Cisco Systems, Inc. and Juniper Networks, Inc.
v.
Orckit Corporation

Patent Owner Demonstratives
IPR2023-00554
U.S. Patent No. 10,652,111
July 9, 2024

James Carmichael
Steve McBride
Mitch Yang
Carmichael IP, PLLC

Michael Ng
George Stamatopoulos
Kobre & Kim LLP

The ’111 Patent

1.The ’111 Patent

2.The Challenged Claims Are Patentable

The ’111 Patent

(12) **United States Patent**
Barsheshet et al.

(10) **Patent No.: US 10,652,111 B2**
(45) **Date of Patent: May 12, 2020**

(54) **METHOD AND SYSTEM FOR DEEP PACKET INSPECTION IN SOFTWARE DEFINED NETWORKS**

(71) Applicant: **ORCKIT IP, LLC**, Newton, MA (US)

(72) Inventors: **Yossi Barsheshet**, Ashdod (IL); **Simhon Doctori**, Gan-Yavne (IL); **Ronen Solomon**, Ramat-Gan (IL)

(73) Assignee: **ORCKIT IP, LLC**, Dover, DE (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 306 days.

(21) Appl. No.: **15/126,288**

(22) PCT Filed: **Apr. 21, 2015**

(86) PCT No.: **PCT/US2015/026869**
§ 371 (c)(1),
(2) Date: **Sep. 15, 2016**

(87) PCT Pub. No.: **WO2015/164370**
PCT Pub. Date: **Oct. 29, 2015**

(65) **Prior Publication Data**

US 2017/0099196 A1 Apr. 6, 2017

Related U.S. Application Data

(60) Provisional application No. 61/982,358, filed on Apr. 22, 2014.

(51) **Int. Cl.**
H04L 12/26 (2006.01)
H04L 12/64 (2006.01)
(Continued)

(52) **U.S. Cl.**
CPC ***H04L 43/028*** (2013.01); ***H04L 12/6418*** (2013.01); ***H04L 43/026*** (2013.01);
(Continued)

(58) **Field of Classification Search**
CPC . H04L 43/026; H04L 12/6418; H04L 43/028; H04L 49/70; H04L 69/161
(Continued)

(56) **References Cited**

U.S. PATENT DOCUMENTS

2010/0208590 A1* 8/2010 Dolganow H04L 43/026 370/235
2010/0212006 A1 8/2010 Dolganow et al.
(Continued)

FOREIGN PATENT DOCUMENTS

EP 2672668 A1 12/2013

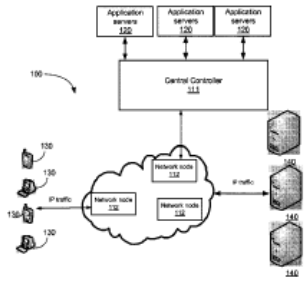
OTHER PUBLICATIONS

Supplementary Search Report of EP 15783292 dated Nov. 7, 2017.
(Continued)

Primary Examiner — Jae Y Lee
Assistant Examiner — Jean F Voltaire
(74) *Attorney, Agent, or Firm* — May Patents Ltd. c/o Dorit Shem-Tov

(57) **ABSTRACT**

A method for deep packet inspection (DPI) in a software defined network (SDN). The method includes 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, the first packet matches the at least one probe instruction and includes a first sequence number; receiving from a network node a second packet of the flow, the second packet matches the at least one probe instruction and includes a second sequence number, the second packet is a response of the first packet; computing a mask value respective of at least the first and second sequence numbers indicating which bytes to be mirrored from subsequent packets belonging to the same flow; generating at least one
(Continued)



(10) **Patent No.: US 10,652,111 B2**
(45) **Date of Patent: May 12, 2020**

(57) **ABSTRACT**

A method for deep packet inspection (DPI) in a software defined network (SDN). The method includes 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, the first packet matches the at least one probe instruction and includes a first sequence number; receiving from a network node a second packet of the flow, the second packet matches the at least one probe instruction and includes a second sequence number, the second packet is a response of the first packet; computing a mask value respective of at least the first and second sequence numbers indicating which bytes to be mirrored from subsequent packets belonging to the same flow; generating at least one

Ex. 1001 [’111 Patent], p. 1

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 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 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;

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.

Ex. 1001, Cl. 1.

The ’111 Patent

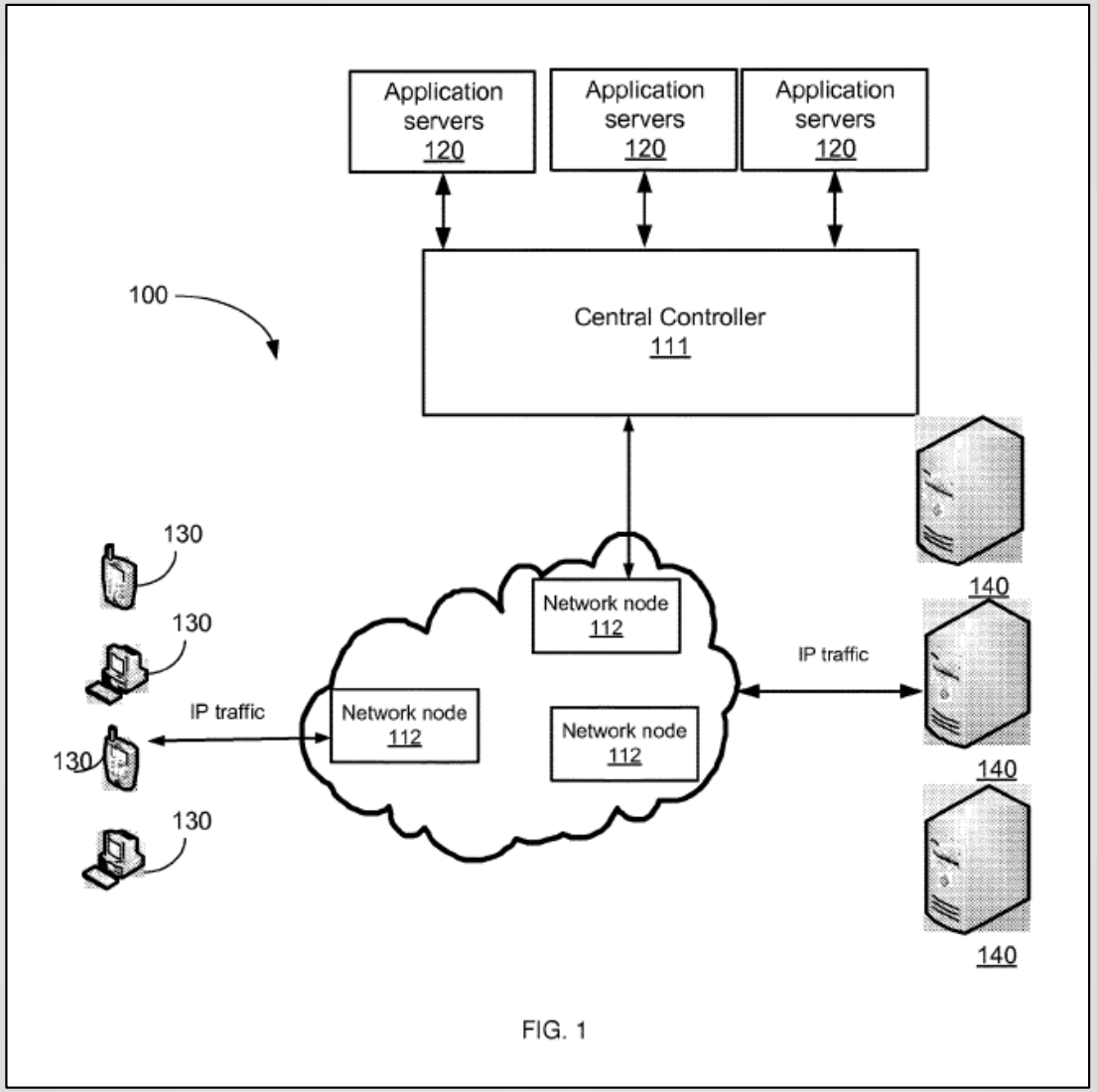


FIG. 1

Ex. 1001, Fig. 1

The ’111 Patent

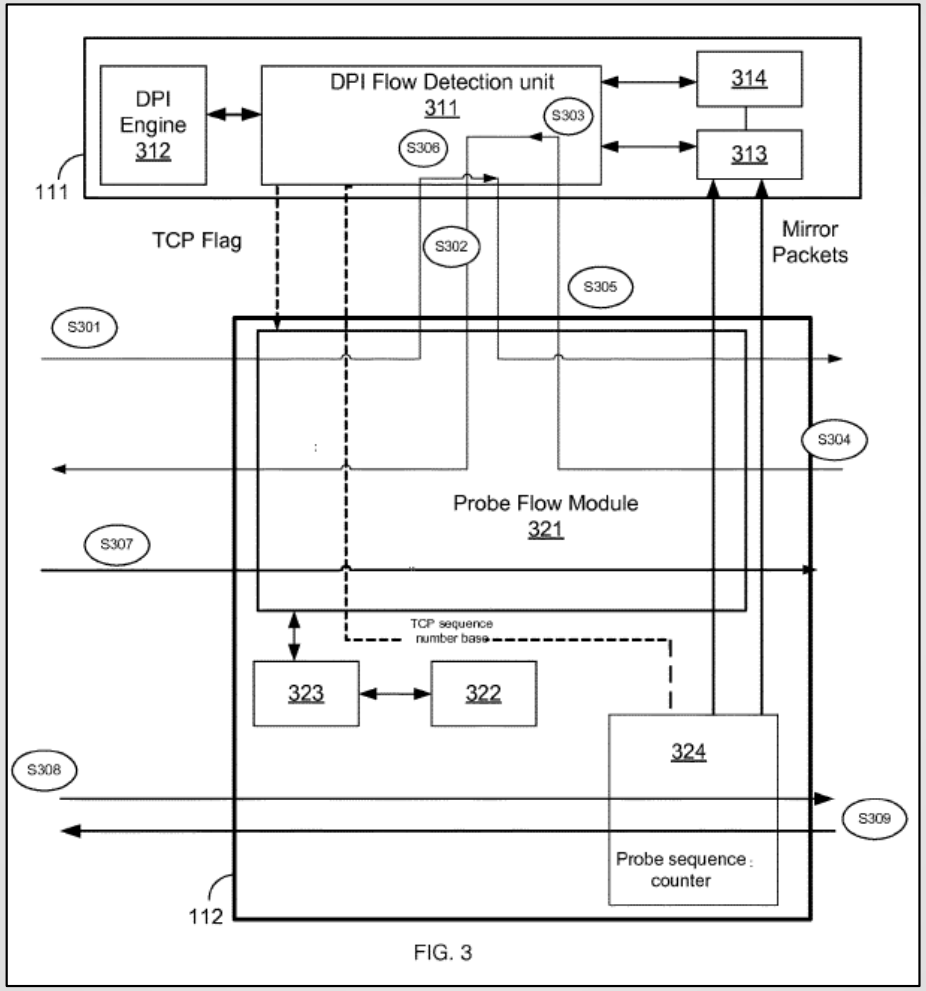


FIG. 3

Ex. 1001, Fig. 3

The Challenged Claims Are Patentable

1. The ’111 Patent

2. The Challenged Claims Are Patentable

The Challenged Claims Are Patentable

Ground #1: Claims 1-9, 12-24 and 27-31 of the '111 Patent are obvious under 35 U.S.C. § 103 over Lin in view of Swenson and the knowledge of a POSA.

Ground #2: Claims 1, 5-9, 12-24 and 27-30 of the '111 Patent are obvious under 35 U.S.C. § 103(a) over Shieh in view of Swenson and the knowledge of a POSA. Shieh was filed as a provisional application on April 11, 2012 and as a non-

Paper 1, p. 4-5

Lin

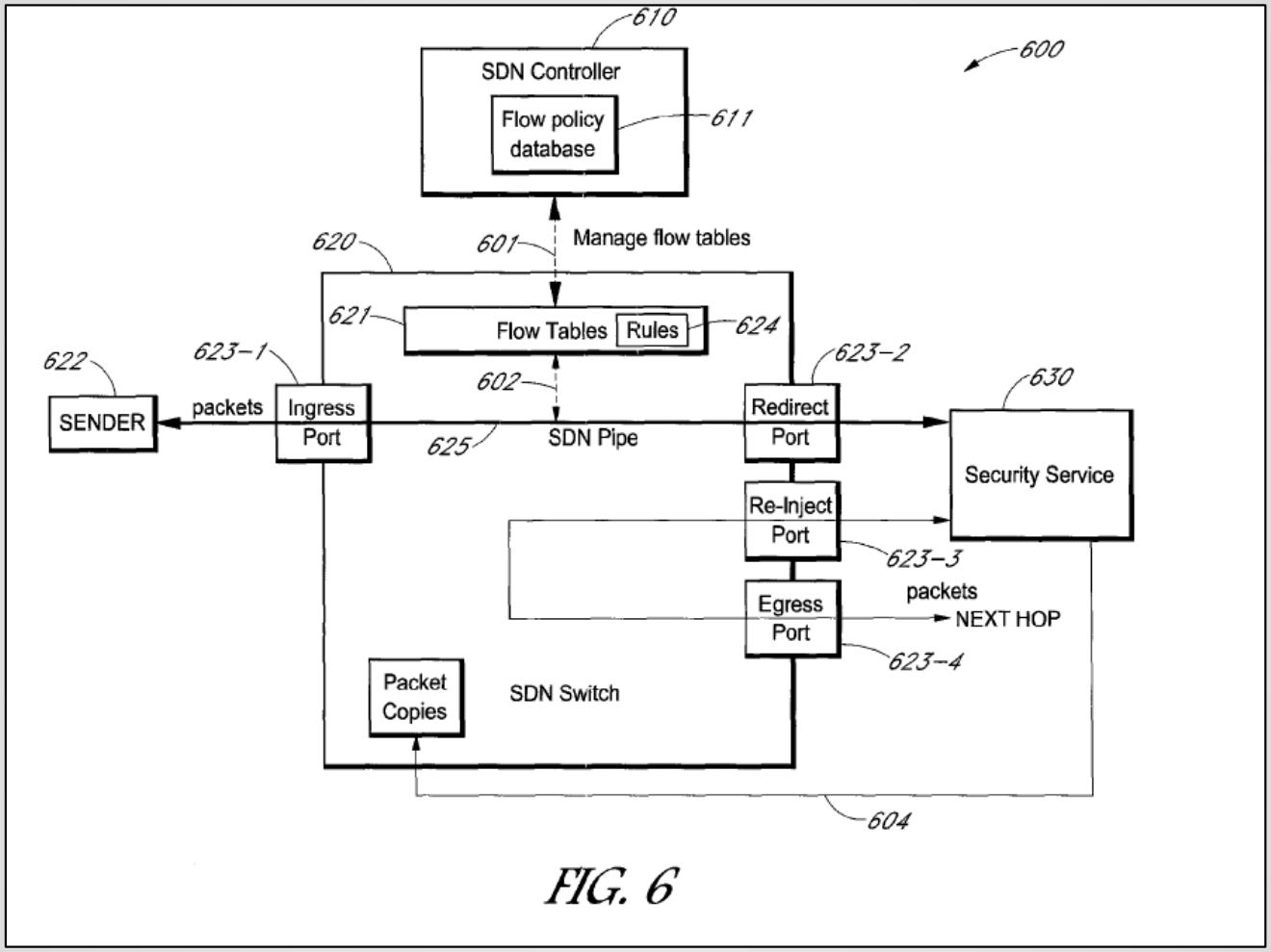


FIG. 6

Sur-reply, 2-3; Ex. 1005 [Lin], Fig. 6

Lin

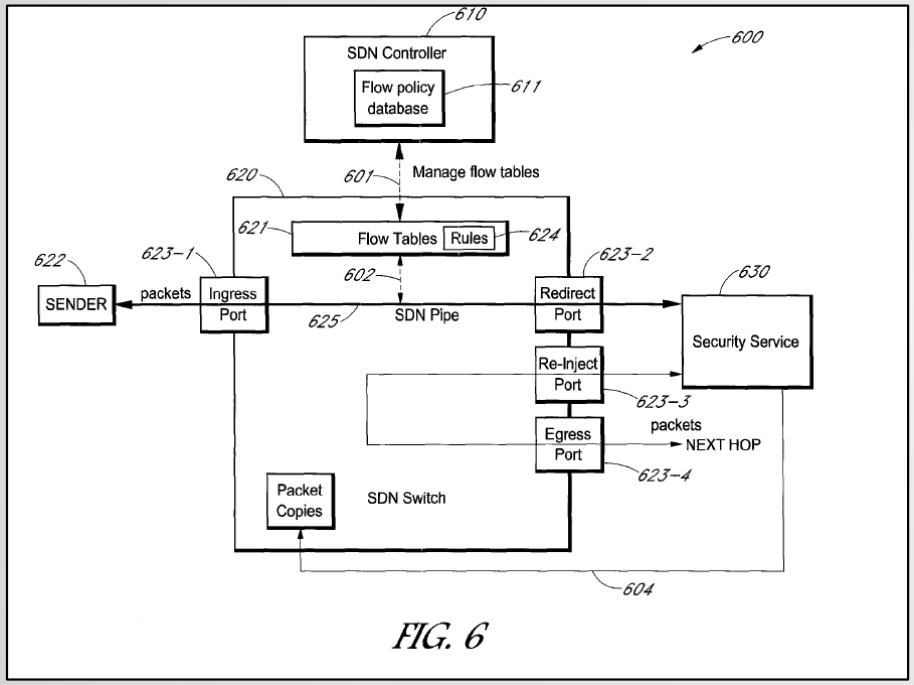


FIG. 6

Referring now to FIG. **6**, there is shown a schematic diagram of an SDN computer network **600** in accordance with an embodiment of the present invention. In one embodiment, the SDN computer network **600** is compliant with the OpenFlow™ protocol. Accordingly, in one embodiment, the SDN controller **610** comprises an OpenFlow™ controller and the SDN switch **620** comprises an OpenFlow™ switch. The SDN controller **610** and the SDN switch **620** comprise the control plane and data plane, respectively, of the SDN computer network **600**. The SDN computer network **600** may have a plurality of SDN switches **620** but only one is shown for clarity of illustration. The SDN controller **610** and the SDN switch **620** are logically separate components.

Sur-reply, 3-4; Ex. 1005 [Lin], Fig. 6, 3:45-52

Lin

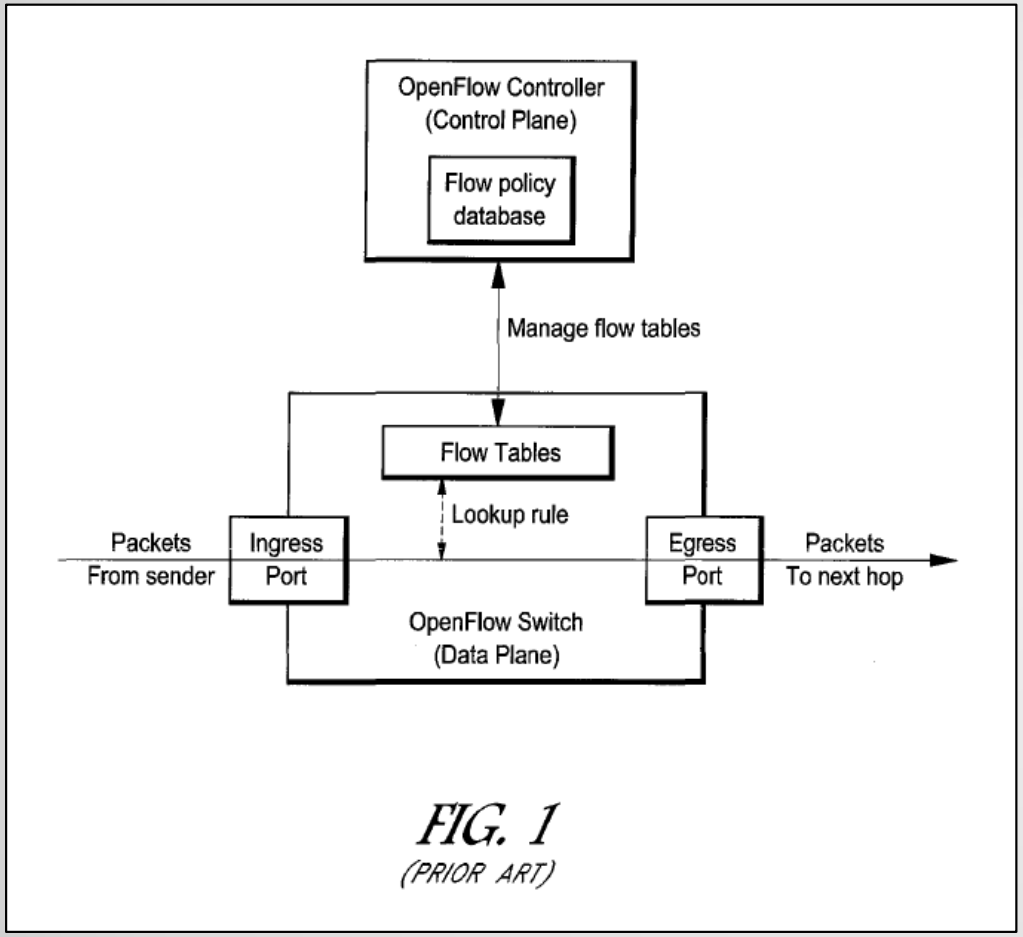


FIG. 1
(PRIOR ART)

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.

Sur-reply, 3-4; Ex. 1005, Fig. 1; Ex. 1001, 1:50-67

Lin

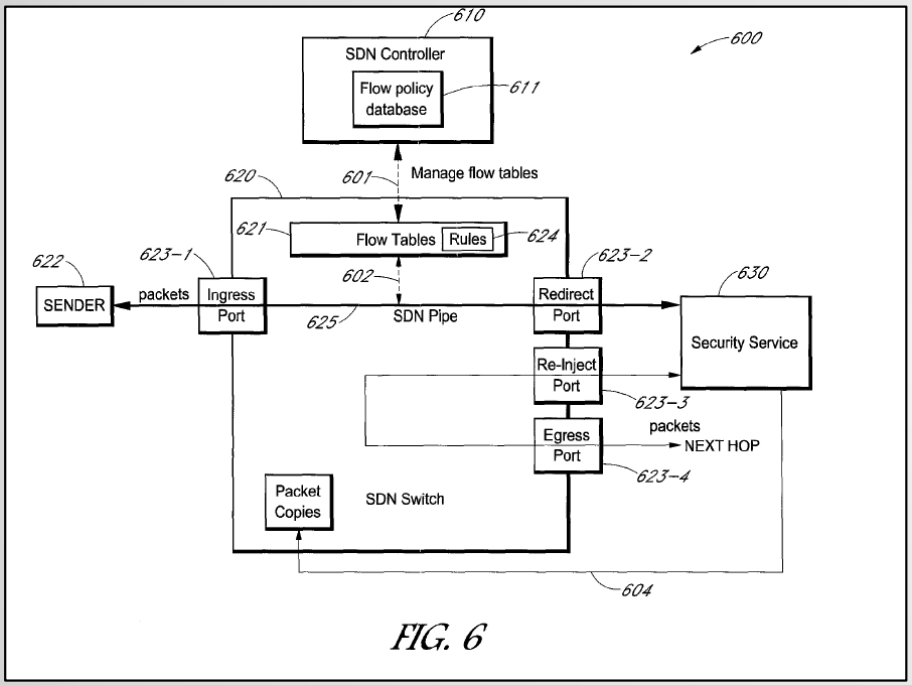


FIG. 6

Network security vendors provide network security services, such as firewall or deep packet inspection (DPI). Generally speaking, to provide network security services, packets of network traffic are intercepted for inspection. One way of intercepting network traffic is to place the security service in the middle of the packet forwarding path. This is illustrated in FIG. **3**, where packets from a sender component (e.g., a sender computer) are received in an ingress port of a switch, forwarded to an egress port of the switch, and forwarded to the ingress port of a security component, such as a security service. The security service may inspect the packets, and forward the packets to an egress port of the switch toward the next hop, which may be another switch or a destination component (e.g., destination computer), for example.

Sur-reply, 2-3;Ex. 1005, 3:11-24

Lin

Network security vendors provide network security services, such as firewall or deep packet inspection (DPI). Generally speaking, to provide network security services, packets of network traffic are intercepted for inspection. One way of intercepting network traffic is to place the security service in the middle of the packet forwarding path. This is illustrated in FIG. **3**, where packets from a sender component (e.g., a sender computer) are received in an ingress port of a switch, forwarded to an egress port of the switch, and forwarded to the ingress port of a security component, such as a security service. The security service may inspect the packets, and forward the packets to an egress port of the switch toward the next hop, which may be another switch or a destination component (e.g., destination computer), for example.

Sur-reply, 2-3;Ex. 1005, 3:11-24

The straightforward approach of routing all traffic from network nodes to the central controller introduces some significant drawbacks, such as increased end-to-end traffic delays between the client and the server; overflowing the controller capability to perform other networking functions; and a single point of failure for the re-routed traffic.

Therefore, it would be advantageous to provide a solution that overcomes the deficiencies noted above and allow efficient DPI in SDNs.

Sur-reply, 2-3;Ex. 1001, 2:1-10

Lin

components for security. Reply, 6-7. To the extent that Petitioner is attempting to construe "controller" as "one component" for performing DPI and controlling a network, the Board should reject such a construction as untimely because it was not raised in the Petition. *SAS Institute Inc. v. Iancu*, 138 S.Ct. 1348, 1351 (2018). Further, as discussed below, Swenson does not disclose DPI, let alone in a controller with conventional components.

Sur-reply, 6.

Swenson

Further, Swenson teaches the use of a controller configured to perform DPI. EX1004, ¶80. Swenson discloses that, when its system detects a HTTP packet flow matching a particular signature, "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." EX1007, ¶[0059]; *see id.*, ¶[0060]. "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." EX1007, ¶[0059] (emphasis added); *see id.*, ¶[0060] (stating that the "controller 140 ingests the network flow for inspection"), Figures 1, 4A-4B. A POSA would have known that a "flow" is a series of packets having a specific signature. EX1004, ¶80. As such, it would have been obvious to a POSA that Swenson's reference to "deep flow inspection" refers to performing DPI on one or more packets in a flow. *Id.*

Pet at 19; Sur-reply, 7-11.

Swenson

57. A POSA would understand that Lin discloses deep ***packet*** inspection (DPI) performed at a dedicated security service, whereas Swenson discloses deep ***flow*** inspection (DFI) performed in the flow analyzer of a network controller. Thus, Lin and Swenson disclose ***different*** techniques, which a POSA would understand from the use of different terms of art in the two references.

Sur-reply, 7-11;
Ex 2025 [Gomez], ¶57.

Swenson

58. Furthermore, a POSA would understand that deep ***flow*** inspection is not the same as deep ***packet*** inspection. **Deep Packet Inspection** (or "**DPI**") is well-known technique used for computer network packet filtering that may examine a packet payload (i.e., the part of the packet that contains content relevant to an application, such as an email message or audio/video content) and in some instances the packet header, as a packet is transmitted through an inspection point (e.g., a node, controller, or security device). Networks use DPI for a number of purposes

blocking malware or illegal content). DPI devices are capable of looking at Layer 2 and beyond Layer 3 of the OSI model, and can even analyze up to Layer 7. DPI thus offers control of network traffic that goes beyond classification of packets based on header information and can be used in enterprise security to detect various attacks, including buffer overflow attacks or data leaks.

Sur-reply, 7-11;
Ex 2025 [Gomez], ¶¶ 57-58.

Swenson

59. DPI also presents certain unique challenges. Because it is a thorough packet inspection technique, DPI requires significant processing power and may struggle with high-bandwidth networks or encrypted traffic. Further, DPI tools may have limited visibility into the overall traffic patterns of networks in which less traffic flows through core switches (e.g., in modern distributed network environments). In such scenarios, less granular metadata analysis techniques may be deployed to overcome certain limitations of DPI by giving a more comprehensive view of the entire network, without being affected by limitations that inhere in DPI.

Sur-reply, 7-11;
Ex 2025 [Gomez], ¶59.

Swenson

60. On the other hand, **Deep Flow Inspection** (DFI), is a less well-known technique that—in general—refers to a type of network monitoring focused on the flow of data within the network rather than the inspection of individual packets. Flows convey information that is relevant to different aspects of network monitoring than the information conveyed by individual packets. In particular, inspection of flows conveys information about the state of the network that is not conveyed by individual packets.

Sur-reply, 7-11;
Ex 2025 [Gomez], ¶60.

Swenson

Mr. Gomez also testified that: (i) DFI is used to inspect flows (rather than individual packets) to glean network information (including IP addresses, ports, protocol types, traffic metrics, and state information) that is not conveyed by or ascertainable by inspecting individual packets, (ii) the two techniques have different capabilities and technical challenges, and (iii) Swenson's disclosure of DFI for among other things "collection of statistic information" and bandwidth monitoring reinforced a POSITA's understanding that DFI was used to inspect flows, not individual packets. *Id.* ¶60-65.

Sur-reply, 7-11;
Ex 2025 [Gomez], ¶¶60-65.

DPI & DFI: a Malicious Behavior Detection Method Combining Deep Packet Inspection and Deep Flow Inspection

Yu-tong Guo[&], Yang Gao[&], Yan Wang*, Meng-yuan Qin, Yu-jie Pu, Zeng Wang, Dan-dan Liu, Xiang-jun Chen, Tian-feng Gao, Ting-ting Lv, Zhong-chuan Fu

Department of Computer Science and Technology, Harbin Institue of Technology, Harbin 150001, P.R. China

Abstract

A malicious behavior detection approach which combines both the DPI (Deep Packet Inspection) and DFI (Deep Flow Inspection) is proposed, namely DPI & DFI. For the DPI & DFI method an outlier data mining method is employed. The fine-grained DPI is suitable for plaintext traffic, while DFI is a complementary for encrypted or emerging traffic. The collaborative detection approach includes three phases: DPI detection, DFI detection & comparison, and feedback. In present work, the C4.5 data-mining decision tree is adopted as classifier. The KDD Cup'99 benchmark is used and representative attack categories such as Probing, DOS, R2L (Remote to User) and U2R (User to Root) are evaluated. In-depth analysis demonstrates that the U2R and R2L attack categories lead to lower detection rate, and in particular the attack types contribute most are put forward. In future work, some other types of classifiers suitable to R2L and U2R attack categories should be investigated.

Sur-reply, 7-11; Ex 2025, ¶22;Ex1016, 106; Ex2027

Swenson

2. The Proposed Method

2.1. Framework

The proposed DPI & DFI collaborative detection approach is depicted in Fig. 1. It includes three phases: Phase I DPI detection, phase II DFI detection & comparison, phase III feedback and DPI restarts.

Phase I. DPI detection.

In this phase, A fine-grained packet level inspection, e.g. DPI, is conducted. By resolving the protocol headers, fingerprints of application-layer content of network packets are recognized, e.g. the IP address, port number, the name of application protocol and etc. The DPI is effective for unencrypted traffic and known protocols. After that a result is accquired.

Phase II. DFI detection & Comparison.

DPI is not suitable for the encrypted traffic and unknown protocols, besides the high false-positive detection rate is unfordable. Accordingly, in this work it is coupled with a coarse-grained DFI engine and a data-mining analysis is employed. In this work, an outlier data mining method is adopted for deep flow inspection. The data-mining includes the following processes, e.g. mapping symbolic attributes to numeric attributes, dataset fragmentation, feature reduction, and data-mining, etc. In fact, the coarse-grained DFI and the data minging classifier are not only effective for encrypted traffic and unknown protocols, but also aiming at false-positive detections of DPI [4].

Phase III. Feedback

When an encrypted traffic and an unknown protocol is resolved, the detection result of DFI is the final result. Otherwise, a comparison is made for DFI and DPI deteciton. When the fingerprints of DFI detection is the same with that of the DPI, the final result is accauired. Otherwise, a false-positive sample is signaled and a feedback is forwarded to the Phase I and the DPI is restarted.

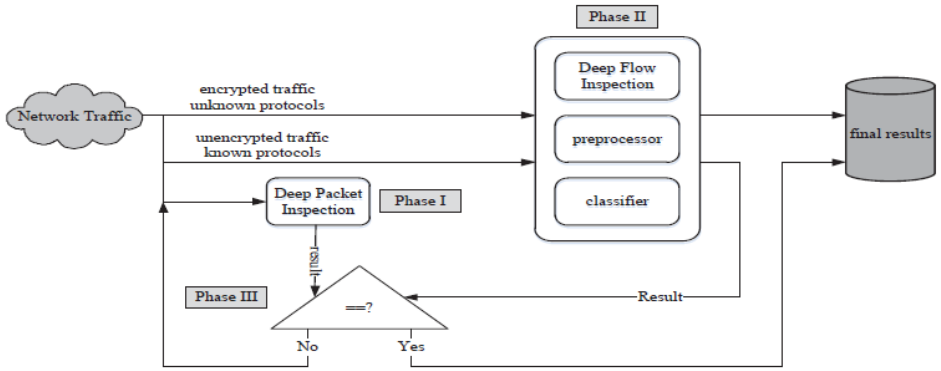


Fig. 1. The overall framework of DPI & DFI

Sur-reply, 7-11; Ex 2025, ¶22;Ex1016, 106; Ex2027

Swenson

2. The Proposed Method

2.1. Framework

The proposed DPI & DFI c
DPI detection, phase II DFI de
Phase I. DPI detection.
In this phase, A fine-graine
fingerprints of application-laye
name of application protocol a
result is accquired.
Phase II. DFI detection & C
DPI is not suitable for the e
is unfordable. Accordingly, in t
employed. In this work, an o
includes the following proces
feature reduction, and data-mining, etc. In fact, the coarse-grained DFI and the data minging classifier are not only effective for encrypted traffic and unknown protocols, but also aiming at false-positive detections of DPI [4].

Phase III. Feedback

When an encrypted traffic and an unknown protocol is resolved, the detection result of DFI is the final result. Otherwise, a comparison is made for DFI and DPI deteciton. When the fingerprints of DFI detection is the same with that of the DPI, the final result is accauired. Otherwise, a false-positive sample is signaled and a feedback is forwarded to the Phase I and the DPI is restarted.

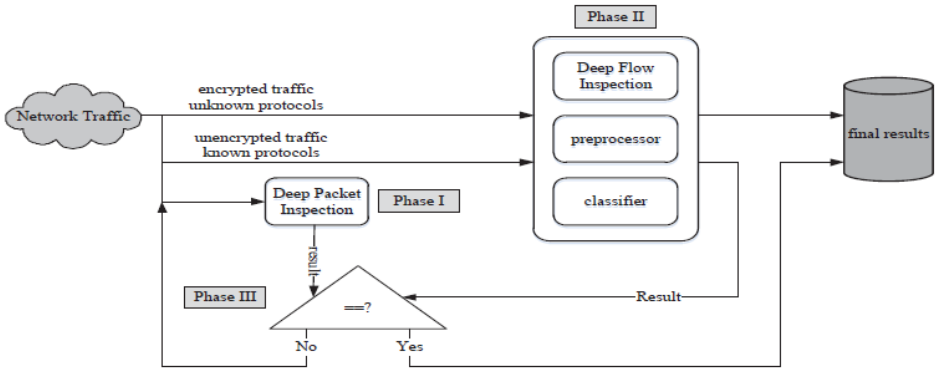


Fig. 1. The overall framework of DPI & DFI

Phase I. DPI detection.

In this phase, A fine-grained packet level inspection, e.g. DPI, is conducted. By resolving the protocol headers, fingerprints of application-layer content of network packets are recognized, e.g. the IP address, port number, the name of application protocol and etc. The DPI is effective for unencrypted traffic and known protocols. After that a result is accquired.

Phase II. DFI detection & Comparison.

DPI is not suitable for the encrypted traffic and unknown protocols, besides the high false-positive detection rate is unfordable. Accordingly, in this work it is coupled with a coarse-grained DFI engine and a data-mining analysis is employed. In this work, an outlier data mining method is adopted for deep flow inspection. The data-mining

Sur-reply, 7-11; Ex 2025, ¶22;Ex1016, 106; Ex2027

Swenson

Mr. Gomez's testimony is thus supported by: (i) the different terms of art used for the two techniques, (ii) Swenson's disclosure of DFI functionalities that are not performed with DPI, (iii) Lin's disclosure of outsourcing DPI to a dedicated security service, (iv) extrinsic evidence distinguishing between DPI and DFI, and (iv) his own experience.

In contrast, Petitioner asserts (incorrectly and transparently oversimplifying things) that a "flow is a series of packets," and Swenson's DFI includes inspection of a HTTP request/response payload. Reply, 7-11. The Board should reject this assertion.

Sur-reply, 7-11; Ex 2025, ¶¶ 57-69;Ex1016, 106; Ex2027

Lin & Swenson

> In short, Swenson does not disclose deep packet inspection, let alone a controller that performs, is capable of performing, or is capable of controlling deep packet inspection. Thus, even if a POSA were motivated to combine Swenson with Lin, the resulting combination would not practice the claimed controller.

POR, 32

Lin & Swenson

> The crux of Petitioner's argument is that a POSITA would have found it obvious to implement security processing in Lin's controller in light of Swenson because ***(a)*** Lin and Swenson share generic similarities (e.g., a central controller, packet routing, and different forms of data inspection) and ***(b)*** determining "which packets to redirect … increases the efficient operation of a computer network." Reply, 11 (citing Paper 1 ("Petition"), 20-25). There are several problems with this argument.

Sur-reply, 11-16

Lin & Swenson

[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. Optimization 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 network state as well as historical traffic data in its decision for

Sur-reply, 11-16; Ex 1007 [Swenson] ¶28

Lin & Swenson

[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.

Sur-reply, 11-16; Ex 1007 [Swenson] ¶39

Lin & Swenson

59. DPI also presents certain unique challenges. Because it is a thorough packet inspection technique, DPI requires significant processing power and may struggle with high-bandwidth networks or encrypted traffic. Further, DPI tools may have limited visibility into the overall traffic patterns of networks in which less traffic flows through core switches (e.g., in modern distributed network environments). In such scenarios, less granular metadata analysis techniques may be deployed to overcome certain limitations of DPI by giving a more comprehensive view of the entire network, without being affected by limitations that inhere in DPI.

Sur-reply, 11-16; Ex 2025, ¶59

Shieh

The ID found that Petitioner failed to explain "how identification of 'TCP FIN or TCP RST packets' corresponds to the teachings surrounding filtering rules in Shieh;" "Shieh does not appear to teach what happens to a packet [when] a TCP FIN or TCP RST packet is identified;" and "Shieh discloses that packet destinations are determined by the bypass flag—which is separate from the TCP FIN or TCP RST packet." ID, 19. The Reply does not address these findings; thus, the Board should reject Ground 2.

Sur-reply, 20

Shieh

[0030] The configuration information may be stored in a memory or storage device of a network access device. In one embodiment, such configuration information may be stored as part of a flow table or session table as shown in FIG. **5**. Referring to FIG. **5**, a bypass flag **501** may be received from a security device indicating that the security device no longer wishes to receive further packets of the same session for security inspection. In addition, a security device may register certain notification events **502** with a network access device, such that when the network access device detects such events, it will notify the security device. Further, a set of one or more filtering rules **503** may be received from an administrator to filter and send only certain types of packets to a security device for inspection.

Sur-reply, 20; Ex 1006 [Shieh], ¶30

Shieh

[0036] During the bypass phase, the I/O function may notify the security-processing function if there are special events in the packet stream. These events could be receipt of TCP FIN or TCP RST packets, or not receiving any packets of the connection within a time threshold. The notification from I/O functions to security processing functions could help to clean up the state in the security-processing nodes.

Sur-reply, 20; Ex 1006 [Shieh], ¶ 36

Shieh

[0037] FIG. **2**B is a processing flow diagram illustrating a process of security inspection according to one embodiment of the invention. Referring to FIG. **2**B, as an example, network switch **272** may represent any of network access devices **204**A-**204**C and security device **273** may represents any of security processing devices **211**A-**211**B as described above with respect to FIG. **2**A. When device **272** receives a packet from a source node **271** via transaction **281**, device **272** may determine whether the packet should be forwarded to security device **273**. For example, device **272** may look up in its session table such as the one as shown in FIG. **5** to determine whether a bypass flag has been set to a predetermined value. If the bypass flag matches the predetermined value, the packet is forwarded to security device **273** via path **282**; otherwise, the packet is routed to destination node **274**. Alternatively, if there is no entry in the session table corresponding to the current session, the packet will also be transmitted to security device **273**. After network device **272** receives a response from security device **273** via path **283**, dependent upon the response, the packet may then be routed to destination node **274** via path **284**. These processes may continue until a notification is received from security device **273** via path **285** indicating that it no longer wishes to receive further packets of the same session for inspection, such that subsequent packets will be directly routed to destination node **274** via path **286** without routing to security device **273**. If there are certain events that have been registered from security device **273**, network device **272** may notify security device **274** via path **287** upon detecting the registered events.

Sur-reply, 20; Ex 1006 [Shieh], ¶37

Lin does not disclose the "packet applicable criterion" limitation

TABLE 3

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

Table 3 of Lin (Annotated)

TABLE 3

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

Table 3 of Lin (Annotated)

Sur-reply, 16-19.