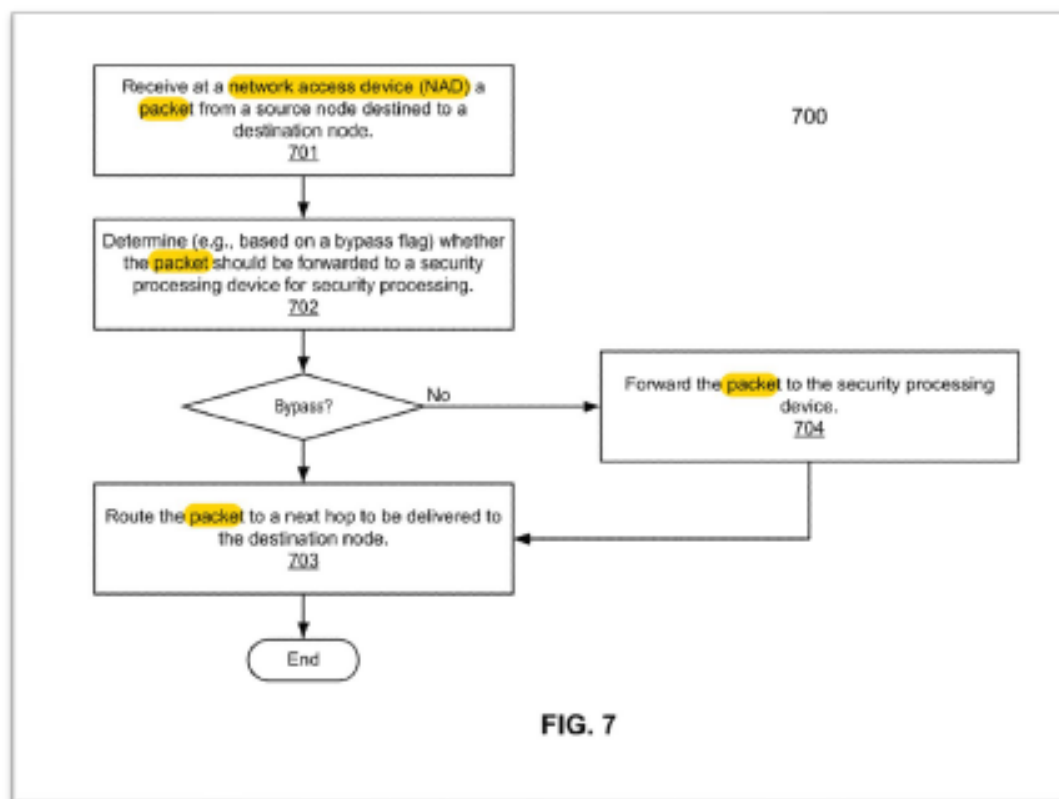
Shieh Discloses A Client And A Server Communicating Through The Network Access Device



EX1004, ¶234

Figure 1 of Shieh (Annotated)

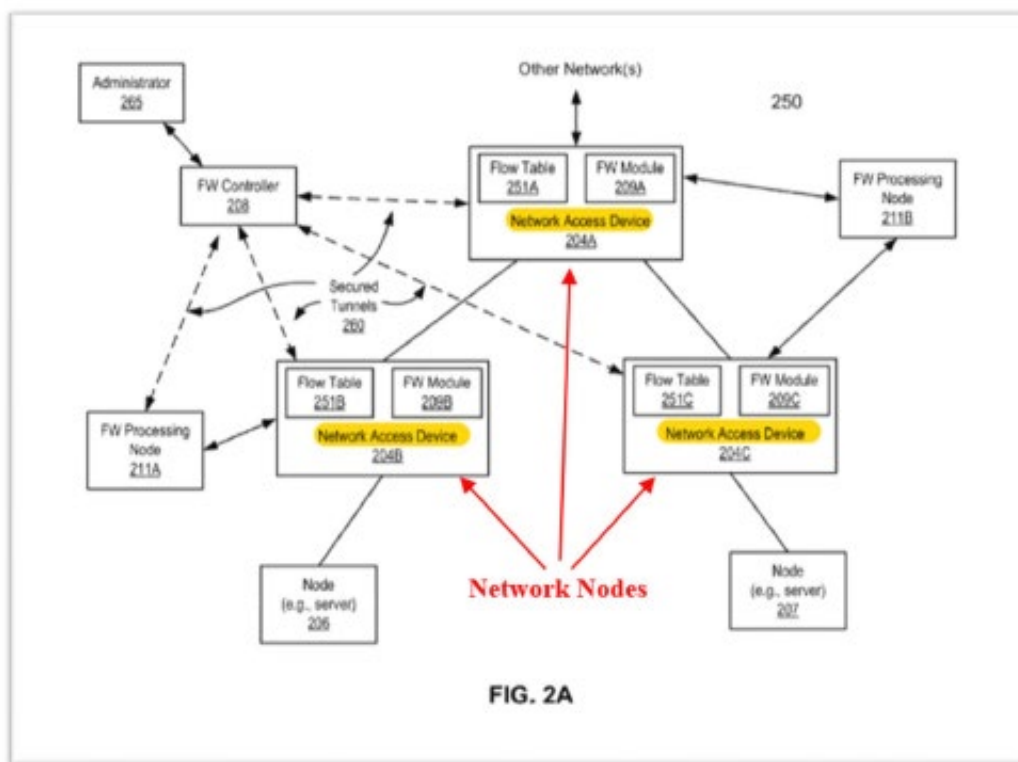
Shieh Discloses Packets Flowing Through A Network Access Device



EX1004, ¶209

Figure 7 of Shieh (Annotated)

Shieh's Network Access Devices Are The Claimed Network Nodes



EX1004, ¶210

Figure 2A of Shieh (Annotated)

Shieh Discloses The Claimed Controller

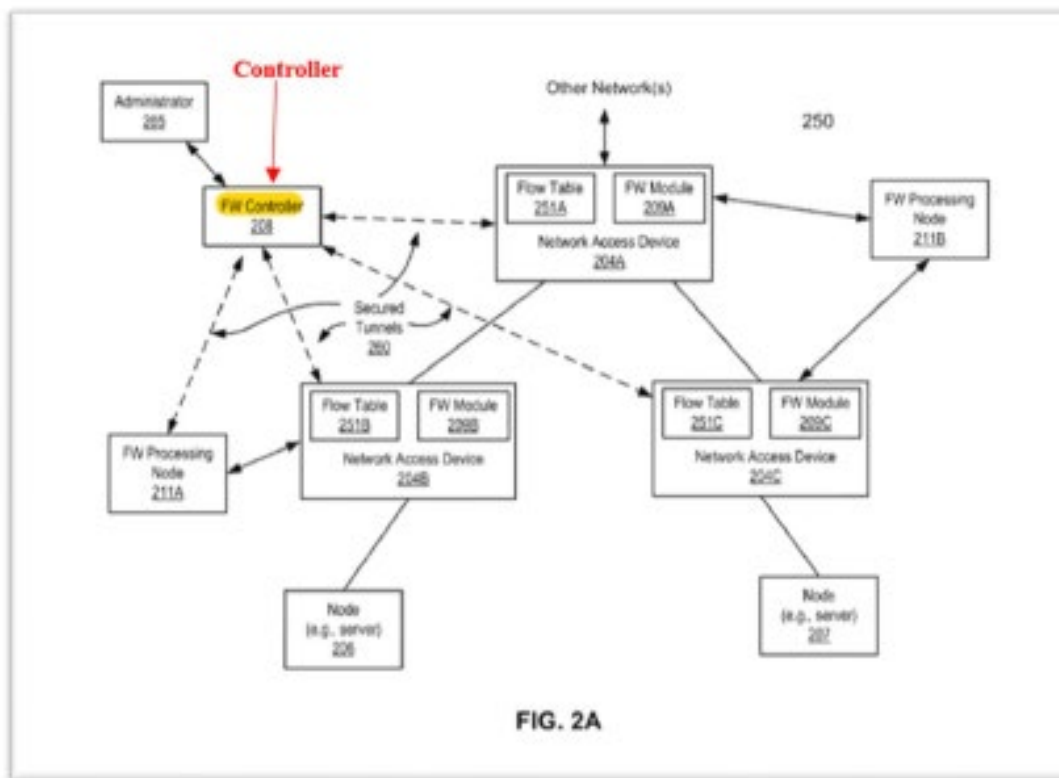


Figure 2A of Shieh (Annotated)

EX1004, ¶211, ¶216

Shieh Discloses A Flexible Security System That Optimizes The Use Of Resources

BACKGROUND

[0003] Network security devices are designed to be put in the data path of the network traffic in order to inspect and control the network traffic. A popular way to deploy network security is so called “bump-in-the-wire” that the devices are deployed in the data path for security inspection. However, it may not be practical to deploy security devices on every data path of network traffic in a data center. There are also needs to be able to flexibly perform security inspection on different parts of networks. The traditional bump-in-the-wire deployment cannot fulfill the needs.

EX1006, ¶[0003]

Shieh Performs Dynamic Security Inspection On Network Packets

[0017] According to some embodiments, a mechanism is utilized to dynamically perform security inspection in a network. In one embodiment, the mechanism includes two functions: 1) an input/output (IO) function that performs the distribution of network traffic; and 2) a security-processing function that performs security processing, including security inspection and policy enforcement. The IO function receives the packets and uses a session table to forward the packets to the security-processing function. A session table is a data structure that stores connection states, including the destination of the security-processing function. In one embodiment,

EX1006, ¶[0017]

Shieh Discloses That A Command From An Administrator Can Control How Packets Are Forwarded

tion of the security-processing function. In one embodiment, the IO function determines, based on an internal data structure such as a session or flow table, whether the packet should be forwarded to the security processing function for security inspection. The configuration of the IO function to control whether to forward the packets to the security processing function can be set based on a command received from an administrator or alternatively, based on a signal received from the security processing function.

EX1006, ¶[0017]

Shieh's Controller Configures A Network Access Device To Set Up Filtering Rules

[0018] According to one embodiment, an administrator can configure, for example, via a controller or a management entity, a network access device to set up a set of filtering rules specifying whether and/or what types of packets should be forwarded to a security device and which of the security devices for security inspection. In this embodiment, the controller is configured to manage multiple network access

EX1006, ¶[0018]

Shieh's Network Access Device Can Forward Certain Packets To A Security Device

troller is configured to manage multiple network access devices and/or multiple security devices. Alternatively, a security device may inform a network access device that subsequent packets of a particular session should be forwarded from the network access device for security inspection. In one embodiment, a security device performs the security inspection at the beginning of the flow or session, and at a certain point, the security device decides that it no longer needs to inspect further packets of the same session.

EX1006, ¶[0018]

Shieh's System Operates On Switches Or Routers

[0019] Advantages of embodiments of the present invention include, without limitation, providing a way to integrate partial network security functions into other network equipment, such as switches or routers. The integration allows network administrators to turn on security inspection functionality when there are needs for such, thus one can flexibly perform security inspection if needed. The notification between I/O functions and security-processing functions can reduce the number of packets to be inspected, thus enhancing the performance without lax the network security.

EX1006, ¶[0019]

Shieh Discloses DPI

[0021] According to one embodiment, network access device 204 is associated with a distributed firewall 212 that includes various firewall processing modules, for example, each being executed within a virtual machine (VM). In one embodiment, each firewall module is responsible for performing one or more firewall functions, but it does not include all of the firewall functions of a firewall. Examples of the firewall functions include, but are not limited to, network address translation (NAT), virtual private network (VPN), deep packet inspection (DPI), and/or anti-virus, etc. In one embodiment, some of the firewall processing modules are located within network access device 204 (e.g., firewall mod-

EX1006, ¶[0021]

Shieh Can Utilize A Session Table To Store Rules Used To Decide What Packets Are Sent To A Security Processing Function

2) a security-processing function (e.g., firewall module(s) 210) that performs security processing, including security inspection and policy enforcement. IO function 209 receives the packets and uses a session table to forward the packets to security-processing function 210. A session table is a data structure that stores connection states, including the destination of security-processing function. In one embodiment, IO

EX1006, ¶[0023]

Shieh Can Utilize A Session Table To Store Rules Used To Decide What Packets Are Sent To A Security Processing Function

function 209 determines, based on an internal data structure such as a session or flow table (e.g., session table as shown in FIG. 5), whether the packet should be forwarded to security processing function 210 for security inspection. The configuration of IO function 209 to control whether to forward the packets to security processing function 210 can be set based on a command received from an administrator or alternatively, based on a signal received from security processing function 210.

EX1006, ¶[0023]

Shieh Discloses A Persistent Connection Between Its Controllers And Its Network Access Devices

[0025] According to one embodiment, each of network access devices 204A-204C maintains a persistent connection such as secure connections or tunnels 260 with a controller or management entity 208 for exchanging management messages and configurations, or distributing routing information to network access devices 204A-204C, etc. In one embodiment, controller 208 communicates with each of the network access devices 204A-204C using a management protocol such as the OpenFlow™ protocol. OpenFlow is a Layer 2 communications protocol (e.g., media access control or MAC layer) that gives access to the forwarding plane of a network switch or router over the network. In simpler terms, OpenFlow allows the path of network packets through the network of switches to be determined by software running on multiple routers (minimum two of them, primary and secondary, having a role of observers). This separation of the control from the forwarding allows for more sophisticated traffic management than is feasible using access control lists (ACLs) and routing protocols.

EX1006, ¶[0025]

Shieh Can Utilize OpenFlow Technology

[0026] The OpenFlow technology consists of three parts: flow tables installed on switches, a controller, and an OpenFlow protocol for the controller to talk securely with switches. Flow tables are set up on switches or routers. Controllers talk to the switches via the OpenFlow Protocol, which is secure, and impose policies on flows. For example, a simple flow might be defined as any traffic from a given IP address. The rule governing it might be to route the flow through a given switch port. With its knowledge of the network, the controller could set up paths through the network optimized for speed, fewest number of hops or reduced latency, among other characteristics. Using OpenFlow takes control of how traffic flows through the network out of the hands of the infrastructure, the switches and routers, and puts it in the hands of the network owner (such as a corporation), individual users or individual applications.

Shieh Discloses That An Administrator Configures The Network Access Devices Via A Controller

[0029] According to one embodiment, an administrator 265 configures, for example, via a controller or a management entity 208, a network access device (e.g., network access devices 204A-204C) to set up a set of filtering rules concerning whether and/or what types of packets should be forwarded to a security device and which of the security devices (e.g., security devices 211A-211B) for security inspection. In this embodiment, controller 208 is configured to manage multiple network access devices 204A-204C and/or multiple security devices 211A-211B. Alternatively, a security device, such as security device 211A, may inform a network access device, such as network access device 204B, whether subsequent packets of a particular session should be forwarded from the network access device for security inspection. A security device may perform the security inspection on packets at the beginning of the flow or session, and at a certain point, the security device decides that it no longer needs to inspect further packets of the same session.

EX1006, ¶[0029]

Shieh Discloses That Security Inspection Can Be Bypassed For Certain Packets

[0035] An embodiment of the invention also controls the communication between I/O functions and security-processing functions to enable packets to bypass security-processing function if there is no more need to inspect the packets of the connection. Some of the security functions do not need to inspect all the packets of a connection. For examples, to identify the application of a connection, there may be only need to inspect first four or five packets to make the identification. In this case, the security-processing function can notify I/O functions to bypass the security-processing function for the rest of the packets of the connections. Once the I/O function receives the notification, it will forward the packets out without redirecting the packets to the security-processing functions. This would greatly improve the performance even when security inspection is turned on.

EX1006, ¶[0035]

The Use Of Shieh's Bypass Flag

[0037] FIG. 2B is a processing flow diagram illustrating a process of security inspection according to one embodiment of the invention. Referring to FIG. 2B, as an example, network switch 272 may represent any of network access devices 204A-204C and security device 273 may represents any of security processing devices 211A-211B as described above with respect to FIG. 2A. When device 272 receives a packet from a source node 271 via transaction 281, device 272 may determine whether the packet should be forwarded to security device 273. For example, device 272 may look up in its session table such as the one as shown in FIG. 5 to determine whether a bypass flag has been set to a predetermined value. If the bypass flag matches the predetermined value, the packet is forwarded to security device 273 via path 282; otherwise, the packet is routed to destination node 274. Alternatively, if there is no entry in the session table corresponding to the current session, the packet will also be transmitted to security device 273. After network device 272 receives a response from security device 273 via path 283, dependent upon the

EX1006, ¶[0037]

Shieh's Disclosure Of Load Balancing

ciated connection and forwards the packet to the corresponding security processing module identified by the forwarding table. If it cannot find the connection in its local cache, the packets are forwarded to central controller 208 for processing. In such a case, controller 208 assigns the connection to one of security processing modules 309-311 based on one or more of a variety of factors such as load balancing. The virtual I/O modules 302-304 can be located at multiple locations of the networks to receive and send out packets.

EX1006, ¶[0039]

Shieh's Controller Has "Central Control And Monitoring Functionality"

[0043] By dividing a firewall into different modules, it allows putting virtual I/O and security processing functions at the best locations to protect the network entrance, while keeping the central control and monitoring functionality at the central controller. It also enhances the scalability of the system since all modules can be expanded to multiple instances to share the system load. Note that a service processing module is optional in the architecture, as it is only required when there are needs to use additional resources to handle the security functions.

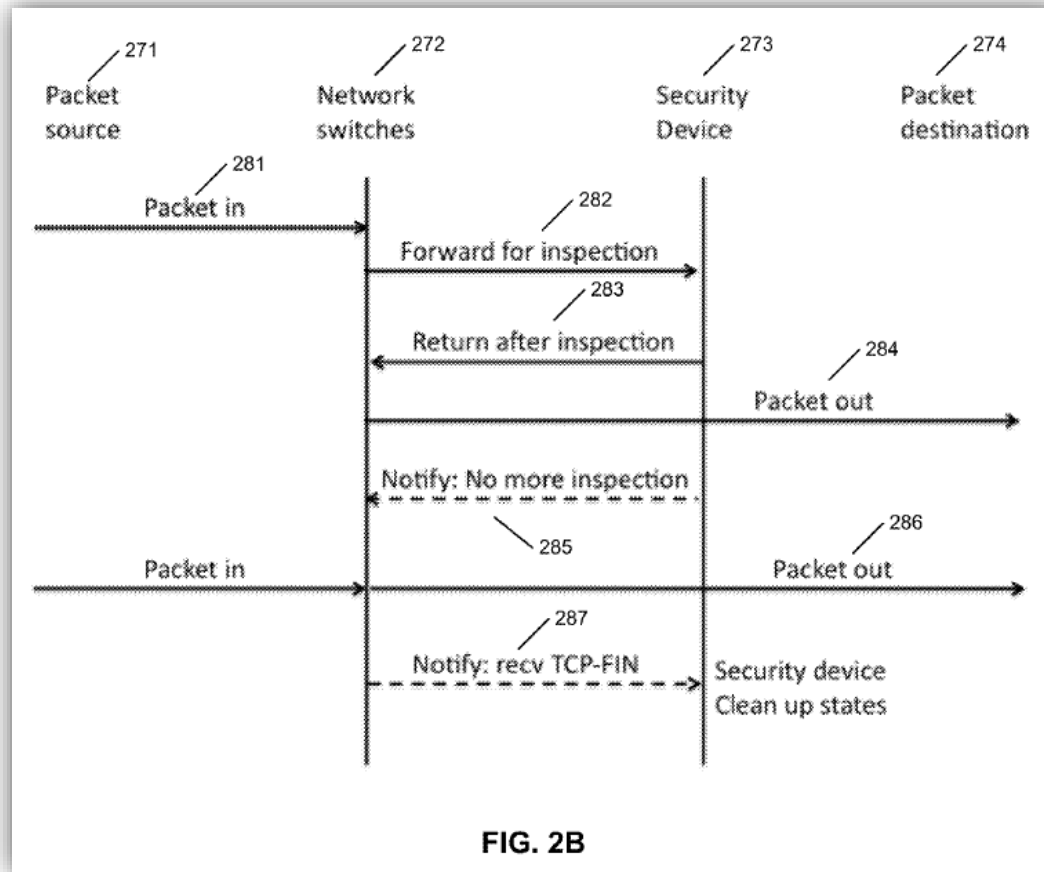
EX1006, ¶[0043]

Shieh – Claim 1

1. A computer-implemented method, comprising:
receiving at a network access device a packet from a source node destined to a destination node;
examining a data structure maintained by the network access device to determine whether the data structure stores a data member having a predetermined value, the data member indicating whether the packet should undergo security processing;
if the data member matches the predetermined value,
transmitting the packet to a security device associated with the network access device to allow the security device to perform content inspection, and
in response to a response received from the security device, routing the packet to the destination node dependent upon the response; and
transmitting the packet to the destination node without forwarding the packet to the security device, if the data member does not match the predetermined value.

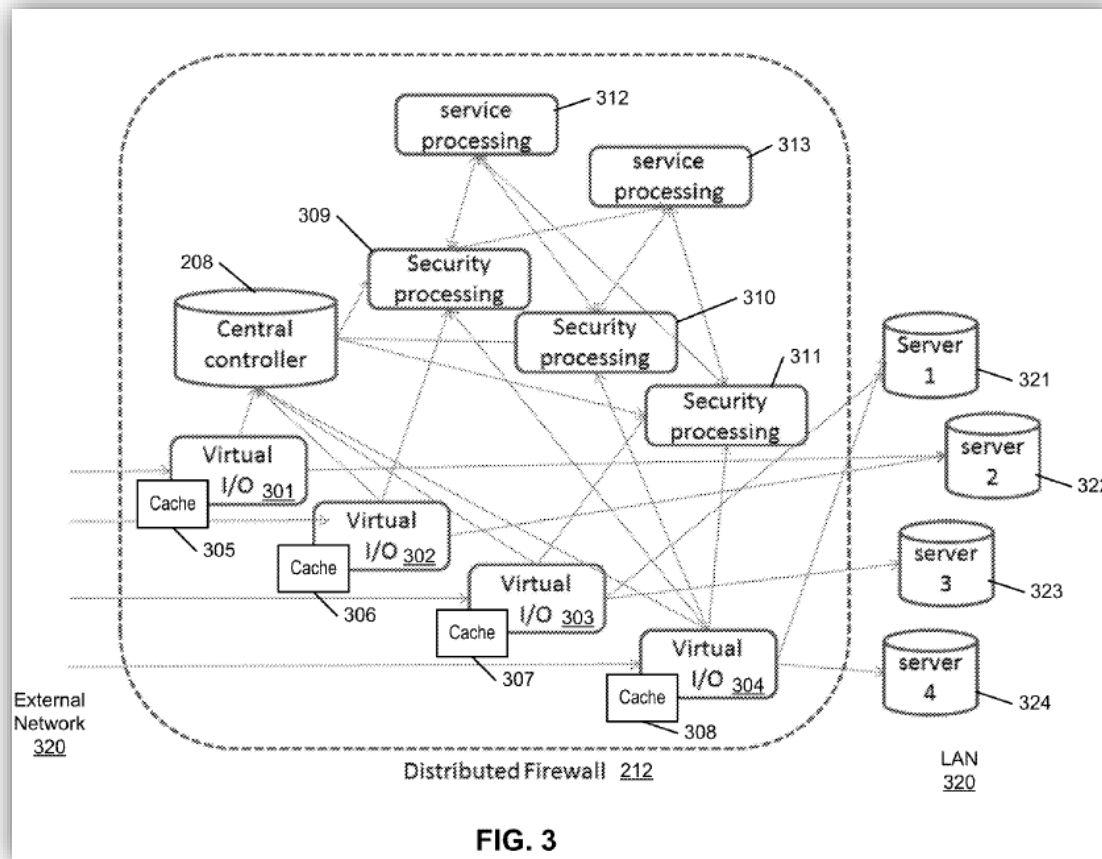
EX1006, Claim 1

Figure 2B of Shieh



EX1006, Figure 2B

Figure 3 of Shieh



EX1006, Figure 3

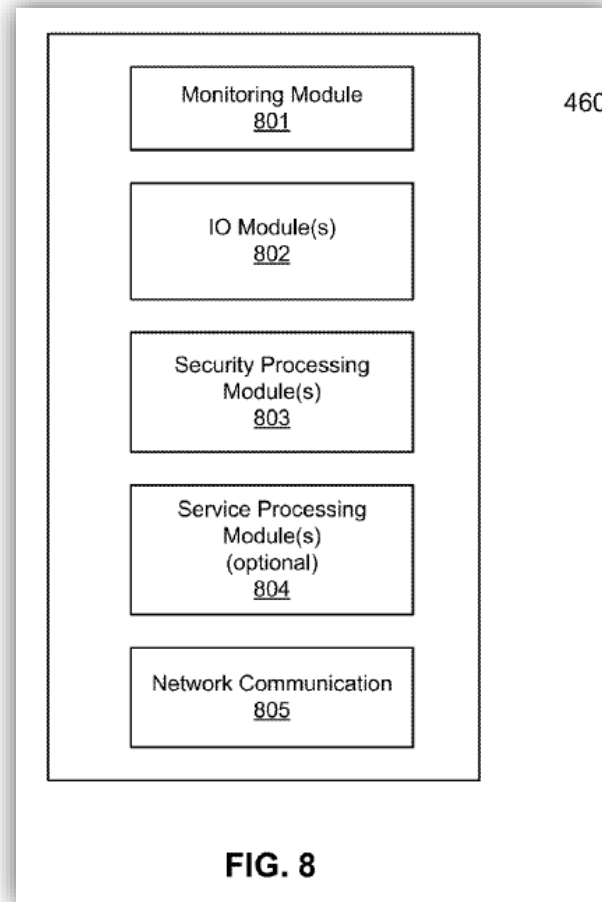
Figure 5 of Shieh

Session/Flow ID	Address(es) of Security Processing Module(s)	Bypass Flag <u>501</u>	Notification Event(s) <u>502</u>	Filtering Rule(s) <u>503</u>
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...

FIG. 5

EX1006, Figure 5

Figure 8 of Shieh



EX1006, Figure 8

Swenson Teaches “Real-Time Traffic Monitoring” Using A Network Controller

Real-Time Traffic Monitoring

[0023] FIG. 1 illustrates a high-level block diagram of an example communications environment 100 for selective on-demand real-time network monitoring and subscriber identification. The environment 100 comprises user devices 110, an origin server 160, a steering device 130, a network controller 140, a video optimizer 150, and a network 120. The network 120 is a communication network that transmits data between the user devices 110, the steering device 130 and the origin server 160 and/or the video optimizer 150. In one embodiment the network 120 includes wireless network and the Internet.

EX1007, ¶[0023]

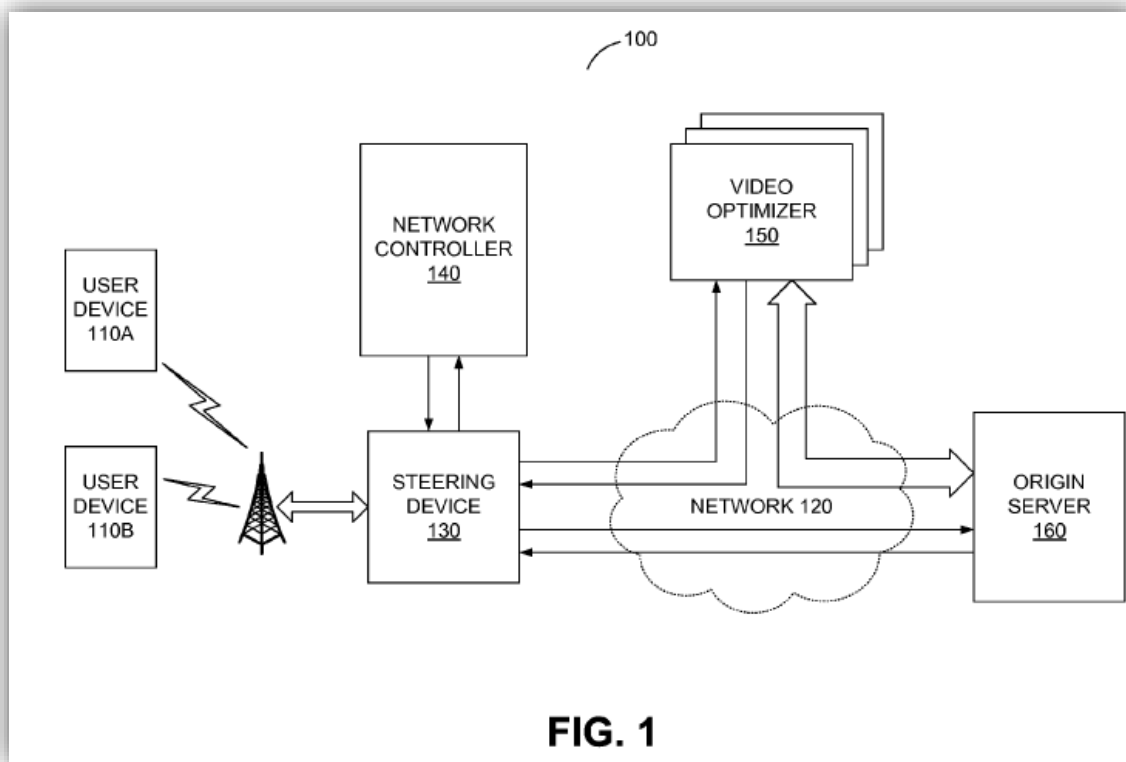
Swenson Discloses A System That Maximizes Available Bandwidth

[0018] Embodiments disclosed include a network controller system for real-time data gathering on the state of existing

network traffic flows and mapping flow data to respective users in the network to predict available bandwidth and level of congestion. By gathering a history of flow statistics in the network, the network controller system establishes a relationship between base stations (or other network segments) and their capability to deliver the amount of data typically required by a particular user of the network. The very recent

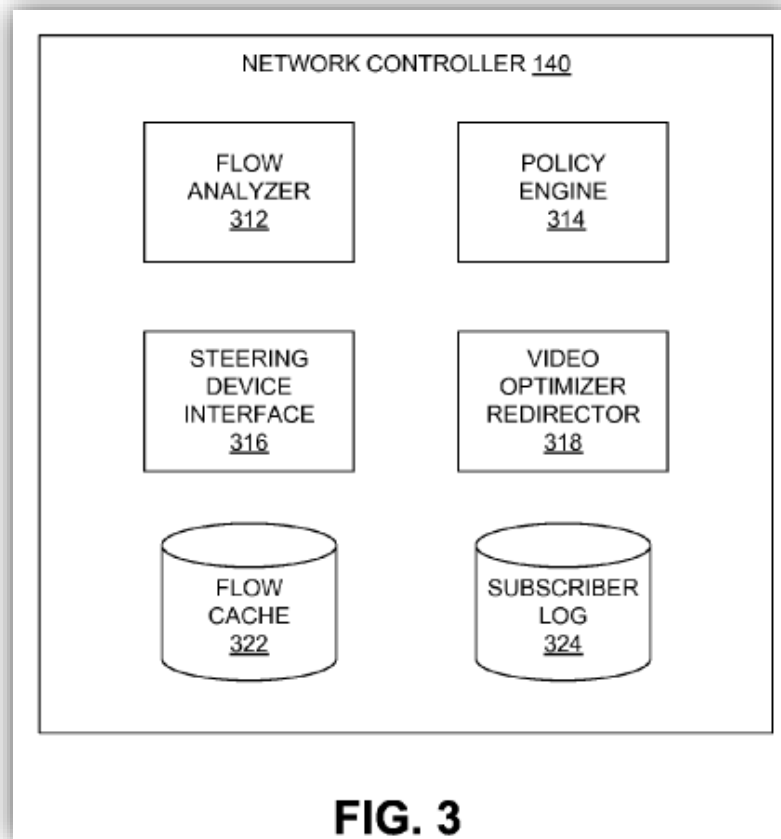
EX1007, ¶[0018]

Figure 1 of Swenson



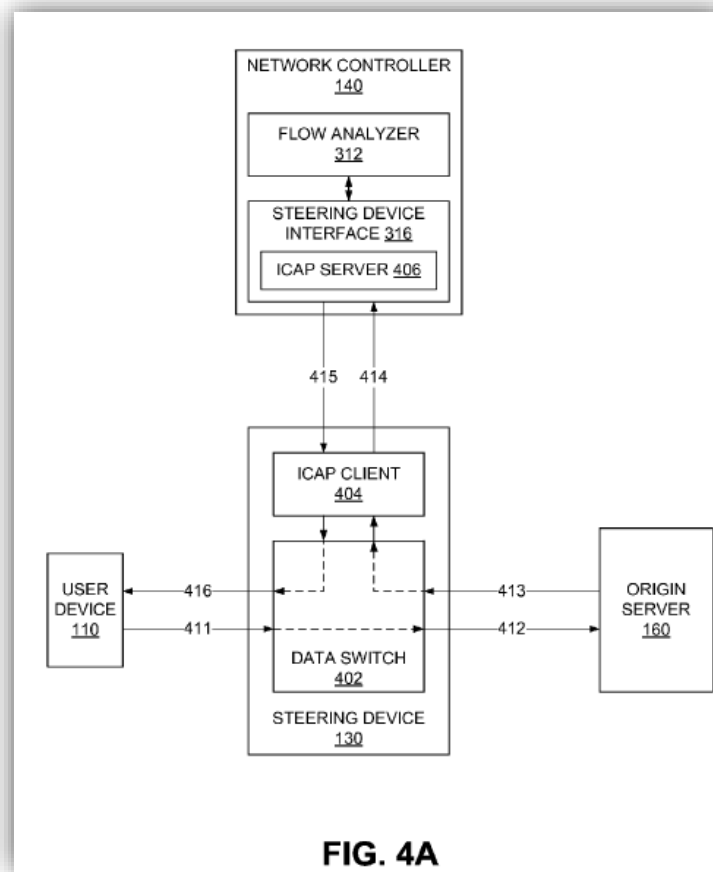
EX1007, Figure 1

Figure 3 of Swenson



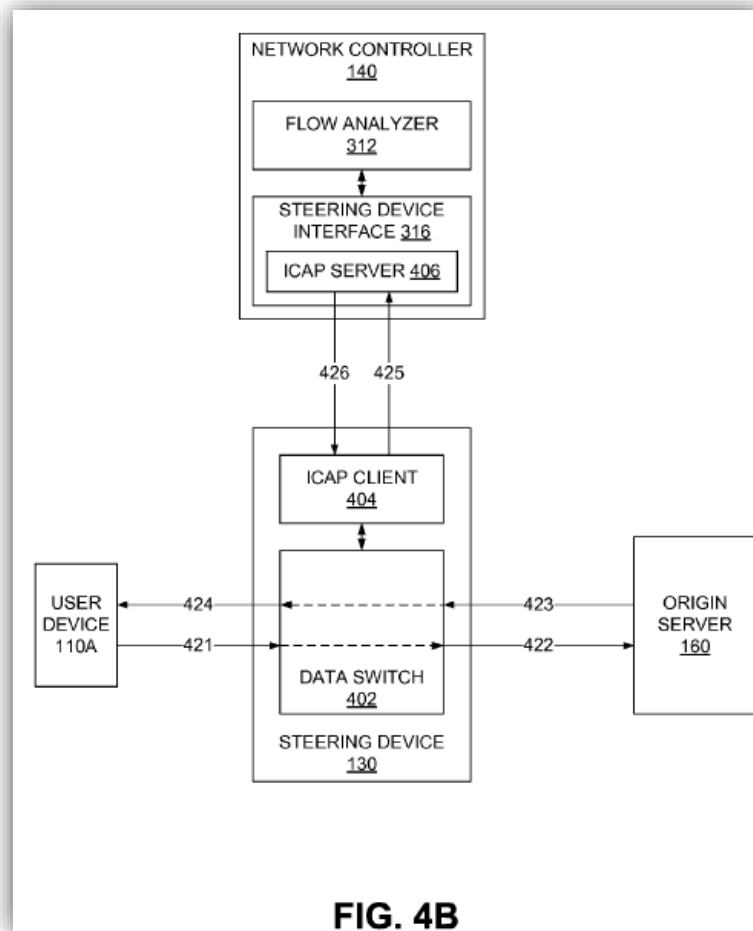
EX1007, Figure 3

Figure 4A of Swenson



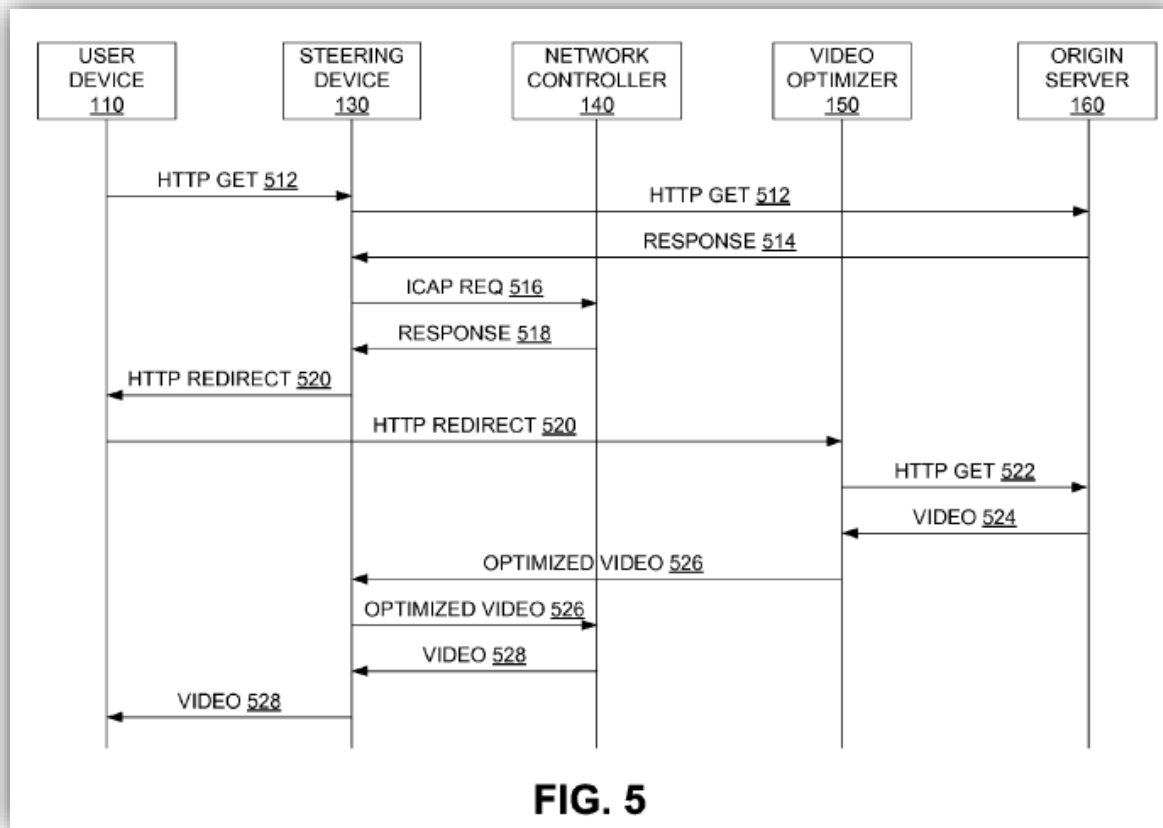
EX1007, Figure 4A

Figure 4B of Swenson



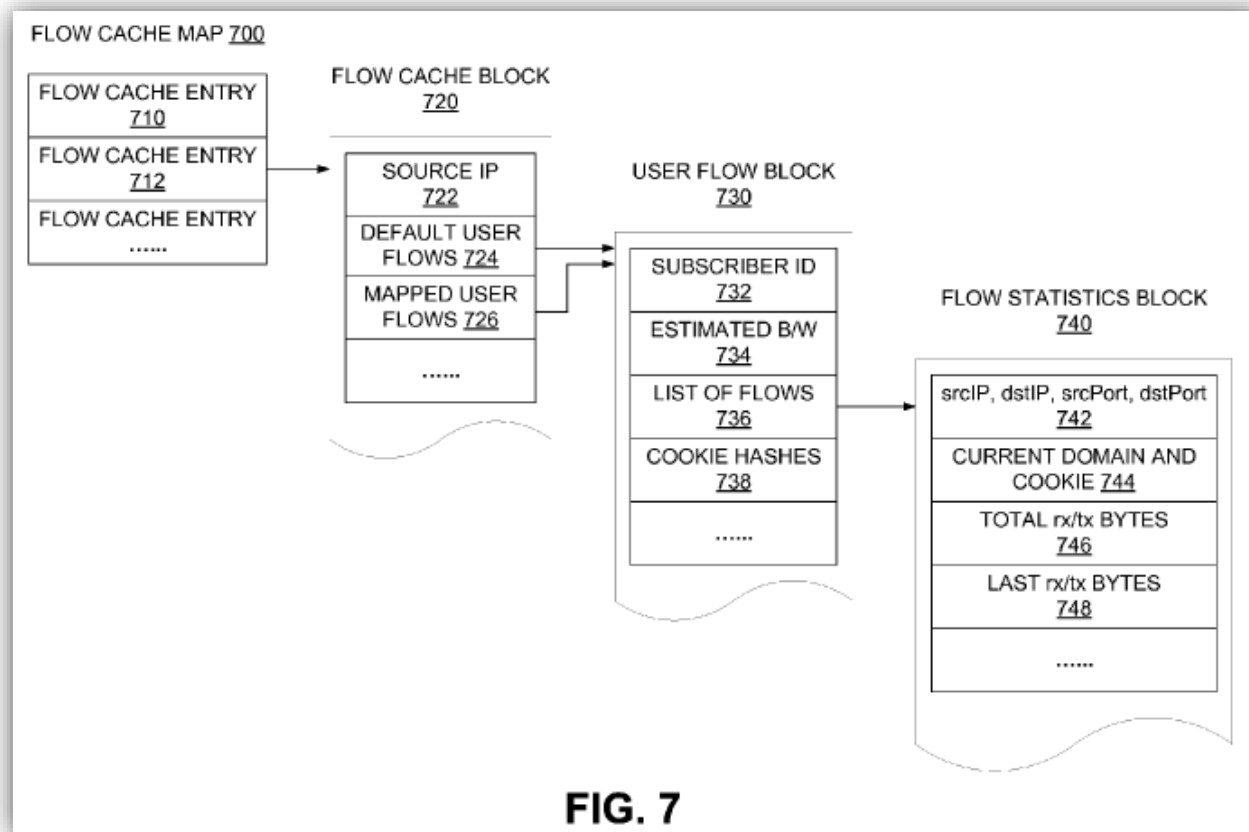
EX1007, Figure 4B

Figure 5 of Swenson



EX1007, Figure 5

Figure 7 of Swenson



EX1007, Figure 7

Swenson Categorizes Network Traffic, Including HTTP Traffic Flows

[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 interest for further inspection at the network controller 140. Alternatively, the network controller 140 interfaces with the steering 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.

EX1007, ¶[0026]

Swenson's Controller Is Out-Of-Band And Inspects Network Packets For Traffic Monitoring And Optimization

[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 steering 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 implemented 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 monitoring 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-

EX1007, ¶[0028]

Swenson Teaches That Network Packets Are Sent To The Controller For “Inspection”

[0029] As a flow is sent to the network controller 140 for inspection, historical network traffic data stored at the network controller 140 may be searched. The historical network traffic data includes information such as subscriber information, the cell towers to which the user devices attached, routers through which the traffic is passing, geography regions, the backhaul segments, and time-of-day of the flows. For

EX1007, ¶[0029]

Swenson Discloses That Its System Performs “Security Functions”

[0039] Referring now to FIG. 3, it illustrates one embodiment of an example architecture of the network controller 140 for providing selective real-time network monitoring and subscriber identification. The network controller 140 comprises 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 consoles, and the like are not shown so as to not obscure the details of the system architecture.

EX1007, ¶[0039]

Swenson Provides “On-Demand Real-Time Network Monitoring”

Network Monitoring and Congestion Detection

[0056] FIGS. 4A and 4B each illustrates one embodiment of an example working mode of the network controller for providing selective on-demand real-time network monitoring and subscriber identification. Shown with the network controller 140 are the user device 110, the steering device 130, and the origin server 160. The network controller 140 is

EX1007, ¶[0056]

Swenson Performs DPI

[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 matching 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 network controller 140 performs a deep flow inspection to determine 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 analyzer 312 may also determine if the flow needs to be optimized based on historical flow statistical data.

Swenson Discloses That Its System Performs Inspection Of Network Packets

[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 function, 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 forwards 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**.