

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

CISCO SYSTEMS INC.,

Petitioner

U.S. Patent No. 7,623,518

**DECLARATION OF DR. MARK E. CROVELLA
UNDER 37 C.F.R. § 1.68 IN SUPPORT OF PETITION
FOR *INTER PARTES* REVIEW**

Table of Contents

I.	INTRODUCTION	4
II.	QUALIFICATIONS	5
III.	LEVEL OF ORDINARY SKILL IN THE ART	8
IV.	RELEVANT LEGAL STANDARDS	9
V.	THE '518 PATENT	11
A.	Summary of the '518 patent	11
B.	Prosecution History of the '518 patent	13
VI.	CLAIM CONSTRUCTION	16
A.	“at least one of” and “each of” (Claim 1)	16
VII.	DETAILED UNPATENTABILITY ANALYSIS	17
A.	Ground 1: Claims 1 and 3 are Obvious over Gai in combination with Yip and Kwan	18
1.	Summary of Gai	18
2.	Summary of Yip	22
3.	Summary of Kwan	23
4.	Reasons to Combine Yip with Gai	26
5.	Reasons to Combine Kwan with Gai	30
6.	Claim 1	37
7.	Claim 3	61
B.	Ground 2: Claims 15, 17, 19-21 are Obvious over Gai in combination with Yip, Kwan, and Georgiou	62
1.	Summary of Georgiou	62
2.	Reasons to Combine Georgiou with Gai	64
3.	Claim 15	68
4.	Claim 17	106

5.	Claim 19	107
6.	Claim 20	112
7.	Claim 21	113
VIII.	DECLARATION.....	113

I, Mark E. Crovella, Ph.D., do hereby declare as follows:

I. INTRODUCTION

1. I am making this Declaration at the request of Cisco Systems, Inc. in the matter of the *Inter Partes* Review of U.S. 7,623,518 (“the ’518 patent”).

2. I am being compensated for my work in this matter at my standard hourly rate. I am also being reimbursed for reasonable and customary expenses associated with my work and testimony in this proceeding. My compensation is not contingent on the outcome of this matter or the specifics of my testimony.

3. I have been asked to provide my opinions regarding whether the subject matter of claims 1, 3, 15, 17, and 19-21 (“the Challenged Claims”) of the ’518 patent would have been obvious to a person having ordinary skill in the art (“POSITA”) at the time of the alleged invention, in light of the prior art. I have reviewed the Petition and agree that the Challenged Claims would have been obvious to a POSITA.

4. In the preparation of this Declaration, I have studied:

- (1) The ’518 patent, Ex.1001;
- (2) Prosecution History of the ’518 patent, Ex.1002;
- (3) U.S. Patent No. 6,651,096 to Gai *et al.* (“Gai”), Ex.1006;
- (4) U.S. Patent No. 7,523,485 to Kwan (“Kwan”), Ex.1007;
- (5) U.S. Patent No. 7,003,597 to Georgiou *et al.* (“Georgiou”), Ex.1008;
- (6) U.S. Patent No. 6,105,086, Ex.1009;

Ex.1003

- (7) U.S. Patent Publication No. 2001/0012294A1, Ex.1010;
- (8) U.S. Patent No. 5,720,032, Ex.1011;
- (9) U.S. Patent No. 5,797,033, Ex.1012;
- (10) U.S. Patent No. 8,751,647, Ex.1013 (“Yip”);
- (11) U.S. Patent No. 7,343,485, Ex.1014;
- (12) U.S. Patent No. 7,343,485, Ex.1015;
- (13) U.S. Patent Publication No. 2005/0055570A1, Ex.1016;
- (14) Intellinet User Manual, Ex.1017;
- (15) Netgear WG602 Users Guide, Ex.1018; and
- (16) U.S. Patent No. 7,343,485, Ex.1019.

5. In forming the opinions expressed below, I have considered: the documents listed above; the relevant legal standards, including the standard for obviousness; and my own knowledge and experience based upon my work in the field of wireless communications as described below, as well as any additional materials cited herein.

6. Unless otherwise noted, all **bold** emphasis in any quoted material has been added and claim language is in *italics*.

II. QUALIFICATIONS

7. The details of my background and education, and a listing of all publications that I have authored, are provided in my *Curriculum Vitae*, a copy of which is submitted as Ex.1004. The following is a brief summary of my relevant qualifications and professional experience.

8. I am a Professor in the Department of Computer Science at Boston University. A brief summary of my background is as follows: I earned an undergraduate degree in Biology from Cornell University, which I received in 1982. I received a master's degree in Computer Science from the University of Buffalo in 1989. I received a Ph.D. in Computer Science from the University of Rochester in 1994. The subject of my Ph.D. thesis was "Performance Prediction and Tuning of Parallel Programs."

9. From 1982 to 1984, I worked as a computer programmer for the State of Colorado. From 1984 to 1994, I was employed at Calspan Corporation, a research and development firm in Buffalo, NY, where I rose to the level of Senior Computer Scientist. My work at Calspan focused on development of experimental software and large-scale simulation software in support of contracts between Calspan and the U.S. Department of Defense.

10. In 1994, I joined the faculty of Boston University, initially as an Assistant Professor of Computer Science. I was promoted to the rank of Associate Professor in 2000 and became a full Professor in 2006. From 2013 to 2018 I served as Chair of the Department of Computer Science.

11. I have been pursuing research in the area of computer networking since 1994. I have conducted research in a variety of areas related to the Internet and the World Wide Web. Among other areas, I have studied the efficient design of Web

servers and content distribution systems; I have studied the statistical properties of Internet traffic; and I have made extensive measurements of Internet infrastructure and the behavior of Internet protocols. From 2007 to 2009, I was the Chair of ACM SIGCOMM (the Special Interest Group in Computer Communication), the main professional organization for scientists in the field of computer networking.

12. In addition, I have been pursuing research in the area of computer security since around 2000. I have published extensively on techniques for intrusion detection, security analysis, and the identification of malicious behavior in networks and computer systems. My work in the area of computer and network security is highly cited.

13. I regularly teach courses that cover aspects of computer systems security, such as *Operating Systems*, *Introduction to Computer Systems*, and *Computer Networking*. In these courses we cover topics such as authentication, secure storage of data, and methods for establishing secure connections between devices. *Computing Networking* includes coverage of the addressing methods used in networks, the architecture and operation of the Domain Name System (DNS), and methods for secure communication over networks, including principles of access control and the role of access control lists (ACLs).

14. I am co-author of *Internet Measurement: Infrastructure, Traffic, and Applications* (Wiley Press, 2006) and I am the author of over two hundred research

papers. According to Google Scholar, my work has been cited by over 30,000 other publications. Also, according to Google Scholar, as of March 2014, only nine other researchers worldwide are more cited in the area of computer systems. I hold ten U.S. Patents derived from my Internet related research. I am a past editor of principal journals in the field of networking: *Computer Communication Review*, *IEEE/ACM Transactions on Networking*, *Computer Networks*, and *IEEE Transactions on Computers*.

III. LEVEL OF ORDINARY SKILL IN THE ART

15. I understand there are multiple factors relevant to determining the level of ordinary skill in the pertinent art, including (1) the levels of education and experience of persons working in the field at the time of the invention; (2) the sophistication of the technology; (3) the types of problems encountered in the field; and (4) the prior art solutions to those problems.

16. I am familiar with the computer network art that is pertinent to the '518 patent. A POSITA in the field of the '518 patent, as of its earliest claimed priority date of April 8, 2004, would have been someone knowledgeable with the computer network techniques available at the time. Such a POSITA would include someone who had a bachelor's degree in computer science, computer engineering, electrical engineering, or an equivalent, and two years of experience working with

computer networks, including network security. Lack of work experience can be remedied by additional education, and vice versa.

17. For purposes of this Declaration, in general, and unless otherwise noted, my statements and opinions, such as those regarding my own experience and what a POSITA would have understood or known generally (and specifically related to the references I consulted herein), reflect the knowledge that existed in the relevant field as of the priority date of the '518 patent.

IV. RELEVANT LEGAL STANDARDS

18. I am not an attorney. In preparing and expressing my opinions and considering the subject matter of the '518 patent, I am relying on certain legal principles that counsel has explained to me.

19. I understand that prior art to the '518 patent includes patents and printed publications that predate the priority date of the '518 patent. For purposes of this Declaration, I am applying the earliest claimed date of April 8, 2004, as the priority date of the '518 patent.

20. I have been informed by Cisco's counsel that a claimed invention is unpatentable under 35 U.S.C. § 103 if the differences between the claimed invention and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a POSITA. I have also been informed by Cisco's counsel that the obviousness analysis considers factual

inquiries, including the level of ordinary skill in the art, the scope and content of the prior art, and the differences between the prior art and the claimed subject matter.

21. I have been further informed by Cisco's counsel that there are several recognized rationales for combining references or modifying a reference to show obviousness. These rationales include: (a) combining prior art elements according to known methods to yield predictable results; (b) simple substitution of one known element for another to obtain predictable results; (c) use of a known technique to improve a similar device (method, or product) in the same way; (d) applying a known technique to a known device (method, or product) ready for improvement to yield predictable results; (e) choosing from a finite number of identified, predictable solutions, with a reasonable expectation of success; and (f) some teaching, suggestion, or motivation in the prior art that would have led a POSITA to modify the prior art or to combine prior art teachings to arrive at the claimed invention.

22. Also, I have been informed and understand that obviousness does not require physical combination/bodily incorporation, but rather consideration of what the combined teachings would have suggested to a POSITA at the time of the alleged invention.

V. THE '518 PATENT

A. Summary of the '518 patent

23. The '518 patent, titled "Dynamic Access Control Lists," is based on an application filed on April 8, 2004. Ex.1001, Cover.

24. The '518 patent generally discloses "a method [for] control[ing] access of a user to a network including a plurality of hosts coupled together through a network switch." Ex.1001, 3:20-22. The '518 patent's method uses an enhanced access control list that may contain "data related to at least one of user names, DNS names, domain names, and physical addresses." Ex.1001, 3:22-29. Data in the enhanced access control list is used to generate a "dynamic access control list." Ex.1001, 3:22-29.

25. In one example, the '518 patent discloses that the dynamic access control list is generated by converting a MAC address (e.g., a physical address in the enhanced access control list) into an IP address. Ex.1001, 7:51-54; Ex,1001, 1:7-8 ("...each host is uniquely identified by a physical address which is designated a media access control (MAC) address..."). The '518 patent's converting is based on correlating an unknown IP address with a known MAC address. For example, the '518 patent discloses that when a new IP address is identified, the switch monitors dynamic host configuration protocol ("DHCP") packets to determine if the new IP address is assigned to a known MAC address, which is used "to generate a

Ex.1003

corresponding record in the dynamic access control list 104.” Ex.1001, 7:54-67. In another examples, conversion may include employing other techniques, such as address resolution protocol (“ARP”), domain name server (“DNS”) monitoring, among others. Ex.1001, 8:1-43.

26. The ’518 patent includes twenty-two claims. Independent claims 1 and 15 are reproduced below as illustrative of the challenged claims.

1. A method of developing an access control list, comprising:
developing an enhanced access control list including data related to at least one of user names, DNS names, Windows domain names, and physical addresses;
converting, in a network switch, at least one of:
user names into corresponding IP and physical addresses according to data in the enhanced access control list, wherein converting user names into corresponding IP and physical addresses according to data in the enhanced access control list includes:
detecting login packets being communicated over the network;
determining a MAC address from the login packets;
detecting server message block login packets being communicated over the network;
determining an IP address from the server message block login packets; and
developing records in the access control list using the obtained IP address for the respective user name;
DNS names into corresponding IP addresses according to data in the enhanced access control list; and
physical addresses into IP addresses according to data in the enhanced access control list; and
developing, in the network switch, the access control list from each of the operations of converting.

15. A network switching circuit, comprising:
a forwarding circuit operable to detect specific received packets and to provide the specific packets on a processor port, and further operable to receive packets on one of a plurality of ports including the processor port and to forward each received packet to a port corresponding to a destination address contained in the packet subject to access restrictions contained in a dynamic access control list;
a memory circuit coupled to the forwarding circuit, the memory circuit operable to store packets and operable to store an enhanced access control list and a dynamic access control list; and
a processor coupled to the forwarding circuit and to the memory circuit, the processor operable to define the specific packets detected by the forwarding circuit and operable to process the specific packets stored in the memory circuit using the enhanced access control list to generate the dynamic access control list and store the dynamic access control list in the memory circuit, and further operable to provide the specific packets to the processor port of the forwarding circuit after processing the packets.

B. Prosecution History of the '518 patent

27. The '518 patent was filed on April 8, 2004. During prosecution, in an Office Action dated March 20, 2008, claims 1-17 of the '518 patent were rejected over Droms et al. (US Patent No. 7,143,435) in combination with Donaldson (US Pat: 7,249,175) and claims 18-25 were rejected over Droms et al. (US Patent No. 7,143,435) in combination with Fan et al (US Pat: 6,219,706). Ex.1002, 110-120.

28. In response, Applicant did not amend the claims but argued that the cited art failed to teach “*developing an enhanced access control list including data related to at least one of user names, DNS names, Windows domain names, and physical addresses; converting at least one of, user names into corresponding IP and physical addresses according to data in the enhanced access control list,*” as recited in claim 1. Ex.1002, 88-105. Further, Applicant argued that Droms does not teach (1) a memory circuit in a network switching circuit that is operable to store an enhanced access control list and a dynamic access control list, (2) a network switching circuit that “*processes specific packets stored in the memory circuit using the enhanced access control list to generate the dynamic access control list*” recited in claim 18, or (3) “*a processor...operable to process the specific packets stored in the memory circuit using the enhanced access control list to generate the dynamic access control list and store the dynamic access control list in the memory circuit,*” as recited in claim 22. Ex.1002, 88-105.

29. In a subsequent Office Actions dated October 17, 2008, and February 4, 2009, the Examiner maintained the rejections of claims 1-3, 5-11, 13-17 over Droms in view of Green et al (US Pub:2004/0193918) but indicated that claims 4, 10, and 18-25 were allowable. In response, the Applicant amended claim 1 to recite the features of allowable claim 4 (now canceled).

1. (Currently amended) A method of developing an access control list, comprising:

developing an enhanced access control list including data related to at least one of user names, DNS names, Windows domain names, and physical addresses;

converting, via a network switch, at least one of: [[.]]

user names into corresponding IP and physical addresses according to data in the enhanced access control list, wherein converting user names into corresponding IP and physical addresses according to data in the enhanced access control list includes:

detecting login packets being communicated over the network;

determining a MAC address from the login packets;

detecting server message block login packets being communicated over the network;

determining an IP address from the server message block login packets; and

developing records in the access control list using the obtained IP address for the respective user name;

DNS names into corresponding IP addresses according to data in the enhanced access control list; and

physical addresses into IP addresses according to data in the enhanced access control list; and

developing, via the network switch, the access control list from each of the operations of converting.

Ex.1002, 21-29. The Examiner then withdrew the rejections and allowed the claims.

Ex.1002, 8-13.

30. Notably, as I demonstrate below, there is nothing novel about the Challenged Claims of the '518 patent because the claimed features were known in the art, and it would have been obvious to combine the relevant prior art teachings as claimed.

VI. CLAIM CONSTRUCTION

31. It is my understanding that in order to properly evaluate the '518 patent, the terms of the claims must first be interpreted. It is my understanding that for the purposes of this *inter partes* review, the claims are to be construed under the so-called *Phillips* standard, under which claim terms are given their ordinary and customary meaning as would have been understood by a POSITA in light of the specification and prosecution history, unless the inventor has set forth a special meaning for a term. I have also been informed that claim terms only need to be construed to the extent necessary to resolve the obviousness inquiry.

32. I have applied the so-called *Phillips* standard in the unpatentability analysis. I have reviewed the entirety of the '518 patent, as well as its prosecution history. It is my opinion that for purposes of applying the prior art presented herein to evaluate the patentability of the claims most of the claim terms do not require express construction. Below I identify how the claim term “*at least one of*” is to be construed for the purpose of this proceeding.

A. “*at least one of*” and “*each of*” (Claim 1)

33. Claim 1 in step [1.1] recites “*developing an enhanced access control list including data related to **at least one of** user names, DNS names, Windows domain names, and physical addresses*” and in step [1.2.1] further recites “*converting, in a network switch, **at least one of**... user names...DNS names...[and]*

physical addresses...” in the enhanced access control list into IP addresses.

Additionally, the claim in step [1.3] recites “*developing, in the network switch, the access control list from **each of** the operations of converting.*”

34. In the context of co-pending District Court litigation, Patent Owner communicated to Petitioner that it “disagree[s]” that the construction for the term “*at least one of*” requires “an enhanced ACL that includes data related to all four of the following: user names, DNS names, Windows domain names, and physical addresses.” Ex.1019. Patent Owner further indicated that, along these lines, it will “provide its proposed claim constructions pursuant to the schedule set forth in the Docket Control Order.” Ex.1019. Accordingly, I have applied Patent Owner’s apparent district court interpretation of the claim term “*at least one of*” (in steps [1.1] and [1.2.1]) as requiring only one of the listed items. Additionally, because the phrase “*each of*” (in step [1.3]) relates back to the “*at least one of*” converting operations (in step [1.2.1]), I have applied Patent Owner’s apparent district court interpretation to mean developing from the at least one of the listed items of converting.

VII. DETAILED UNPATENTABILITY ANALYSIS

35. I have been asked to provide my opinion as to whether the Challenged Claims of the ’518 patent would have been obvious in view of the prior art. In rendering my opinion, I have considered the scope and content of the prior art and any potential differences between the claimed subject matter and the prior art. I

Ex.1003

have also considered the level of ordinary skill and the knowledge of a POSITA as of April 8, 2004, the filing date of the '518 patent.

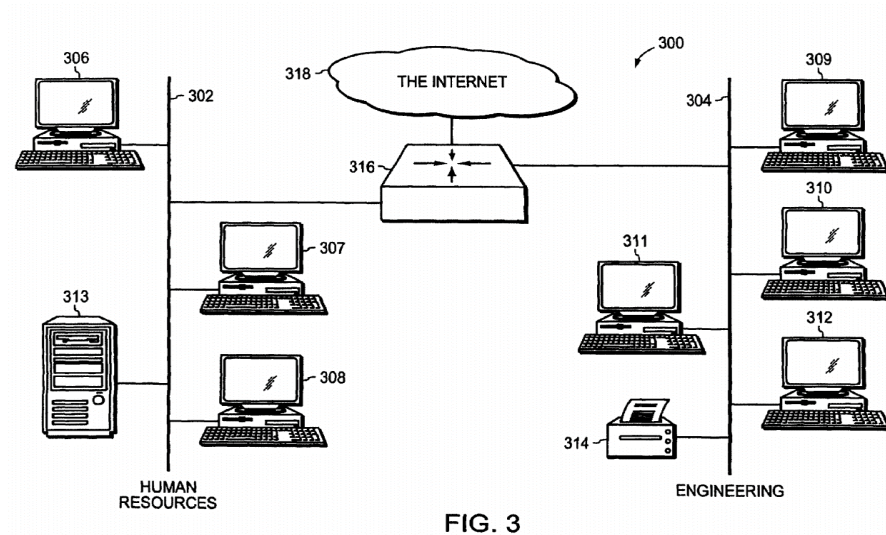
36. As I discuss below, it is my opinion that the claimed subject matter of the Challenged Claims of the '518 patent would have been obvious in view of the teachings of the identified prior art references as well as the knowledge of a POSITA.

A. Ground 1: Claims 1 and 3 are Obvious over Gai in combination with Yip and Kwan

1. Summary of Gai

37. The primary reference that I rely upon is U.S. 6,651,096 to Gai *et al.* (Ex.1006, "Gai"), which was filed by Cisco Systems, Inc. on April 20, 1999.

38. Gai general discloses a method and apparatus for a packet switched network 300. Ex.1006, 1:14-18, 1:59-2:4, 5:23-34, Figs. 3, 4. Gai's Figure 3, reproduced below, illustrates its packet switched network.



Ex.1006, Fig. 3.

39. Gai's network 300 includes a network device 316, which is a switch, that supports data communication standards such as "Ethernet," "Internet Protocol (IP)," among others. Ex.1006, 1:21-40, 5:3-59. Additional detail of Figure 3's network 300, including the various components, is provided by Gai. *See e.g.*, Ex.1006, 5:4-22, Fig. 3.

40. Gai's network device 316 is illustrated below at Figure 4, with its corresponding description.

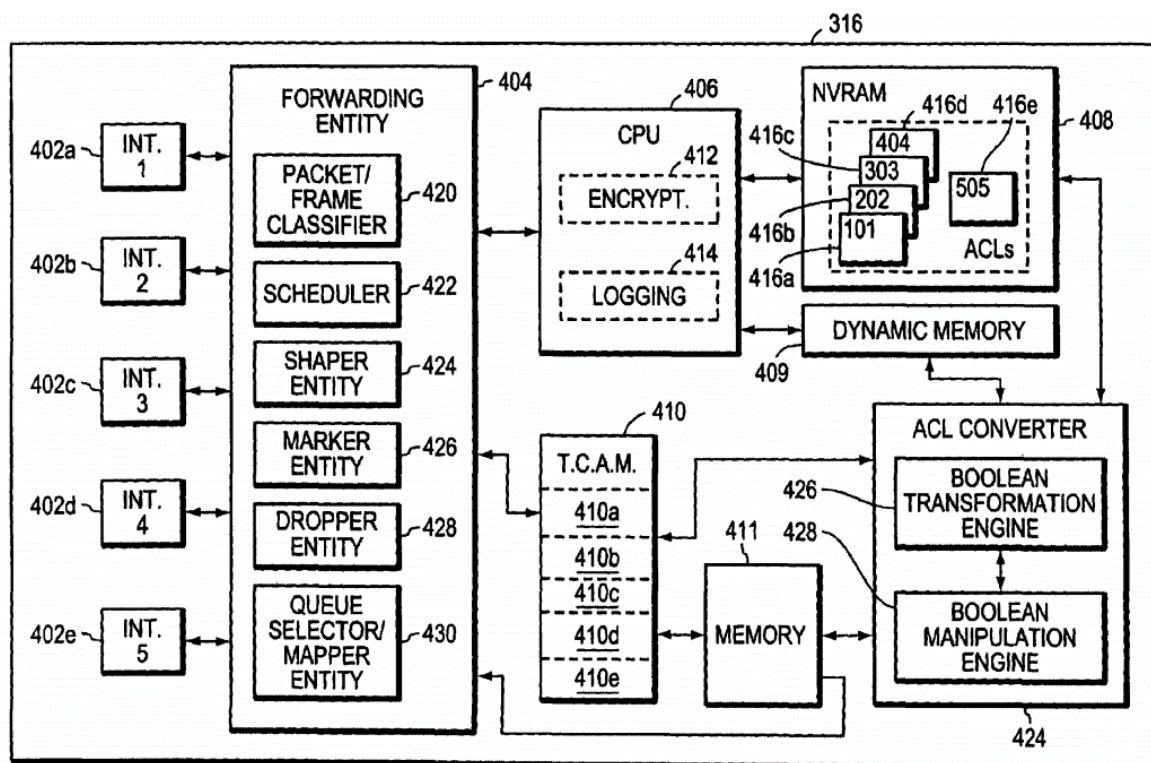


FIG. 4

Ex. 1006, Fig. 4.

FIG. 4 is a partial block diagram of the intermediate network device 316. Device 316 preferably includes a plurality of interfaces 402*a*-402*e* that provide connectivity to the network 300. That is, interfaces 402*a*-402*e* are in communication with LANs 302 and 304 and Internet 318. Each interface 402*a-e*, moreover, may be associated with one or more physical ports (not shown). Device 316 further includes at least one forwarding entity 404, a central processing unit (CPU) 406, non-volatile random access memory (NVRAM) 408, dynamic memory 409 and at least one content addressable or associative memory 410. The associative memory 410 is preferably a ternary content addressable memory (TCAM).

Ex.1006, 5:60-6:7, Fig. 4.

41. Gai's NVRAM 408 includes "one or more text-based access control lists (ACLs) 416a-416e, such as ACLs 101, 202, 303, 404 and 505" that "may be downloaded to device 316 by the network administrator." Ex.1006, 6:13-18, 7:60-66. Exemplary ACLs are provided at Figures 5A-5E, of which Figure 5A is reproduced below for reference.

	516 SOURCE ADDRESS	518 DESTINATION ADDRESS	520 SOURCE PORT	522 DESTINATION PORT	524 PROTOCOL	526 ACTION
502	1.2.3.3	x.x.x.x	x	x	x	PERMIT
504	1.2.3.5	x.x.x.x	80	x	x	PERMIT
506	1.2.3.x	x.x.x.x	x	x	x	DENY
508	2.5.4.x	x.x.x.x	100	x	x	PERMIT AND LOG
510	2.5.3.x	x.x.x.x	x	x	x	DENY AND LOG
512	x.x.x.x	3.5.4.x	x	x	TCP	DENY
514	x.x.x.x	x.x.x.x	x	x	x	DENY

FIG. 5A

Ex. 1006, Fig. 5A.

42. With respect to Figure 5A, Gai discloses that each row "corresponds to an Access Control Entry (ACE) statement, such as ACE statements 502-514, which specify the various criteria for the ACL." Ex.1006, 7:20-27. The ACL also includes "a separate column for source address 516, destination address 518, source port 520, destination port 522 and protocol 524" and action 526. Ex.1006, 7:27-32. Gai

Ex.1003

describes other ACLs including other ACE statements. In one example, “[i]f a packet matches an ACE statement of ACL 416c having a ‘permit’ action, then the packet is encrypted” by CPU 406. Ex.1006, 8:46-49; *see also* Ex.1006, 11:60-12:42.

43. Gai discloses a process of compiling the text-based ACLs into a boolean Binary Decision Diagram (“BDD”) representation. Ex.1006, 6:35-55, 9:22-11:7, Figs. 6, 7. The BDD ACL may be further optimized into a ternary content addressable memory (“TCAM”) ACL. Ex.1006, 6:3-7, 15:2-16:5. Ex.1006, 13:66-16:5, Figs. 9A-9B.

2. Summary of Yip

44. U.S. 8,751,647 to Yip et al. (Ex.1007, “Yip”) was filed on June 30, 2001, and issued on June 10, 2014.

45. Yip generally “relates to the field of network security. In particular, the present invention relates to a network security system that controls access to a network at the edges of the network.” Ex.1013, 1:6-1:10. Yip teaches that it was known in the art to include MAC addresses in the access list.

For example, access lists were developed to combat cyber attacks on the LAN Switch by providing an administrator-controlled list of Internet Protocol (IP) addresses or media access control (MAC) addresses that were authorized to access the network.

Ex.1013, 2:43-47.

3. Summary of Kwan

46. U.S. 7,523,485 to Kwan (Ex.1007, “Kwan”) was filed on July 31, 2003, and claims priority to Provisional application No. 60/472,158, filed on May 21, 2003, and Provisional application No. 60/472,170, filed on May 21, 2003.

47. Kwan generally relates to “a method of providing for enhanced security on a computer network to reduce the risk created by the spoofing of IP addresses.” Ex.1007, 1:15-17. Kwan discloses a network device 200 that provides switching functions. Ex.1007, 3:42-46. Kwan’s network device 200 is illustrated at Figure 2, reproduced below.

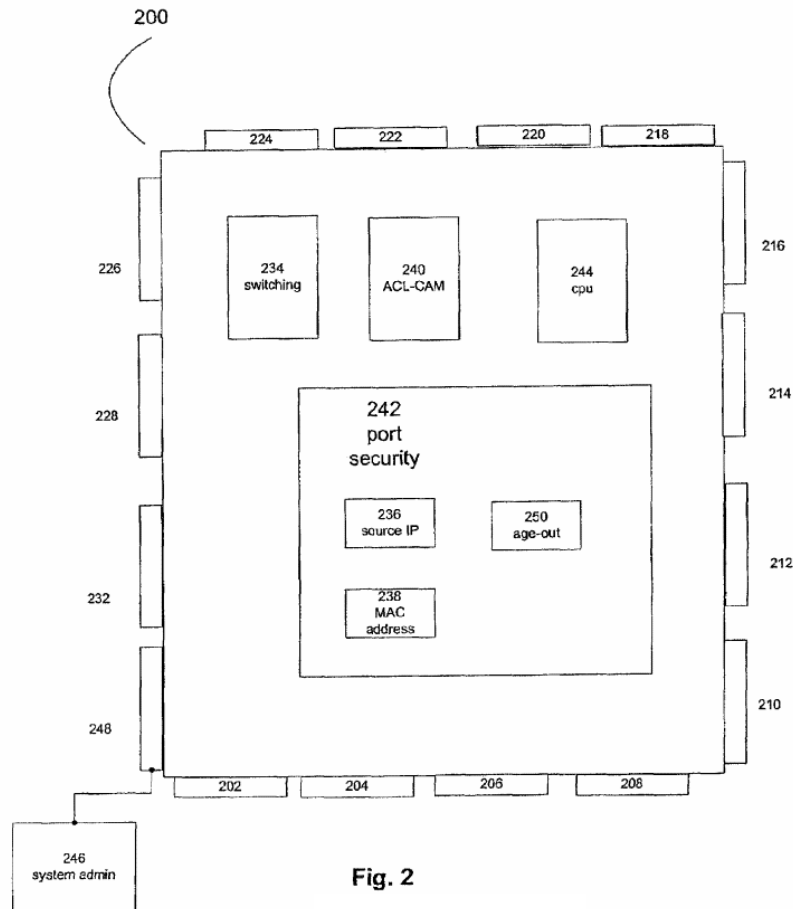


Fig. 2
Ex.1007, Fig. 2

48. Network device 200 includes a plurality of ports 202-232 for receiving and transmitting packets through network device switching 234. Ex.1007, 3:45-58. The network device switching 234 operations were “well known.” Ex.1007, 3:59-60. Kwan discloses that the network device 200 includes an access control list (“ACL”) stored in a content addressable memory (“CAM”). Ex.1007, 4:10-55.

49. Like the ’518 patent, Kwan correlates new IP addresses to known MAC address (referred to as “learning”) so to update the ACL-CAM with new IP addresses. Kwan’s learning—like the converting of the ’518 patent—includes

Ex.1003

employing known protocols such as DHCP and ARP to learn a new IP address based on a MAC address.

For example, the source IP address can be learned by using a reverse address resolution protocol (RARP) which provides for sending out the MAC address on the subnet, and in response the host having the MAC address responds with its IP address. The source IP address detector 236 could also learn the IP address for a host by listening to the DHCP response packet being returned to the host. This response contains the source IP Address for the host. When a DHCP packet is detected, the entry in the table for the MAC address receiving the DHCP packet with a source IP address is cleared and the source IP address provided in the DHCP packet is loaded into the table. This utilization of the DHCP response works well where the port is set to allow for one IP address.

...

Another technique provides for trapping (listening to) ARP requests and ARP reply messages to learn the source IP address and MAC address pairs, and storing the pairs in ACL-CAM for each port.

Ex.1007, 5:38-67.

50. Further, to reduce the risk of possibly learning the wrong IP address, Kwan “allow[s] an [sic] system administrator to seed the IP learning process with the IP address properties that are consistent with the subnet to which the host is coupled.” Ex.1007, 6:14-18. For example, “if the IP Subnet is 10.32.1.0/24, then this information can be used to seed the EP learning process and only match a MAC

Ex.1003

address with [] IP addresses consistent with the subnet 10.32.1.0/24 addresses.”

Ex.1007, 6:18-22. According to Kwan, in some instances, the seed IP address properties “will eliminate false learning of a default” IP address. Ex.1007, 6:22-25. Once learning is successful, the ACL-CAM is updated with this new data so that it may be applied to the port. Ex.1007, 6:28-30; *see also* Ex.1007, 7:60-66 (“...it is important in many applications of the source IP security operations herein, that the table containing the learned source IP addresses be dynamic, such that the table can be updated so that...new learned source IP addresses can be added to the table...”); Ex.1007, 8:43-48. After the ACL has been dynamically updated with the new IP address, the packet is then passed to the destination. Ex.1007, 8:48-51.

4. Reasons to Combine Yip with Gai

51. A POSITA would have been motivated to combine the teachings of Yip with Gai, as discussed below.

a. Analogous Art

52. Gai and Yip are both analogous art to the ’518 patent. As noted above in the prior art summary, like the ’518 patent, Gai and Kwan are in the same field of endeavor of computer networks and address the same problem of using access control lists. Ex.1001, 1:39-46, Abstract; Ex.1006, 1:14-18, 3:17; Ex.1013, 1:5-9, 2:43-47. A POSITA considering Gai would have also considered Yip, which provides additional access control list teachings applicable to Gai.

b. **Motivation to Combine**

53. A POSITA would have been motivated to combine the teachings of Yip with Gai to produce predictable and beneficial results as discussed below.

54. Gai's network device 316 utilizes text-based access control list with various criteria initially set up by an administrator for processing packets. Ex.1006, 7:16-29 (“[A] network administrator creates one or more access control lists in a conventional manner...The columns of the ACL represent the specific criteria with which network messages are compared.”); *see also* Ex.1006, 2:46-57, 3:26-30, 5:3-21, 6:13-17, 21:1-7, Fig. 3, 5A-5E.

55. Gai's ACLs are designed to provide “security...[and] implement additional features.” Ex.1006, 8:3-6. It was generally known in the art for ACLs to include MAC addresses to provide additional security and functionality, such allowing only devices with known addresses to communicate on the network. *See e.g.*, Ex.1014, 6:30-32, 8:27-47, 11:57-62 (disclosing a table that includes IP/MAC addresses corresponding to an access control list); Ex.1017, 55 (“If you choose ‘Allowed Listed’, only those clients whose wireless **MAC addresses are in the access control list will be able to connect** to your Access Point. When ‘Deny Listed’ is selected, these wireless clients on the list will not be able to connect to the Access Point.”); Ex.1018, 13 (“**Access Control List...** Click Allow or Block to enable **MAC address** filtering. Click Disable to disable Access Control

feature...Enter the MAC address for each station to be managed, in the MAC Address field and click Add.”).

56. Consistent with general knowledge in the art, Yip teaches that it was known in the art for an administrator to include MAC addresses in ACLs. Ex.1013, 2:43-47 (“For example, access lists were developed to combat cyber-attacks on the LAN switch by providing an administrator-controlled list of Internet Protocol (IP) addresses or media access control (MAC) addresses that were authorized to access the network.”).

57. It would have been obvious to a POSITA for Gai’s administrator, when creating one or more text-based ACLs, to include MAC addresses of devices authorized to access the network, per Yip. Including the MAC address in Gai’s text-based ACLs would beneficially allow for dynamically generating TCAM ACLs with MAC addresses to thereby restrict or permit access based on MAC address. Such an implementation would improve security because only specific MAC addresses would be permitted to communicate packets over the network. Adding the additional feature of being able to filter packets based on MAC addresses is consistent with Gai’s goal of providing security and implementing additional features with its ACLs. Ex.1006, 8:3-6.

58. The proposed combination merely represents a combination of prior art elements (MAC addresses, per Yip, with Gai’s ACL) according to known methods

Ex.1003

to yield predictable results (facilitate filtering of packets based on MAC addresses). The combination of Yip and Gai is also the use of a known technique (including MAC addresses in an ACL, per Yip) to improve a similar system (Gai's system) in the same way (provide additional security by only permitting known MAC addresses to communicate on the network).

c. Reasonable Expectation of Success

59. The results would have been predictable and there would have been a reasonable expectation of success in the combination given that both Gai and Yip disclose similar network devices that use ACLs. Any modification needed to implement Gai's ACLs to include MAC address would have been conventional and well within the level of skill in the art. *See e.g.*, Ex.1006, 2:58-60 ("To generate an access control list, a network administrator typically defines a sequence of criteria statements using a **conventional** text editor or graphical user interface (GUI).").

60. The reasonable expectation of success is also supported by others that had implemented ACLs with MAC addresses. *See e.g.*, Ex.1014, 6:30-32, 8:27-47, 11:57-62 (disclosing a table that includes IP/MAC addresses corresponding to an access control list); Ex.1017, 55 ("If you choose 'Allowed Listed', only those clients whose wireless MAC addresses are in the access control list will be able to connect to your Access Point. When 'Deny Listed' is selected, these wireless clients on the list will not be able to connect to the Access Point."); Ex.1018, 13 ("Access

Control List... Click Allow or Block to enable MAC address filtering. Click Disable to disable Access Control feature...Enter the MAC address for each station to be managed, in the MAC Address field and click Add.”).

61. The proposed combination of Yip and Gai relies on the teachings, and I do not suggest that physical incorporation of Yip’s elements into Gai is required.

5. Reasons to Combine Kwan with Gai

62. In addition to the above reasons for combining Yip with Gai, a POSITA would have been motivated to combine the teachings in Kwan with Gai (as modified per Yip), as discussed below.

a. Analogous Art

63. Like Gai, Kwan is analogous art to the ’518 patent. As noted above in the prior art summary, Kwan (like Gai) is in the same field of endeavor of computer networks and address the problem of using access control lists. Ex.1001, 3:20-22, Fig. 1; Ex.1006, 1:14-18, 1:59-2:4, 5:23-30, 6:13-7:66, Figs. 3, 4, 5A-5E; Ex.1007, 3:42-4:55, 6:18-25, Fig. 2. A POSITA considering Gai would have also considered Kwan, which provides additional teachings applicable to Gai.

b. Motivation to Combine

64. A POSITA would have been motivated to combine the teachings of Kwan with Gai to produce predictable and beneficial results as discussed below.

65. Gai's network device 316 uses one or more text-based ACLs with various criteria, including source IP address to determine whether to allow a packet to proceed. Ex.1006, 2:46-57, 3:26-30, 5:3-21, 6:13-18, 21:1-7, Fig. 3, 5A-5E. Gai's ACL criteria is initially set up by an administrator. Ex.1006, 7:16-29 (“[A] network administrator creates one or more access control lists in a conventional manner...The columns of the ACL represent the specific criteria with which network messages are compared.”).

66. A POSITA would have recognized that IP addresses (one criteria in Gai's ACL) may be statically assigned, but they are often dynamically assigned, e.g., using Dynamic Host Configuration Protocol, “DHCP.” Ex.1007, 1:66-2:1 (“The IP address is often assigned using the Dynamic Host Configuration Protocol (DHCP).”). In the circumstance of a dynamically assigned IP address, a POSITA would have recognized that it would be beneficial to learn a new IP address so that the network device 316 applies the latest data when processing packets. Ex.1007, 4:52-55 (“One limitation of utilizing a prior switch was that these were not designed to allow for the fact that source IP addresses are generally not static, so the source IP address for a MAC address can change over time.”). In this way, Gai's network device 316 would be able to apply security rules appropriate for—and potentially specific to—the device to which the IP address was assigned.

67. Kwan beneficially discloses how to learn new IP addresses that have been dynamically assigned. *See e.g.*, Ex.1007, 2:7-3:4. In more detail, Kwan discloses a network device 200 that provides switching functions (analogous to Gai's network device 316). Ex.1007, 3:43-58. Kwan's network device 200 uses an ACL that is formatted for content addressable memory ("ACL-CAM") that includes a "source IP address and [a] corresponding MAC address [that are] correlated with each other as a source IP address/MAC address pair." Ex.1007, 4:10-39. The stored IP address and MAC address pair is used to "control access and switching through the ports." Ex.1007, 4:34-48.

68. In one example, Kwan discloses that an IP address in a received packet is compared with the ACL-CAM to determine if the IP address is new. Ex.1007, 8:17-25. If the packet has "a new source IP address... then a learn source IP routine is performed 314 (various methods related to this are described above)." Ex.1007, 8:25-38; *see also* Ex.1007, 5:38-67 (disclosing different ways to learn the new IP address). For example, "the source IP address can be learned by using a reverse address resolution protocol (RARP) which provides for sending out the MAC address on the subnet, and in response the host having the MAC address responds with its IP address." Ex.1007, 5:38-46. As another example, a new IP address can be learned "by listening to the DHCP response packet being returned to the host...[which] contains the source IP Address for the host. When a DHCP packet is

Ex.1003

detected, the entry in the table for the MAC address receiving the DHCP packet with a source IP address is cleared and the source IP address provided in the DHCP packet is loaded into the table”). Ex.1007, 5:44-50. “If the learning and confirmation of the new source IP address is successful, then the new source IP address is stored [] in a table...include[ing] storing the information in an ACL-CAM.” Ex.1007, 8:38-48.

69. Additionally, Kwan recognizes that there is a “risk of possibly learning the wrong IP address [which] can be reduced by allowing an [sic] system administrator to seed the IP learning process with the IP address properties that are consistent with the subnet to which the host is coupled.” Ex.1007, 6:14-18. According to Kwan, this process “will eliminate false learning of a default 169.0.0.0 addresses assigned by Microsoft or Apple operating systems when a DHCP server is not initially located.” Ex.1007, 6:18-25. A POSITA would have recognized that the administrator provided seed IP address properties allow a new IP address access to the network via the learning process. That is, the seed IP address properties represent data (text data such as IP subnet) that is used to restrict access to the network such that only consistent IP addresses will be learned and permitted to communicate on the network.

70. As analyzed above, in the combination of Gai and Yip it would have been obvious to a POSITA to include MAC addresses in Gai’s administrator

Ex.1003

created text-based ACLs. Furthermore, in view of Kwan's teachings, it would have been obvious to a POSITA to implement Gai's administrator created text-based ACLs to include MAC addresses as well as seed IP address properties and use this information to learn new IP addresses. By including a MAC address and seed IP address properties in the text-based ACLs, Gai's network device 316 would be able to process packets from a host that has a known MAC address but has been assigned a new IP address (e.g., using DHCP), as was often done at the time. Ex.1007, 1:66-2:1 ("The IP address is often assigned using the Dynamic Host Configuration Protocol (DHCP)."). Such an implementation would obviate the need for Gai's administrator to manually generate updated text-based ACLs when the network changes. Ex.1006, 2:57-60, 4:6-13, 7:16-20, 18:3-5; Ex.1007, 7:67-8:1. Since a computer's MAC address is generally static, incorporating Kwan's use of a MAC address in Gai's ACLs would allow the administrator to specify access control rules that are applied to a specific computer irrespective of the IP address assigned dynamically to that computer.

71. Additionally, in implementing the combination it would have been obvious to a POSITA to apply one or more of Kwan's IP learning techniques (e.g., RARP, DHCP). Ex.1007, 5:38-67, 8:25-38. To ensure that learning takes place only for new IP addresses that are consistent, it would have also been obvious to use the seed IP address properties in the ACLs, thereby lowering the "risk of

Ex.1003

possibly learning the wrong IP address.” Ex.1007, 6:14-18. This process would beneficially “eliminate false learning of a default 169.0.0.0 addresses assigned by Microsoft or Apple operating systems when a DHCP server is not initially located.” Ex.1007, 6:18-27; *see also* Ex.1007, 3:1-4 (“In order to increase the efficiency and effectiveness of combating source IP spoofing it would be beneficial to provide source IP spoofing at lower level in the network hierarchy, in a manner which has not previously been provided.”).

72. The proposed combination merely represents a combination of prior art elements (MAC address, seed IP address properties, and RARP or DHCP learning techniques, per Kwan, with Gai’s administrator crated text-based ACLs) according to known methods to yield predictable results (facilitate filtering of packets based on MAC addresses and learning new IP addresses). Also, the combination of Kwan and Gai is merely the use of a known technique (using MAC addresses and seed IP address properties to learn new IP addresses using RARP and DHCP, per Kwan,) to improve a similar system (Gai’s network device 316) in the same way, including (1) provide additional security by only permitting known MAC addresses on the network, (2) learning new IP addresses that are consistent with the seed IP address properties, (3) reducing false learning, and (4) combating source IP spoofing.

c. **Reasonable Expectation of Success**

73. The results would have been predictable and there would have been a reasonable expectation of success in the combination given that both references disclose similar network devices that use ACLs. Any modification to implement Gai's administrator created text based ACLs to include MAC addresses and seed IP address properties, would have been well within the level of skill in the art. *See e.g.*, Ex.1006, 2:58-60 ("To generate an access control list, a network administrator typically defines a sequence of criteria statements using a **conventional** text editor or graphical user interface (GUI).").

74. Also, a POSITA would have had a reasonable expectation of success in applying Kwan's IP address learning techniques (e.g., RARP and DHCP) based on the MAC addresses, since RARP and DHCP were standardized protocols and well known in the art. Ex.1011, 21:1-3; Ex.1014, 1:15-28; Ex.1017, 5; Ex.1018, 4. The reasonable expectation of success is also supported by the specific details that Kwan provides to learn new IP addresses based on MAC addresses using RARP and DHCP techniques.

75. Lastly, I note that the proposed combination relies on the teachings of the respective references, and I do not suggest that physical incorporation of Kwan's elements into Gai is required.

6. Claim 1

a. [1.0] *A method of developing an access control list, comprising:*

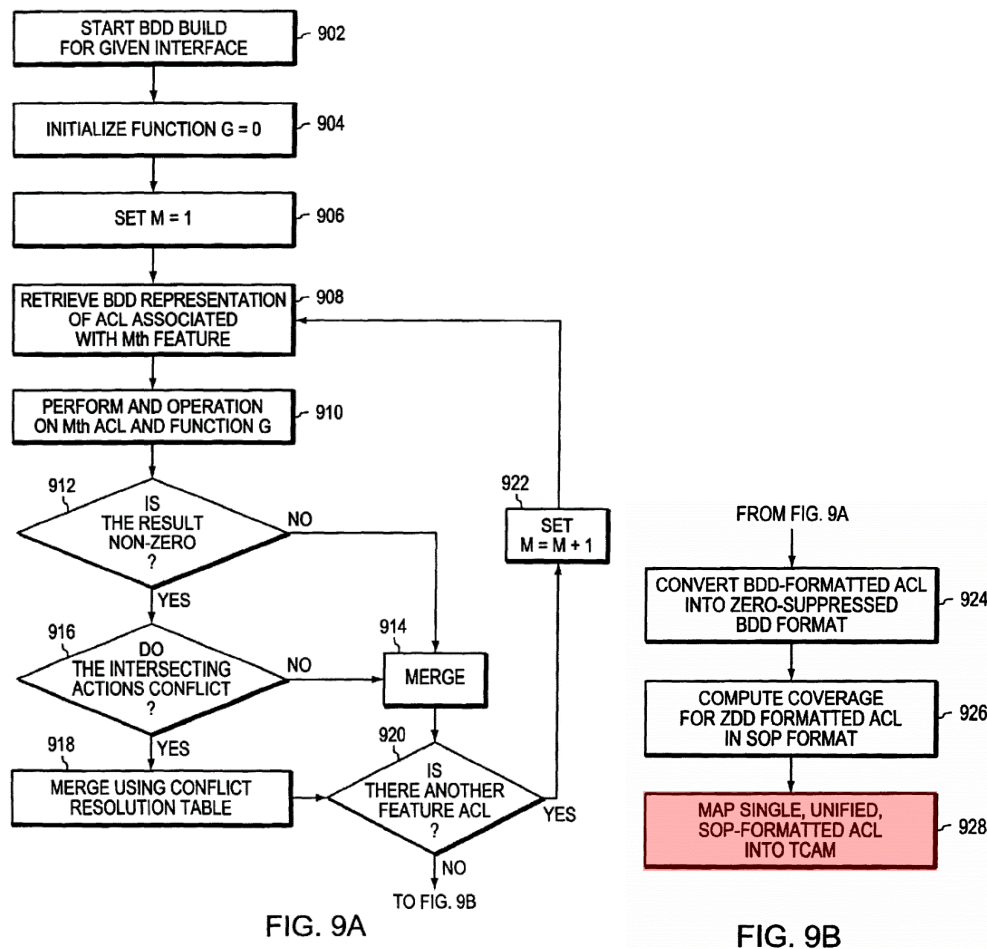
76. To the extent limiting, Gai in combination with Yip and Kwan renders obvious the preamble.

77. First, Gai discloses “**a method and apparatus for efficiently organizing...access control lists (ACLs).**” Ex.1006, 3:55-57. Gai’s method is implemented on a network device 316 (e.g., a switch). Ex.1006, 5:3-6:30. The network device 316 includes an “ACL converter 424 [that] transforms...text-based ACLs into corresponding boolean Binary Decision Diagram (‘BDD’) representations.” Ex.1006, 9:4-27, Fig. 4; *see also* Ex.1006, 9:42-10:26, Fig. 6. “The ACL converter 424 then merges all of the ACLs assigned to each interface into a single, unified ACL and stores these single, unified ACLs in the TCAM 410.” Ex.1006, 9:4-30; *see also* Ex.1006, 10:27-11:4, 12:20-16:17, Figs. 7, 9A-9B.

78. Gai explains that its processing “dynamic[ally]” generates a BDD ACL that is optimized and merged into a single unified TCAM ACL. *See* Ex.1006, 10:49-51 (“...optimizes...using dynamic re-ordering...”), 11:2-4 (“At the end of this process, a single, optimized BDD representation for the entire ACL (e.g., ACL 416a) will have been generated by the ACL converter 424”), 14:57-62 (“At this point, the ACL converter 424 has generated a single, unified, BDD-formatted ACL

that merges all of the ACLs assigned to a particular interface. **The next step is to map this single, unified ACL into the selected storage device (e.g., TCAM 410 and memory 411).”).**

79. Gai’s Figures 9A and 9B are provided below, illustrating a method for dynamically generating optimized ACLs that are stored in TCAM.



Ex.1006, Figs. 9A and 9B (annotated).

80. Gai’s dynamically generated TCAM ACL corresponds to an “*access control list*” as recited in the preamble.

81. Second, as analyzed below at limitations [1.2.4] and [1.3], the further combination with Yip and Kwan adds to Gai's method such that the TCAM ACL is updated to include MAC addresses and newly learned IP addresses.

82. Thus, Gai in combination with Yip and Kwan discloses a method of dynamically generating an optimized TCAM ACL, which renders obvious “[a] *method of developing an access control list*,” as recited in the preamble.

b. [1.1] *developing an enhanced access control list including data related to at least one of user names, DNS names, Windows domain names, and physical addresses*;

83. Gai in combination with Yip and in further combination with Kwan renders obvious this limitation.

84. As discussed in the claim construction section, I am applying Patent Owner's apparent district court interpretation of the claim term “*at least one of*” as requiring only one of the listed items. In the below analysis, I demonstrate that the prior art renders obvious “*developing an enhanced access control list including data related to...physical addresses*.”

85. First, Gai teaches “*developing an enhanced access control list*” by disclosing that the administrator creates one or more text-based access control lists (“text-based ACLs”).

First, a network administrator creates one or more access control lists in a conventional manner. For example, the administrator preferably utilizes a conventional text editor at a management station (not shown) to create the access control lists. FIGS. 5A-5E are highly schematic representations of **text-based ACLs** 416a-416e, respectively. Each access control list, such as ACL 416a, is given a name, such as ACL 101, and is preferably arranged in a table array having multiple rows and columns. Each row of the ACL, such as ACL 416a, corresponds to an Access Control Entry (ACE) statement, such as ACE statements 502-514, which specify the various criteria for the ACL 416a. The columns of the ACL represent the specific criteria with which network messages are compared. For example, ACLs 416a-416d each include a separate column for source address 516, destination address 518, source port 520, destination port 522 and protocol 524.

Ex.1006, 7:15-32.

86. Gai's one or more text-based ACLs provide security and additional features, including packet filtering based on source and destination IP addresses, source and destination port, protocol, QoS, encryption, NAT, etc. *See e.g.*, Ex.1006, 7:29-39, 8:7-9:3, Figs. 5A-5E, 10A-10B.

It is proposed, however, to **expand the use of access control lists for additional features besides just security decisions**. For example, access control lists may also be used to determine whether a given packet should be **encrypted** and/or whether a particular **quality of service (QoS) treatment** should be applied.

Ex.1006, 3:24-30.

In the preferred embodiment, the network administrator utilizes one **ACL for security and one or more other ACLs to implement additional features** at the interface per direction.

Ex.1006, 8:3-6.

In particular, certain layer 3 intermediate devices utilize **access control lists to decide whether received messages should be forwarded or filtered (i.e., dropped) based on certain predefined criteria**. The criteria may be IP source address, IP destination address, or upperlayer application based on TCP/UDP port numbers. For example, an access control list may allow e-mail to be forwarded, but cause all Telnet traffic to be dropped. Access control lists may be established for both inbound and outbound traffic and are most commonly configured at border devices (i.e., gateways or firewalls) to provide security to the network.

Ex.1006, 2:46-57, Figs. 5A-5E; *see also* Ex.1006, 7:41-9:21 (describing that the text-based ACLs support enhanced features such as “encryption function” “QoS” “Network Address Translation (NAT) function” “a web cache control protocol (WCCP) function and so on.”).

87. To the extent argued that the claimed “*enhanced access control list*” requires a single access control list, such would have been obvious. Although Gai’s embodiment discussed above contemplates a plurality of text-based ACLs 416a-

416e, Gai expressly discloses “**one** or more text-based ACLs.” Ex.1006, 3:57-61; *see also* Ex.1006, 20:25-29 (“...**one** or more Access Control Lists (ACLs)...”).

Moreover, as a simple matter of text file management, it would have been obvious for the administrator to organize the rules into a single text-based ACL, with Gai’s subsequent optimization being performed on the rules within the single text-based ACL.

88. Accordingly, Gai’s one or more text-based ACLs, which support enhanced features such as security, encryption, QoS, NAT, WCCP, etc., correspond to the claimed “*enhanced access control list*.”

89. Second, it was generally known in the art for ACLs to include “*data related to...physical addresses*” (e.g., MAC addresses). *See e.g.*, Ex.1014, 6:30-32, 8:27-47, 11:57-62 (disclosing a table that includes IP/MAC addresses corresponding to an access control list); Ex.1017, 55 (“If you choose ‘Allowed Listed’, only those clients whose wireless **MAC addresses are in the access control list** will be able to connect to your Access Point. When ‘Deny Listed’ is selected, these wireless clients on the list will not be able to connect to the Access Point.”); Ex.1018, 13 (“**Access Control List**... Click Allow or Block to enable **MAC address** filtering. Click Disable to disable Access Control feature...Enter the MAC address for each station to be managed, in the MAC Address field and click Add.”) Ex.1017, 84 (“Your PC’s **MAC address is the one listed as Physical**

Ex.1003

Address”); Ex.1016, claim 2 (“...**physical address comprises a Media Access Control (MAC) address**”).

90. Consistent with general knowledge in the art, Yip teaches that it was known in the art to include MAC addresses in an access control list created by an administrator, which in the proposed combination renders obvious for Gai’s text-based ACL to include “*data related to...physical addresses.*”

For example, access lists were developed to combat cyber-attacks on the LAN Switch by providing an **administrator-controlled list of Internet Protocol (IP) addresses or media access control (MAC) addresses that were authorized to access the network.**

Ex.1013, 2:43-47.

91. As already noted, Gai’s text-based ACLs support security and additional functionality. *See e.g.*, Ex.1006, 7:15-32, 8:3-6. Gai also contemplates that additional message criteria may be included in the ACLs. *See e.g.*, Ex.1006, 7:32-34 (“Those skilled in the art will understand that **greater or fewer message criteria may be employed [in the ACLs].**”). A POSITA would have understood that MAC addresses are message criteria (since the Ethernet messages sent and received by Gai’s switch would have contained source and destination MAC addresses) and also that the MAC addresses may be used to provide additional security (e.g., by only permitting access to known MAC addresses). Such an

implementation would improve security because only specific MAC addresses would be permitted to communicate packets, while others would be blocked. And since a MAC address is a physical address associated with a specific piece of hardware, specifying access control rules using MAC address would also allow an administrator to apply a security policy to a specific computer, irrespective of what IP address might be dynamically assigned to that computer. Accordingly, it would have been obvious to a POSITA for Gai's administrator, when creating the text-based ACLs, to include MAC addresses of devices authorized to access the network, per Yip. *See also* Reasons to Combine Yip with Gai.

92. In implementing the proposed combination, Gai's ACL converter 424 would be used to transform the text-based ACLs (including MAC addresses) into BDD representations and then generate a single unified ACL that is stored into TCAM 410. Ex.1006, 9:4-10:26, Fig. 4; *see also* Ex.1006, 10:27-11:4, 12:20-16:17, Figs. 7, 9A-9B. The MAC addressees would enable Gai's network device 316 to restrict access to unauthorized devices based on MAC address.

93. A POSITA would have understood that a MAC address represents data (e.g., a unique number) related to a physical device assigned by the manufacturer, and therefore the MAC addresses in the text-based ACLs (per the Gai and Yip combination) correspond to "*data related to...physical addresses.*" Ex.1018, 21 ("Each device on ethernet and wireless LAN networks has a **media access control**

Ex.1003

(MAC) address, which is a unique 48-bit number assigned to each device by the manufacturer.”); Ex.1017, 84 (“Your PC’s **MAC address is the one listed as Physical Address**”); Ex.1016, claim 2 (“...**physical address comprises a Media Access Control (MAC) address**”).

94. Thus, Gai in combination with Yip discloses creating text-based ACLs that include MAC addresses, which renders obvious “*developing an enhanced access control list including data related to...physical addresses,*” as recited in the claim.

95. Third, Gai and Yip in further combination with Kwan also renders obvious “*developing an enhanced access control list including data related to...physical addresses.*”

96. As already noted above, Gai’s administrator creates one or more text-based ACLs that include source IP addresses. In view of Yip, it would have been obvious for the text-based ACLs to also include MAC addresses. Although a device’s MAC address is fixed (because it is a physical address assigned by the manufacturer), it was general knowledge in the art that IP addresses may change because they are “often assigned using the Dynamic Host Configuration Protocol (DHCP).” Ex.1007, 1:66-2:1; *see also* Ex.1014, 1:17-19 (“DHCP (Dynamic Host Configuration Protocol) allows the dynamic assignment of IP (Internet Protocol) addresses to hosts.”).

97. Because IP addresses may change, Kwan teaches how to automatically learn new IP addresses based on known MAC addresses. Ex.1007, 4:56-58. For example, Kwan discloses that “the source IP address can be learned by using a reverse address resolution protocol (RARP) which provides for sending out the MAC address on the subnet, and in response the host having the MAC address responds with its IP address.” Ex.1007, 5:38-43. As another example, a new IP address can be learned “by listening to the DHCP response packet being returned to the host...[which] contains the source IP Address for the host. When a DHCP packet is detected, the entry in the table for the MAC address receiving the DHCP packet with a source IP address is cleared and the source IP address provided in the DHCP packet is loaded into the table.” Ex.1007, 5:44-50; *see also* analysis at limitation [1.2.4].

98. Consistent with the combination of Gai and Yip, it would have been obvious to a POSITA to include MAC addresses in Gai’s ACLs to facilitate automatically learning new IP addresses per the techniques of Kwan. Ex.1007, 7:61-66 (“[I]t is **important...that the table containing the learned source IP addresses be dynamic**, such that the table can be updated...and new learned source IP addresses can be added to the table.”). The teachings of Kwan would be beneficial to Gai because a new IP address can be learned based on a known MAC address, thereby allowing Gai’s system to support dynamically assigned IP

Ex.1003

addresses. Because the learning process would be automatically performed based on known MAC addresses, the administrator's workload would be reduced, which is another benefit. *See* Ex.1007, 4:56-58 (“[T]he network device 200...automatically learns the source IP address for each MAC address entering a port of the network device 200.”). *See also* Reasons to Combine Kwan with Gai.

99. Moreover, because Gai's TCAM ACL is generated based on the text-based ACLs (which in the proposed combination would include MAC addresses), the TCAM ACL would also include both MAC and IP addresses, analogous to Kwan. Ex.1007, 5:64-67 (“...learn the source IP address and MAC address pairs, and storing the pairs in ACL-CAM for each port.”); Ex.1007, 4:34-39 (“The table which stores the IP address/MAC address pairs can be implement[ed] using ...an access control list...”). The updated TCAM ACL, with the newly learned IP address and corresponding MAC address, would be utilized to permit access to the network. Ex.1007, 8:17-48 (“...the source IP address, or addresses, which have previously been learned and stored in a table as source IP address which is permitted access to the network...”).

100. Additionally, to reduce the risk of learning wrong IP addresses, Kwan discloses that the system administrator provides seed IP address properties.

Further the risk of possibly learning the wrong IP address can be reduced by allowing an [sic] system **administrator to seed the IP**

learning process with the IP address properties that are consistent with the subnet to which the host is coupled. For example, if the IP Subnet is 10.32.1.0/24, then **this information can be used to seed the EP learning process and only match a MAC address with [] IP addresses consistent with the subnet 10.32.1.0/24 addresses. This will eliminate false learning.**

Ex.1007, 6:14-25. A POSITA would have recognized that the administrator provided seed IP address properties represent data (text data such as IP subnet) that is used to restrict access to the network such that only consistent IP addresses will be learned and permitted to communicate on the network. It would have been obvious to a POSITA for the seed IP address properties to be implemented in a text-based ACL, since this data is used to restrict access to the network.

101. Accordingly, in addition to including a MAC address in Gai's text-based ACLs, it further would have been obvious to a POSITA, in view of Kwan, to include seed IP address properties in Gai's administrator created text-based ACLs, to beneficially reduce or eliminate false learning in certain instances. Ex.1007, 6:14-25 ("For example, if the IP Subnet is 10.32.1.0/24, then this information can be used to seed the EP learning process and only match a MAC address with [] IP addresses consistent with the subnet 10.32.1.0/24 addresses. This will **eliminate false learning.**"). Further, Kwan's teachings would be beneficially helpful to avoid spoofing. Ex.1007, Abstract ("The system and method provide for analyzing MAC

addresses and source IP addresses ... to block access through a port where a host device is using a false, or spoofed, source IP address in transmitted data packets.”).

102. Implementing Gai’s text-based ACLs to include seed IP address properties and MAC addresses would facilitate IP address learning, which would have been recognized by POSITAs as desirable and consistent with Gai’s teaching that the ACLs provide “**security...[and] implement additional features.**”

Ex.1006, 8:3-6; *see also* Reasons to Combine Kwan with Gai.

103. As I noted above, a POSITA would have understood that a MAC address represents data (e.g., a unique number) related to a physical device assigned by the manufacturer, and therefore the MAC addresses in the text-based ACLs (per the Gai, Yip, Kwan combination) correspond to “*data related to...physical addresses.*” Ex.1018, 21; Ex.1017, 84; Ex.1016, claim 2. Additionally, a POSITA would have understood that seed IP address properties are “*data related to...physical addresses,*” at least because the seed IP address properties would be used to restrict the IP addresses to be learned for a known MAC address, in the combination of Gai, Yip, and Kwan. Ex.1007, 6:14-25 (“...this information can be used to seed the EP learning process and only match a MAC address with [] IP addresses consistent...”).

104. Thus, Gai in combination with Yip and Kwan discloses creating one or more text-based ACLs that include MAC addresses and seed IP address properties,

Ex.1003

which renders obvious “*developing an enhanced access control list including data related to...physical addresses,*” as claimed.

c. [1.2.1] *converting, in a network switch, at least one of:*

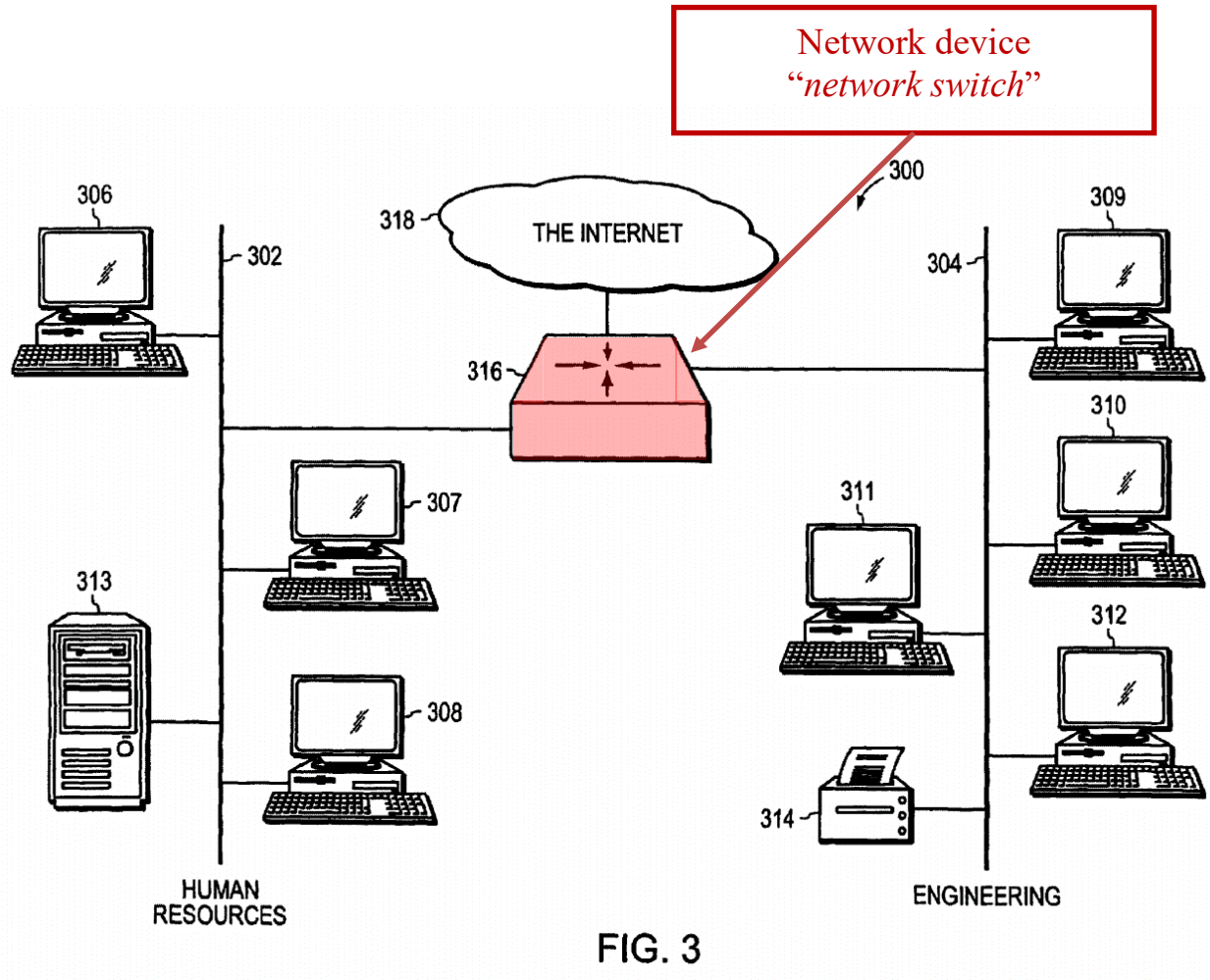
105. Gai in combination with Yip and Kwan renders obvious this limitation.

106. **First**, Gai discloses a “*network switch*” by teaching an “network device 316” that performs “switching functions” in “network 300.”

Preferably, intermediate **network device 316** is a layer 3 intermediate device and thus operates up to the network layer of the communication protocol stack implemented within the network 300. For example, device 316 preferably includes an Internet Protocol (IP) software layer. Device 316 may also implement network services such as route processing, path determination and path **switching functions**. In the illustrated embodiment, device 316 is a computer having transmitting and receiving circuitry and components, including network interface cards (NICs), establishing physical ports and interfaces for exchanging network messages.

Ex.1006, 5:22-34; *see also* Ex.1006, 6:62-65 (“A suitable platform for intermediate **network device 316** are the Catalyst 8500® series of switch routers and/or the Catalyst® 6000 family of **multi layer switches** both from Cisco Systems, Inc.”).

107. Gai’s network device 316, which performs switching functions, is illustrated below at Figure 3. *See also* Ex.1006, Fig. 4 (disclosing details of the network device 316).



Ex.1006, Fig. 3 (annotated).

108. Thus, Gai’s network device 316 that performs switching functions renders obvious “*a network switch,*” as claimed.

109. Second, as discussed in the claim construction section, I am applying Patent Owner’s apparent district court interpretation of the claim term “*at least one of*” in this proceeding to require only one of the listed “*converting*” steps recited at limitations [1.2.2]-[1.2.4]. In the below analysis, I demonstrate that the combination

of Gai, Yip, and Kwan renders obvious “*converting... physical addresses into IP addresses according to data in the enhanced access control list*,” which is recited at limitation [1.2.4].

- d. **[1.2.2] [converting...] *user names into corresponding IP and physical addresses according to data in the enhanced access control list, wherein converting user names into corresponding IP and physical addresses according to data in the enhanced access control list includes: detecting login packets being communicated over the network; determining a MAC address from the login packets; detecting server message block login packets being communicated over the network; determining an IP address from the server message block login packets; and developing records in the access control list using the obtained IP address for the respective user name;***

110. As I have already noted, I am applying Patent Owner’s apparent district court interpretation of the claim term “*at least one of*” at limitation [1.2.1] to require only one of the listed converting steps to be performed, and therefore the converting step recited at limitation [1.2.2] is not required.

- e. **[1.2.3] [converting...] *DNS names into corresponding IP addresses according to data in the enhanced access control list; and***

111. As I have already noted, I am applying Patent Owner’s apparent district court interpretation of the claim term “*at least one of*” at limitation [1.2.1] to require only one of the listed converting steps to be performed, and therefore the converting step recited at limitation [1.2.3] is not required.

f. [1.2.4] *[converting...] physical addresses into IP addresses according to data in the enhanced access control list; and*

112. Gai in combination with Yip and Kwan renders obvious this limitation.

113. **First**, as discussed at limitation [1.1], Gai in combination with Yip and Kwan discloses one or more text-based ACLs that includes data such as MAC addresses and seed IP address properties, which renders obvious “*enhanced access control list including data related to...physical addresses.*” And as discussed at limitation [1.2.1], Gai discloses a network device 316 that performs switching, which renders obvious a “*network switch.*”

114. **Second**, in two different ways, the combination of Gai, Yip, and Kwan renders obvious “*converting, in the network switch...physical addresses into IP addresses according to data in the enhanced access control list.*” Of note is that the ’518 patent discloses using DHCP for converting MAC addresses into IP addresses and ARP for converting IP addresses into MAC addresses. *See* Ex.1001, 7:51-8:21. First, as analyzed below, Kwan’s network device 200 uses a DHCP technique that is substantially similar to the technique of the ’518 patent. Second, Kwan discloses that network device 200 uses a RARP technique to convert physical addresses of devices on the network into IP addresses (this is analogous, but inversely related, to the ARP technique of the ’518 patent). *See e.g.*, Ex.1007, 3:53-58, 5:38-6:27, Fig. 2.

(1) Converting using DHCP technique

115. In a first example, Kwan discloses listening to DHCP response packets to learn new IP addresses according to MAC addresses.

The source IP address detector 236 could also learn the IP address for a host by **listening to the DHCP response packet being returned to the host. This response contains the source IP Address for the host. When a DHCP packet is detected, the entry in the table for the MAC address receiving the DHCP packet with a source IP address is cleared and the source IP address provided in the DHCP packet is loaded into the table.**

Ex.1007, 5:44-50; Ex.1007, 8:46-48 (“As discussed above this storing of the new source IP address in a table, can include storing the information in an ACL-CAM.”).

116. A new IP address in the DHCP packet is only matched to a known MAC address if it is consistent with seed IP properties.

Further the risk of possibly learning the wrong IP address can be reduced by allowing an [sic] system administrator to seed the IP learning process with the IP address properties that are consistent with the subnet to which the host is coupled. For example, if the IP Subnet is 10.32.1.0/24, then **this information can be used to seed the EP learning process and only match a MAC address with [] IP addresses consistent** with the subnet 10.32.1.0/24 addresses. This will eliminate false learning of a default 169.0.0.0 addresses assigned by

Microsoft or Apple operating systems when a DHCP server is not initially located. It is also advantageous to confirm that the learned IP address is correct by performing a reverse IP check.

Ex.1007, 6:14-27.

117. A POSITA would have understood that in the above DHCP technique, a new IP address included in the return DHCP packet is matched with the MAC address in the ACL if the new IP address is consistent with seed IP properties and that the matched IP address is thereby learned. By matching the MAC address with a new IP address, the MAC address (which as discussed at [1.1] is included in the text-based ACL “*enhanced access control list*”) is “*convert[ed]*” into an IP address.

118. Kwan’s disclosure of listening to DHCP return packets is substantially similar to the ’518 patent’s converting technique of monitoring DHCP return packets and correlating an IP address in a DHCP return packet to a MAC address. *See e.g.*, Ex.1001, 7:51-67 (“DHCP monitoring and MAC conversion subprocess 204...converts MAC addresses into corresponding IP addresses. **The subprocess 204 monitors DHCP return packets from a DHCP server, which corresponds to one of the hosts 106 (FIG. 1), and from these return packets determines the IP address assigned by the DHCP server to a given MAC address... the subprocess 204 need merely determine the assigned IP address contained in the return packet for the destination MAC address of the packet.** The subprocess

Ex.1003

204...applies the rules of the enhanced access control list 108 to generate a corresponding record in the dynamic access control list 104.”); *see also* Ex.1001, claim 7 (“...*converting physical addresses into IP addresses...comprises...monitoring DHCP packets...*”).

(2) Converting using RARP technique

119. In a second example, Kwan discloses that “the source IP address can be learned by using a **reverse address resolution protocol (RARP) which provides for sending out the MAC address on the subnet, and in response the host having the MAC address responds with its IP address.**” Ex.1007, 5:38-43. Consistent with the analysis above, learning would occur only if the known MAC address sent out is correlated with the new IP address in the received RARP packet and also if the new IP address is consistent with seed IP properties.

120. Kwan’s RARP learning technique is analogous to the ’518 patent’s use of ARP to correlate a known IP address to a new MAC address—Kwan is performing the ARP protocol in reverse because the MAC address is known. *See* Ex.1001, 8:1-21. A POSITA would have understood that Kwan’s RARP technique, which allows for correlating a new IP address with a known MAC address, would be within the scope of the recited “*converting... physical addresses into IP addresses.*”

121. In addition to the reasons for combining Kwan with Gai (as modified per Yip), a POSITA would have been motivated to apply Kwan's DHCP and RARP techniques to Gai, because it would allow for learning new IP addresses to thereby dynamically update the TCAM ACLs. Ex.1007, 7:61-66 (“[I]t is **important...that the table containing the learned source IP addresses be dynamic**, such that the table can be updated...and new learned source IP addresses can be added to the table.”); *see also* analysis at limitation [1.3]. Kwan's DHCP and RARP techniques would be beneficial because they would allow Gai's system to support dynamically assigned IP addresses, whose use was prevalent. *See* Ex.1014, 1:16-19 (“Various network protocols are used to provide networking services to devices. For example, DHCP (Dynamic Host Configuration Protocol) allows the dynamic assignment of IP (Internet Protocol) addresses to hosts.”); Ex.1017, 20 (“Click ‘OK’ to confirm the setting. Your PC will now obtain an IP address automatically from your router's DHCP server.”); Ex.1018, 10 (“DHCP Client...Default: Enable”). A POSITA would have had a reasonable expectation of success in combining the DHCP and RARP teachings of Kwan with Gai because these were well-known and standardized protocols. *see also* Reasons to Combine Kwan with Gai.

122. In the proposed combination with Kwan, a POSITA would have implemented Gai such that its network device 316 converts physical addresses of devices on the network into new IP addresses according to data (e.g., known MAC

Ex.1003

addresses and seed IP address properties), which in the proposed combination would be included in one or more text-based ACLs (“*an enhanced access control list*” as discussed at limitation [1.1]).

123. Thus, Gai in combination with Yip and Kwan discloses converting in the network device 316 physical addresses of devices on the network into IP addresses (e.g., using a DHCP technique or a RARP technique) according data (e.g., known MAC addresses and seed IP address) in one or more text-based ACLs, which renders obvious converting in a network switch “*physical addresses into IP addresses according to data in the enhanced access control list,*” as claimed.

g. [1.3] *developing, in the network switch, the access control list from each of the operations of converting.*

124. Gai in combination with Yip and Kwan renders obvious this limitation.

125. First, I am applying Patent Owner’s apparent district court interpretation of the claim term “*at least one of*” at limitation [1.2.1] to require only one of the listed converting steps to be performed. Additionally, I am applying Patent Owner’s apparent district court interpretation of “*each of*” at limitation [1.3] to require only one converting operation recited at limitations [1.2.2]-[1.2.4].

126. Second, as discussed at the preamble [1.0], Gai in combination with Yip and Kwan discloses a method to develop an optimized TCAM ACL (“*the access control list.*”). And as discussed at limitation [1.2.4], the combination of Gai,

Yip, and Kwan discloses converting physical addresses of devices on the network into IP addresses using a DHCP technique or a RARP technique (*“the operations of converting”*).

127. Third, Kwan discloses that once a new IP address has been successfully learned (e.g., using a DHCP technique or a RARP technique), the learned IP address is **“loaded into the table.”** Ex.1007, 5:47-50. Kwan’s **“table is embodied as an access control list**, which are data fields, included in a content addressable memory 240, which is **referred to as an ACL-CAM.”** Ex.1007, 4:35-39.

128. A POSITA would have recognized that Kwan’s CAM is a specialized memory circuit that is designed to allow its entire contents to be searched for a matching value all at once and, consequently, very fast. Ex.1007, 4:35-39. That makes a CAM ideal for performing lookup operations as part of packet processing (for example, looking up the rule or rules to apply to a packet based on packet information such as its source and destination addresses). Similarly, Gai disclosure pertains to CAM. Ex.1006, 4:13-16 (*“The network device further includes an additional memory device, such as an associative or **content addressable memory (CAM).**”*), 6:3-7 (*“...a **ternary content addressable memory (TCAM)**...”*). The term “ternary” refers to a CAM that can store and retrieve data using “0,” “1,” and a

wild card “X.” Ex.1006, 8:28 (“The value ‘x’...represents a wildcard.”). As such, a POSITA would have understood that Kwan’s teachings readily apply to Gai.

129. A POSITA implementing Kwan’s DHCP or RARP techniques in the combination with Gai (see limitation [1.2.4]) would have found it obvious to update Gai’s TCAM ACL with the new IP address once learned, so that the network device 314 would permit packets that have new IP addresses. *See* Ex.1006, 18:3-5 (“...networks are changed and up-dated...”). Updating the TCAM ACL is consistent with Gai’s recognition that ACLs may be **“routinely up-dated or changed.”** Ex.1006, 18:44-48. Indeed, Kwan recognizes the importance of updating the ACL-CAM because “the source IP addresses assigned for a given host **can change overtime.**” Ex.1007, 7:66-8:1; *see also* Ex.1007, 7:61-66 (“**[I]t is important** in many applications of the source IP security operations herein, that the table containing the learned source IP addresses be dynamic, such that the table can be updated so that timed out source IP addresses can be removed and **new learned source IP addresses can be added to the table.**”). *See also* Reasons to Combine Kwan with Gai.

130. Consistent with the analysis at the preamble [1.0], Gai’s network device 316 dynamically optimizes one or more text-based ACLs into a single unified TCAM ACL. *See* Ex.1006, 5:3-6:30, 9:4-11:4, 12:20-16:17, Figs. 4, 6, 7, 9A-9B. A POSITA seeking to update Gai’s TCAM ACL to include newly learned

Ex.1003

IP addresses (per Kwan), would have found it obvious to first update one or more of Gai's text-based ACLs to include the new IP addresses, and then to dynamically reoptimize the updated text-based ACLs to generate an updated TCAM ACL that includes new IP addresses. This processing sequence would allow for Gai's network device 316 to apply optimized TACM ACLS with new IP addresses to received packets. The noted implementation is consistent with Gai's statement that ACLs may be "**routinely up-dated or changed.**" Ex.1006, 18:44-48.

131. Thus, Gai in combination with Yip and Kwan discloses developing a reoptimized TCAM ACL in the network device 314 that includes new IP addresses from the operation of converting, which renders obvious "*developing, in the network switch, the access control list from each of the operations of converting,*" as claimed.

7. Claim 3

a. [3.1] *The method of claim 1 wherein each physical address comprises a MAC address.*

132. As discussed at limitations [1.1] and [1.2.4], the combination of Gai, Yip, and Kwan discloses that the text-based ACLs include MAC addresses of devices authorized to access the network, which renders obvious that "*each physical address comprises a MAC address.*"

B. Ground 2: Claims 15, 17, 19-21 are Obvious over Gai in combination with Yip, Kwan, and Georgiou

1. Summary of Georgiou

133. U.S. 7,003,597 to Georgiou *et al.* (Ex.1008, “Georgiou”) was filed on July 9, 2003. Georgiou generally “relates to the implementation of computer networks” that “employ various network protocols such as, for example, the Internet Protocol (IP).” Ex.1008, 1:8-18. Georgiou discloses using a direct memory access controller DMA 140 to store Internet Protocol (IP) packets in memory while awaiting processing.

FIG. 1 is a block diagram showing an embodiment of an environment of the invention. A network device, generally shown as 100, typically includes a host processor and one or more applications 110. The network device 100 is connected to a network 120 for receiving and transmitting packets of data using various types of protocols via a network interface 130. The network interface 130 temporarily stores an incoming packet in its local memory until the incoming packet may be transferred via a direct memory access controller (DMA) (or equivalent) to a buffer pool such as small buffer pool 200 or large buffer pool 250. Any memory data structures (e.g., tables, linked lists, small buffer pool 200, large buffer pool 250, counters, timers, etc.) contained in the memory are typically under control of one or more memory manager components 160 as part of the host processor 110 complex, but may involve distributed architectures.

Ex.1008, 3:36-51; *see also* Ex.1008, 1:22-27 (“Typical arrangements include employing some form of logic, either hardware centric or software centric, to receive and temporarily store the incoming packets in a memory area, e.g., a buffer pool or equivalent, until subsequent processing can access and retrieve the packets, as appropriate, for action on the packet contents.”); Ex.1008, 3:17-22 (“...the packet is typically...transferred to memory to await subsequent processing.”); Ex.1008, 3:36-4:26.

134. Georgiou’s Figure 1, reproduced below, illustrates a direct memory access controller DMA 104 that stores packets in memory (e.g., small buffer 200 and large buffer 250) while awaiting processing by the host processor 110.

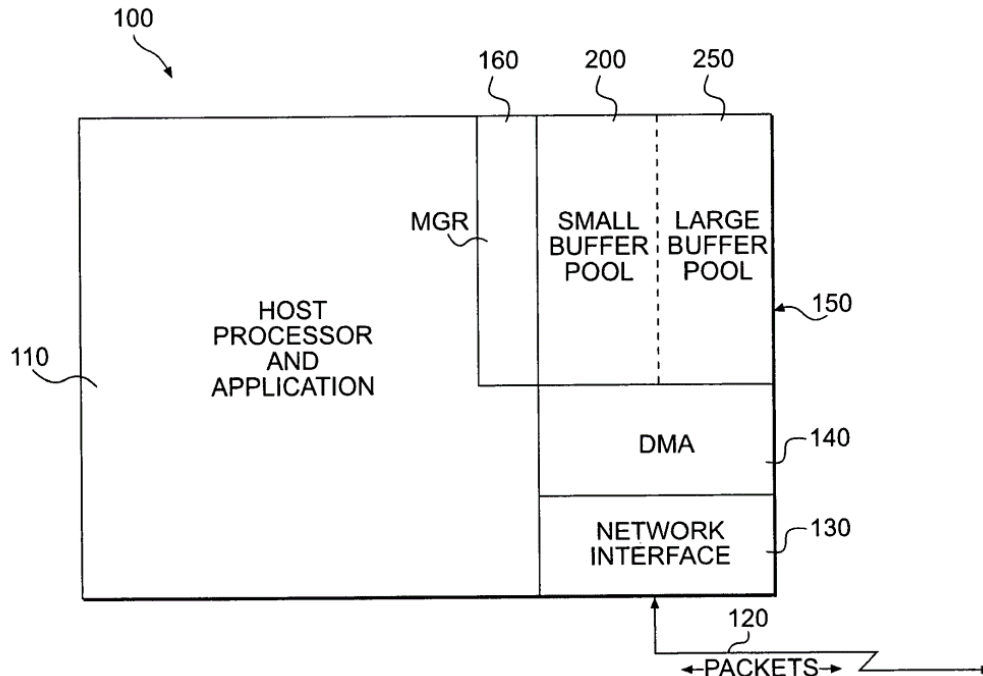


FIG. 1

Ex.1008, Fig. 1.

2. Reasons to Combine Georgiou with Gai

135. A POSITA would have been motivated to combine the teachings of Georgiou with Gai (as modified per Yip and Kwan), as discussed below.

a. Analogous Art

136. Georgiou is analogous art to the '518 patent. Just like the '518 patent, as noted above in the summary section, Georgiou generally pertains to computer networks and address the problem of storing packets in memory for subsequent processing, including utilizing a DMA controller to store packets in memory.

Ex.1001, 9:28-41; Ex.1008, 3:36-4:26, Fig. 1.

Ex.1003

b. Motivation to Combine

137. A POSITA would have been motivated to combine the teachings of Georgiou with Gai (as modified per Yip and Kwan) to produce predictable and beneficial results as discussed below.

138. Gai discloses that packets are “forward[ed] to CPU for additional processing (e.g., encryption, logging, etc.).” Ex.1006, 13:41-45; *see also* Ex.1006, 12:2-7. Further, as discussed in the combination with Yip and Kwan, it would have been obvious to a POSITA to identify packets that have a new IP address. It would have been obvious to a POSITA to forward packets with new IP addresses to Gai’s CPU for additional processing (i.e., learning the new IP address). A POSITA would have recognized that in the instance where there are significant number of packets that need processing (e.g., encryption, logging, learning new IP address, etc.), the CPU may not be able to keep up with the packet flow.

139. Georgiou teaches that typically IP packets are stored in dynamic memory while awaiting processing. Ex.1008, 5:51-67. According to Georgiou, “[t]ypical arrangements include...receive and temporarily store the incoming packets in a memory...until subsequent processing can access and retrieve the packets, as appropriate, for action on the packet contents.” Ex.1008, 1:22-27; *see also* Ex.1008, 3:18-22. Georgiou uses a “DMA circuit that receives the packet, [and] writes it to memory.” Ex.1008, 1:54-57; *see also* Ex.1008, 3:42-46.

140. It would have been obvious to a POSITA, in view of Georgiou, to implement Gai's network switch to store packets forwarded to a CPU for additional processing (e.g., encryption, logging, learning IP addresses, etc.) in its dynamic memory. Storing the packets in memory would allow the packets awaiting processing to be retained until the CPU is freed up, thereby facilitating processing of packets (e.g., encrypt, log, learn new IP address, etc.).

141. Additionally, because the packets are stored in dynamic memory, the packets are not dropped for lack of CPU processing bandwidth and can therefore be forwarded to the destination postprocessing. Ex.1007, 5:22-6:30; Ex.1008, 9:7-12. The proposed combination would also reduce the number of packets that need to be retransmitted (since fewer packets would be dropped), and thus, make more efficient use of Gai's network bandwidth.

142. The combination of Georgiou and Gai (as modified per Yip and Kwan) is merely combining prior art elements (storing packets in memory, per Georgiou, with Gai's dynamic memory) according to known methods (Georgiou describes exemplary methods of implementing the storing of packets in memory, including using a DMA circuit) to yield the predictable result of retaining packets until the CPU has sufficient bandwidth to process the packets.

143. The combination is also the use of a known technique (storing packets that await processing in memory, per Georgiou) to improve a similar system (Gai's

Ex.1003

network device) in the same way, including (1) retaining the packets until CPU has sufficient bandwidth to process the packets, (2) facilitate processing of packets and (3) conserving network bandwidth by avoiding packet retransmission.

144. The above noted benefits, separately and together, would have motivated a POSITA to implement Gai's network device to store packets that await processing by the CPU in dynamic memory, per Georgiou.

c. Reasonable Expectation of Success

145. The results would have been predictable and there would have been a reasonable expectation of success in the combination given the similarity in Georgiou's and Gai's systems. Any modification needed to Gai, in order to accommodate the teachings of Georgiou, including implementing a DMA circuit to store incoming packets in Gai's dynamic memory, would have been a conventional and simple implementation and well within the level of one of ordinary skill in the art. Ex.1008, 1:54-57 ("...often a simple DMA circuit that receives the packet, writes it to memory, and then indicates to the controlling processor that a packet has been received."); Ex.1009, 1:32-34 ("A direct memory access (DMA) controller is often used."); Ex.1010, [0216], ("[M]essages can also be placed directly into CPU memory through a DMA access using an ARL DMA controller"); Ex.1011, 11:43-45 ("The LAN 1 controller 110 then cooperates with a DMA controller 172 to store

the data packet.”); Ex.1012, Abstract (“a direct memory access (DMA) controller and storing the received [packets of] data in a host memory.”).

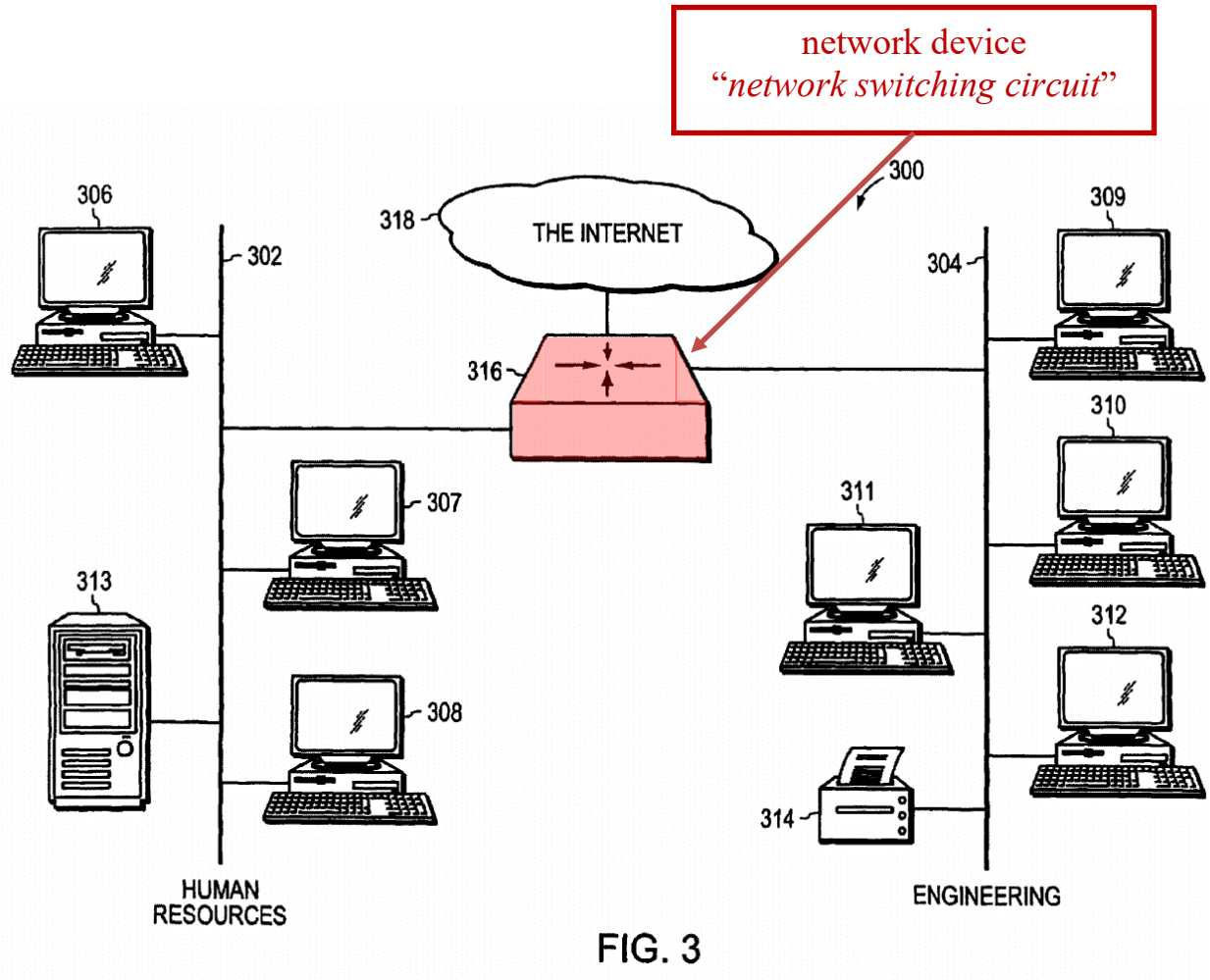
146. Lastly, I note that the proposed combination of Georgiou and Gai (as modified per Yip and Kwan) relies on the teachings, and not on physical incorporation of elements.

3. Claim 15

a. [15.0] *A network switching circuit, comprising:*

147. To the extent limiting, Gai renders obvious the preamble.

Consistent with the analysis at limitation [1.2.1], Gai discloses a “*network switching circuit*” by teaching a “**network device 316**” that includes “**circuitry**” for performing “**switching functions.**” Ex.1006, 5:22-34; *see also* Ex.1006, 5:3-59, 6:27-65.



Ex.1006, Fig. 3 (annotated).

148. Thus, Gai's network device 316 that includes switching circuitry renders obvious "[a] network switching circuit," as recited in the preamble.

149. The remaining claimed features and how the prior art renders them obvious are addressed below. *See* analysis at [15.1.1] to [15.3.5].

b. [15.1.1] *a forwarding circuit operable to detect specific received packets and*

150. Gai in combination with Kwan renders obvious this limitation.

151. First, Gai discloses “a forwarding circuit” by disclosing a “forwarding entity 404,” which is illustrated at Figure 4 reproduced below. Ex.1006, 6:18-30, 8:26-27.

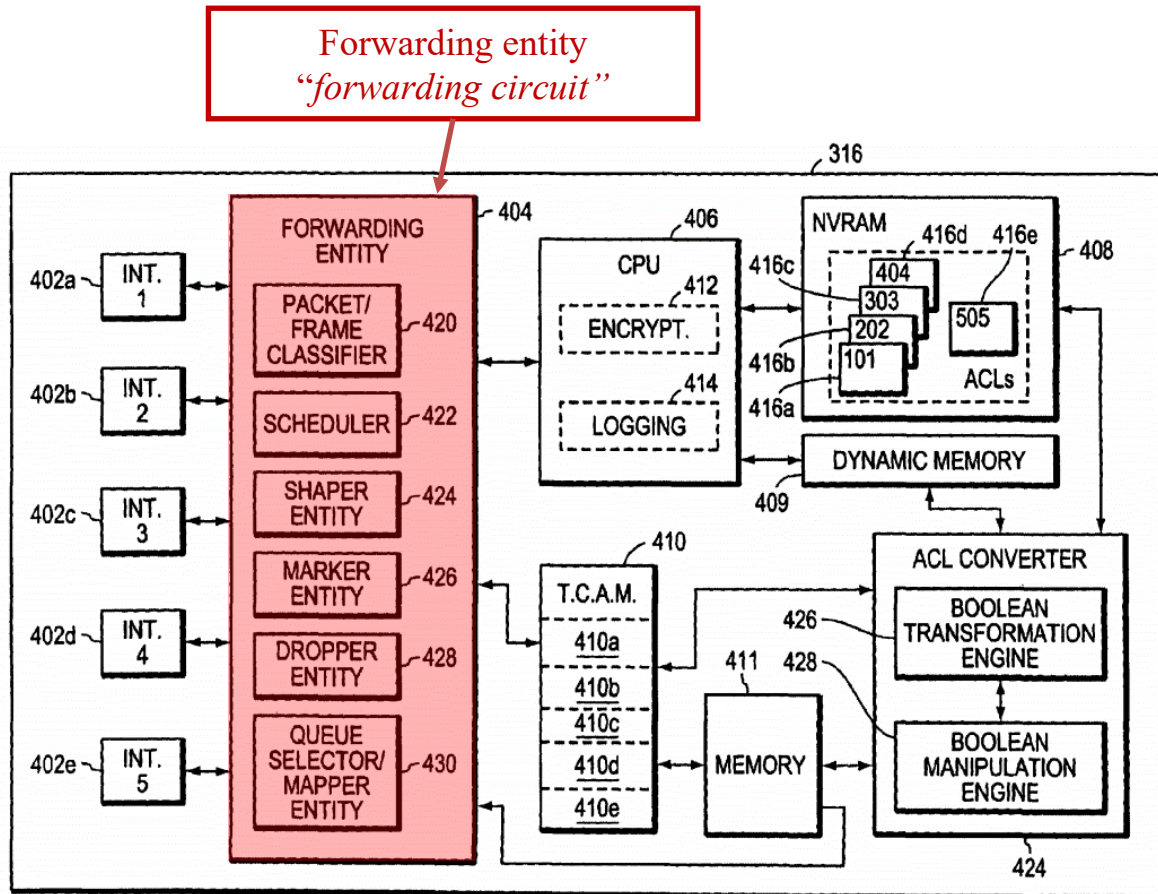


FIG. 4

Ex.1006, Fig. 4 (annotated).

Forwarding entity 404 may include a plurality of conventional sub-components configured to implement QoS treatments, such as a packet/frame classifier 420, a scheduler 422, a shaper entity 424, a marker entity 426, a dropper entity 428, and a queue selector/mapping entity 430. The forwarding entity 404 is also coupled to the CPU 406,

Ex.1003

the TCAM 410, which may be apportioned into segments 410*a-e* whereby each segment 410*a-e* corresponds to a particular interface 402*a-e*, and memory 411. As described below, the forwarding entity 404 is basically configured to forward or switch network messages among the various interfaces 402*a-e* of device 316.

Ex.1006, 6:18-30; *see also* Ex.1006, 17:58-63 (“...**forwarding entity 404 may include one or more Application Specific Integrated Circuits (ASICs)**...”); Ex.1006, 3:37-40 (“[M]ore and more internetworking functionality is being implemented in **hardware circuitry** to increase the speed and performance of internetworking devices.”). Accordingly, Gai’s “forwarding entity 404” renders obvious the claimed “*forwarding circuit*.”

152. Second, Gai in combination with Kwan, in two different ways, renders obvious “*detect[ing] specific received packets*.”

1) Detecting specific received packets that should be encrypted, logged, etc.

153. Gai discloses that its “forwarding entity 404 is basically configured to forward or switch network messages.” Ex.1006, 6:27-30; *see also* Ex.1006, 5:66-6:12. The messages correspond to received “**packets**” communicated over the computer network 300 using standards such as “Ethernet,” “Internet Protocol (IP),” among others. Ex.1006, 5:3-59. Ex.1006, 1:37-2:41, 4:41-42.

154. Gai's forwarding entity 404 utilizes a TCAM ACL to identify specific received packets that need additional processing (e.g., packets that should be encrypted, logged, etc.).

For example, **access control lists may also be used to determine whether a given packet should be encrypted** and/or whether a particular quality of service (QoS) treatment should be applied.

Ex.1006, 3:26-30; *see also* Ex.1006, 13:3-45 ("The final or merged actions to be implemented by a given forwarding entity may include permit, deny, forward to CPU for **additional processing (e.g., encryption, logging, etc.)**"); Ex.1006, 11:60-12:7 ("...a permit action from an encryption ACL requires that the message be forwarded to the CPU 406 for encryption, assuming the corresponding action from the security ACL is also permit.").

The ACL converter 424 then merges all of the ACLs assigned to each interface into a single, unified ACL and stores these single, **unified ACLs in the TCAM 410 for subsequent use by the forwarding entity 404.**

Ex.1006, 9:27-31. Accordingly, Gai teaches that the forwarding entity 404 uses TCAM ACL to identify specific received packets that need additional processing (e.g., packets that should be encrypted, logged, etc.).¹

¹ I note that Gai also teaches that rather than using TCAM, "the ACLs may be left in EDD or ZDD format, stored at the dynamic memory 409 and evaluated,

2) **Detecting specific received packets with new IP addresses²**

155. Gai states that “[t]hose skilled in the art will understand that greater or fewer message criteria may be employed,” which would be understood by a POSITA as suggesting that other criteria may be used with the ACLs to detect specific packets. Ex.1006, 7:32-34.

156. In that regard, Kwan discloses a network device 200 that includes “an **access control list**, which are data fields, included in a content addressable memory 240, **which is referred to as an ACL-CAM.**” Ex.1007, 4:34-39; *see also* 4:47-48. The access control list (called ACL-CAM) stores source IP address and MAC address pairs. Ex.1007, 6:28-67, 8:17-51. The IP address and MAC address pairs are used to identify specific received packets that have a new IP address that has not been learned.

To determine if the data packets received at a port contain a new source IP address that has not been learned, the ACL-CAM

essentially through software, by the forwarding entity 404.” Ex.1006, 18:65-19:3. In the below analysis, I apply Gai’s TCAM ACL disclosure, with the understanding that the analysis equally applies when the ACL is stored in dynamic memory 409.

² To provide antecedent basis for later limitations, I also rely on the combination of Gai and Kwan.

compares the source IP address and the MAC address in a received data packet, with the table of IP address/MAC address pairs.

Ex.1007, 5:25-30. Then, a dynamic “learn source IP routine” is performed.

Ex.1007, 8:17-47, Fig. 3; *see also* Ex.1007, 5:38-39 (“The source IP address detector 236, can learn the source IP address for a host in a number of different ways.”); Ex.1007, 5:39-67, 7:61-66. Kwan’s IP address learning uses the MAC addresses in the ACL. *See e.g.*, Ex.1007, 5:44-6:27. Accordingly, Kwan teaches an ACL that includes IP and MAC address pairs that are used to identify specific packets that have new IP addresses and dynamically learn the new IP addresses based on the MAC addresses.

157. Gai’s ACL includes “source address.” Ex.1006, 7:29-39; *see also* Ex.1006, 21:1-7 (“Internet Protocol (IP) source address.”). It would have been obvious to a POSITA for Gai’s forwarding entity 404 to identify packets that have a new IP address that has not been learned and to dynamically learn the new IP address per the techniques of Kwan, which utilize MAC addresses. Ex.1007, 7:61-66 (“[I]t is important...that the table containing the learned source IP addresses be dynamic, such that the table can be updated...and new learned source IP addresses can be added to the table.”); *see also* Ex.1007, 5:44-6:27. A POSITA would have recognized that the teachings of Kwan would be beneficial because it would allow Gai’s system to support devices that have IP addresses assigned using DHCP. *See*

Ex.1007, 1:66-2:1 (“The IP address is often assigned using the Dynamic Host Configuration Protocol (DHCP).”); *see also* Reasons to Combine Kwan with Gai.

158. Thus, Gai in combination with Kwan discloses a forwarding entity 404 that is operable to identify specific received packets that need additional processing (e.g., encrypted, logged, etc., and packets that have a new IP address that has not been learned), which renders obvious “*a forwarding circuit operable to detect specific received packets,*” as claimed.

c. [15.1.2] to provide the specific packets on a processor port, and

159. Gai in combination with Kwan renders obvious this limitation.

160. First, as discussed at limitation [15.1.1], Gai in combination with Kwan discloses that forwarding entity 404 identifies specific received packets that need additional processing (e.g., encrypted, logged, etc., and packets that have a new IP address that has not been learned), which renders obvious “*the specific packets.*”

161. Second, Gai teaches providing packets to a port of CPU 406 for additional processing.

In particular, upon receipt of a packet at a first interface, a forwarding entity at the intermediate network device tests the packet against the single, unified ACL stored in the corresponding portion of the CAM. When a match is obtained, the corresponding decision is returned to the

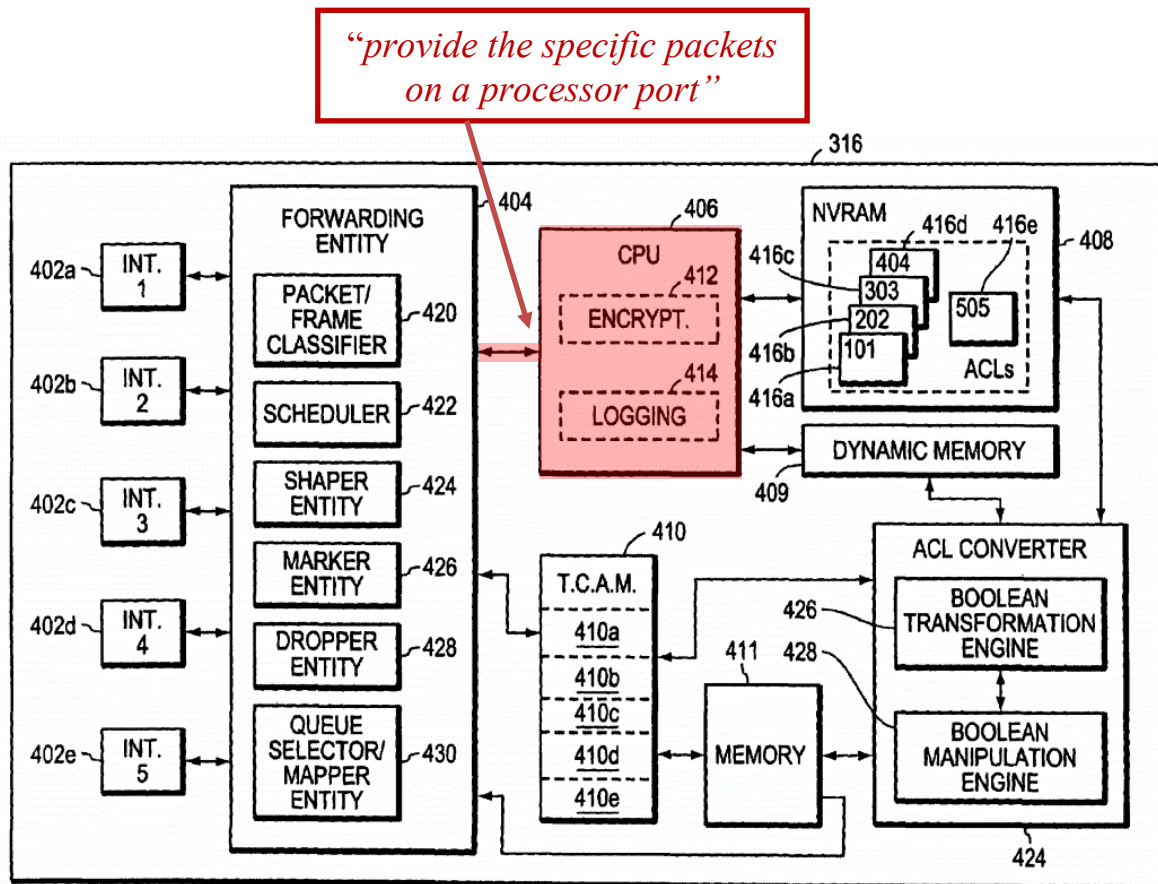
forwarding entity, which then takes the appropriate action (e.g., forward, discard, log and forward, **transfer to CPU for additional processing**, etc.)

Ex.1006, 4:29-36.

In addition, the forwarding entity 404 may not be capable of performing some actions specified by the ACLs. For example, if an encryption algorithm is only executable by the CPU 406 (as opposed to the forwarding entity 404 itself), then a permit action from an encryption ACL requires that the message be **forwarded to the CPU 406 for encryption**, assuming the corresponding action from the security ACL is also permit.

Ex.1006, 11:60-12:7; *see also* Ex.1006, 7:34-43 (“[E]ach ACL includes an action column 526 that corresponds to the particular action that is to be applied to network messages matching a corresponding ACE statement... a possible action may be to execute a particular program stored in the non-volatile or dynamic memory of the device.”). A POSITA would have understood that in certain circumstances (e.g., when a received packet matches ACL rules) the forwarding entity 404 would forward or transfer the specific packets to the CPU.

162. Gai’s Figure 4, reproduced below, illustrates with double-headed arrows that the forwarding entity 404 is operable to provide packets to a port of CPU 406.



Ex.1006, Fig. 4 (annotated).

163. Further, consistent with the analysis at [15.1.1], Kwan discloses that if a packet has a new IP address, a source IP address detector 236 of a port security processor 242 is used to learn the new IP address.

If it is not a previously learned source IP address, then it is treated as a new source IP address...If the maximum number of the source IP addresses are not present on the port, then a learn source IP routine is performed 314 (various methods related to this are described above)...If the learning and confirmation of the new source IP address

Ex.1003

is successful, then the new source IP address is stored 318 in a table indicating that the new source IP address is permitted access through the port.

Ex.1007, 8:17-47, Fig. 3; *see also* Ex.1007, 5:38-39 (“**The source IP address detector 236, can learn the source IP address** for a host in a number of different ways.”), 5:39-67 (disclosing various techniques that port security processor 242 uses to learn IP addresses). As discussed at [1.2.1], an example of a packet containing a new IP address is a DHCP response packet.

164. In implementing the combination of Gai and Kwan (see [15.1.1]), it would have been obvious to a POSITA to provide packets identified as having a new IP address on the port of Gai’s CPU 406 (consistent with Gai’s operation of forwarding/transferring), because this would allow for CPU 406 to learn new IP addresses and dynamically update the ACL. Ex.1007, 9:3-13 (“After the source IP address has been learned...then the [ACL] table is programmed....”). The updated ACL with the new IP address and MAC address pair can then be utilized to permit or restrict access to the network. Ex.1007, 8:17-47 (“...the source IP address, or addresses, which have previously been learned and stored in a table as source IP address which is permitted access to the network...”); Ex.1007, 4:34-39 (“The table which stores the IP address...can be implement using ...an access control list...”); *see also* Reasons to Combine Kwan with Gai.

165. Thus, Gai in combination with Kwan discloses that forwarding entity 404 is operable to provide the specific packets that need additional processing (e.g., encrypted, logged, etc., and packets that have a new IP address that has not been learned) to a port of CPU 406, which renders obvious that the forwarding circuit is operable “*to provide the specific packets on a processor port,*” as claimed.

d. [15.1.3] *further operable to receive packets on one of a plurality of ports including the processor port and*

166. Gai in combination with Kwan renders obvious this limitation.

167. First, consistent with the discussion at limitation [15.1.1], Gai’s forwarding entity 404 receives IP packets, which renders obvious that the forwarding entity is “*further operable to receive packets.*” See also Ex.1006, 1:37-2:41, 4:41-42, 5:14-22.

168. Second, Gai discloses that that the forwarding entity 404 receives the packets “*on one of a plurality of ports.*” For example, Gai’s forwarding entity 404 receives IP messages (packets) on a plurality of interfaces 402a-e corresponding to ports.

In operation, **a network message, such as IP message 100 (FIG. 1) is received at an interface (e.g., interface 402d) ...the forwarding entity 404 identifies the port (e.g., physical port number) on which the message 100 was received.**

Ex.1006, 16:63-17:6.

The switching function includes **receiving message frames at a source port** and transferring them to at least one destination port for receipt by another entity... **IP message packets** include a corresponding header which contains an IP source address and an IP destination address

Ex.1006, 2:1-14; *see also* Ex.1006, 6:27-30 (“...the forwarding entity 404... [switches] network messages among the various interfaces 402a-e of device 316.”); Ex.1006, 7:66-8:6 (“...each interface 402a-e per direction (e.g., inbound or outbound)...”).

169. As shown below at Gai’s Figure 4, the forwarding entity 404 (“*forwarding circuit*”) is operable to receive IP message packets (“*receive packets*”) as indicated by double-headed arrows on a plurality of interfaces 402a-e that correspond to the forwarding entity 404’s ports (“*one of a plurality of ports*”).

“receive packets on one of a plurality of ports”

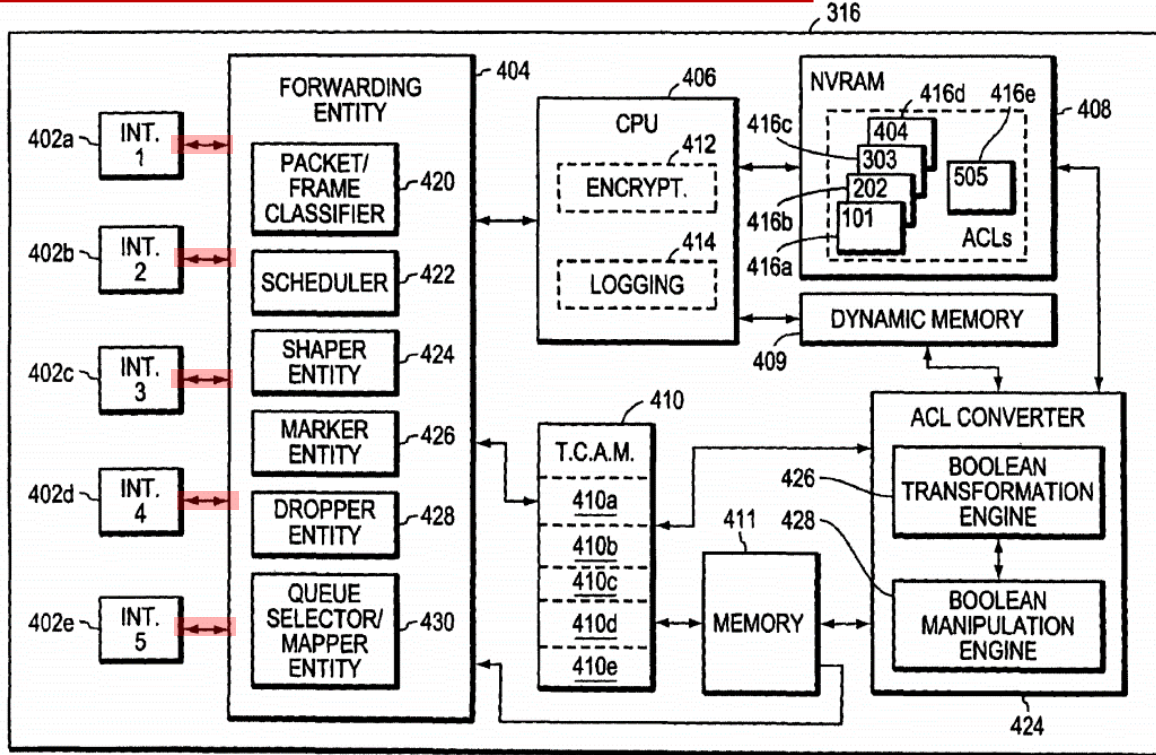


FIG. 4

Ex.1006, Fig. 4 (annotated).

170. Third, Gai in combination with Kwan renders obvious that the forwarding entity 404 (*“forwarding circuit”*) also receives packets on *“the processor port.”* As discussed at limitation [15.1.1], in the combination of Gai and Kwan the forwarding entity 404 forwards specific packets to CPU 406. Ex.1006, 6:13-30, 11:60-12:7; Ex.1007, 5:30-36, Fig. 3.

171. It would have been obvious to POSITA for Gai’s CPU 406 to send the packets back to the forwarding entity 404 as appropriate. A POSITA would have understood that the purpose of sending the packets to CPU 406 in one instance was

Ex.1003

to encrypt the packets for security before outputting the packets on a port. *See* Ex.1006, 8:36-49 (disclosing the implementation of “output security” of packets having a “permit action” by an “encryption function”). Similarly, Kwan’s packets with a new IP address are disclosed as being sent to the destination after successfully learning the IP address. Ex.1007, 8:43-55, Fig. 3 (“If the learning and confirmation of the new source IP address is successful, then...the data packet with new source IP packet can be passed 320 through the port.”). Accordingly, it would have been obvious to a POSITA for Gai’s CPU 406 to send the packets back to the forwarding entity 404 so that the packets may be transmitted to the destination.

172. This is consistent with Gai’s Figure 4, which illustrates with double-headed arrows that the forwarding entity 404 is operable to receive packets from CPU 406 (“*operable to receive packets on ... the processor port.*”).

“receive packets on... the processor port”

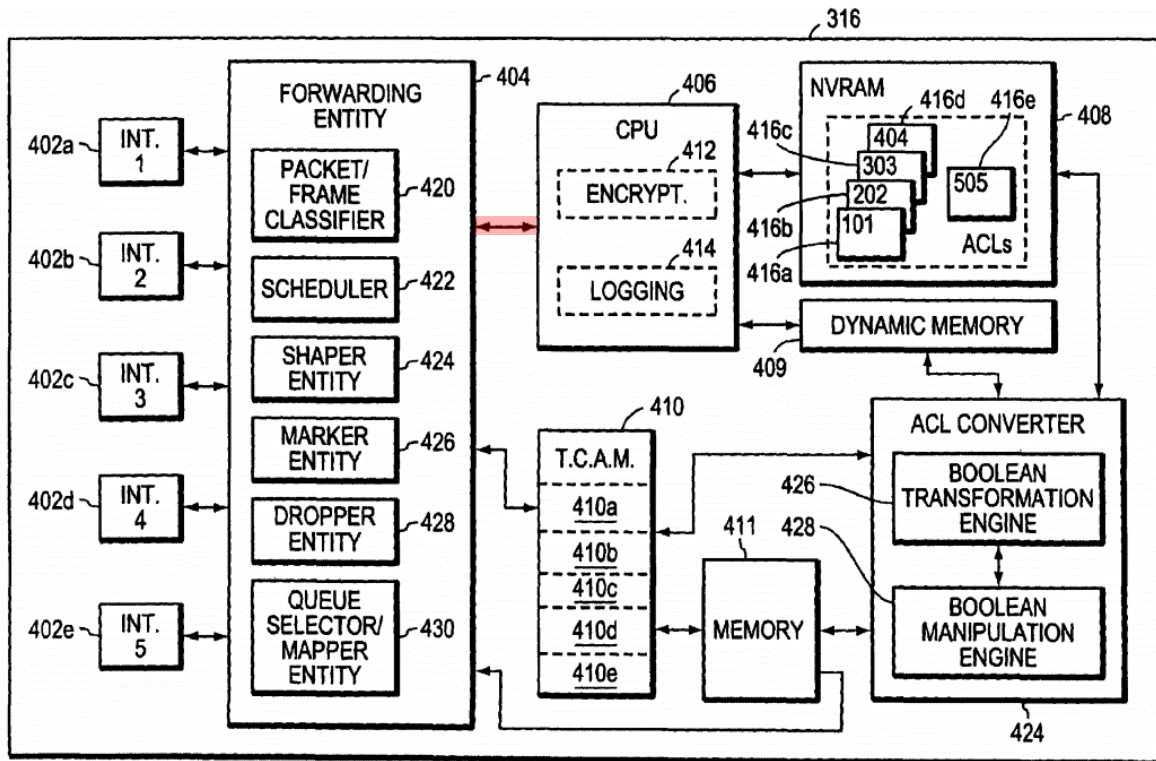


FIG. 4

Ex.1006, Fig. 4 (annotated).

173. Thus, Gai in combination with Kwan discloses that the forwarding entity 404 is operable to receive packets on one of a plurality of ports including CPU 406 port, which renders obvious that the forwarding circuit is *“further operable to receive packets on one of a plurality of ports including the processor port,”* as claimed.

- e. **[15.1.4] *to forward each received packet to a port corresponding to a destination address contained in the packet***

174. Gai in combination with Kwan renders obvious this limitation.

175. Gai's forwarding entity 404 is configured to forward each incoming packet to an output port corresponding to an IP destination address extracted from the packet.

As described below, the **forwarding entity 404 is basically configured to forward or switch network messages among the various interfaces 402a-e of device 316.**

Ex.1006, 6:27-30.

FIG. 4 is a partial block diagram of the intermediate network device 316. Device 316 preferably includes a plurality of interfaces 402a-402e that provide connectivity to the network 300. That is, interfaces 402a-402e are in communication with LANs 302 and 304 and Internet 318. **Each interface 402a-e, moreover, may be associated with one or more physical ports.**

Ex.1006, 5:60-66.

The switching function includes receiving message frames at a source port and **transferring them to at least one destination port for receipt by another entity...** In particular, certain layer 3 intermediate devices utilize access control lists to decide whether **received messages should be forwarded...based on certain predefined criteria... IP destination address....**

Ex.1006, 2:1-51, Fig. 1; Ex.1006, 19:6-14 (“**Forwarding entity, 404 also extracts the necessary information from the message (such as IP ... destination addresses, etc.)**”).

176. A POSITA would have understood that the forwarding entity 404 selects an interface and associated port for forwarding based on the destination address contained in the packet, as was common in the art. *See also* Ex.1007, 1:52-61, 2:2-7, 3:53-4:6; Ex.1008, 1:49-54.

177. As shown below at Gai’s Figure 4, the forwarding entity 404 (“*forwarding circuit*”) forwards (as indicated by double-headed arrows) each received packet to an interface 402*a-e* associated with a physical port corresponding to the IP destination address in the packet (“*forward each received packet to a port corresponding to a destination address contained in the packet*”).

“forward each received packet to a port”

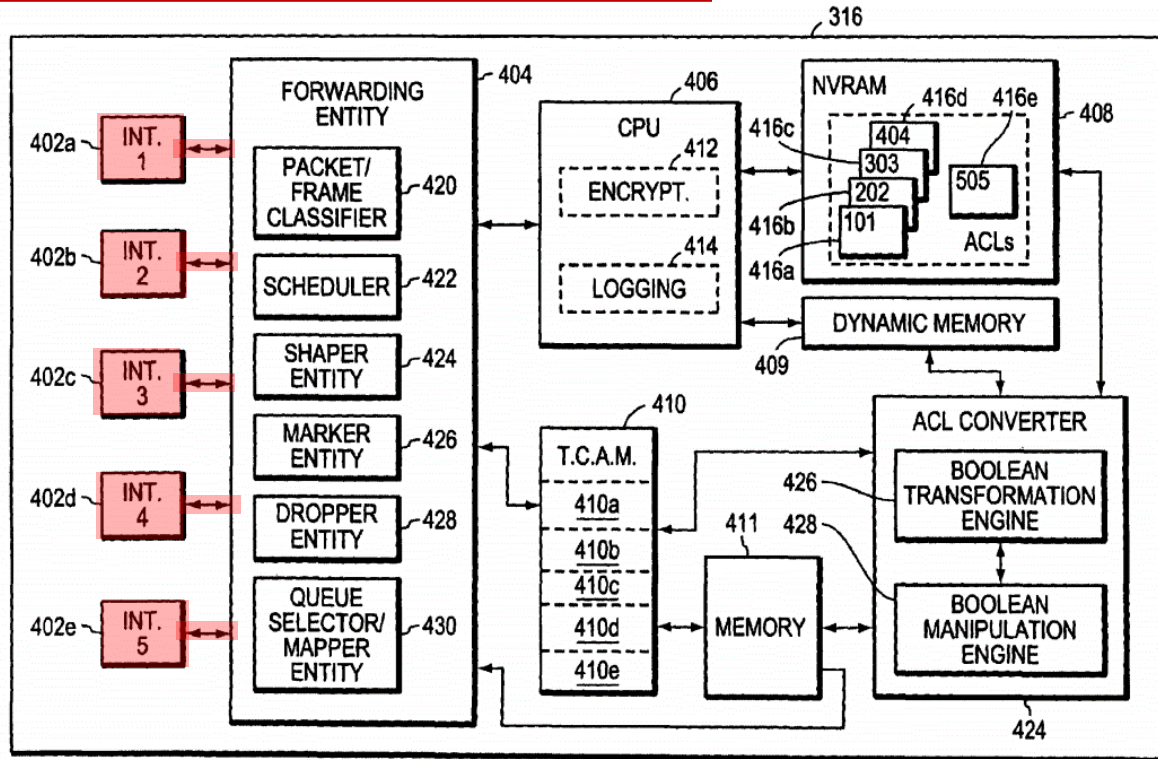


FIG. 4

Ex.1006, Fig. 4 (annotated).

178. Further, consistent with the analysis at limitation [15.1.3], in the instance where the packet is received via the CPU port, it would have been obvious to a POSITA for the packet to be forwarded to a port corresponding to the destination address in the packet.

179. Thus, Gai renders obvious that the forwarding circuit is operable “to forward each received packet to a port corresponding to a destination address contained in the packet,” as claimed.

f. [15.1.5] *subject to access restrictions contained in a dynamic access control list;*

180. Gai in combination with Yip and Kwan renders obvious this limitation.

181. **First**, as discussed at limitations [1.0] and [1.3], Gai in combination with Yip and Kwan discloses dynamically generating an optimized TCAM ACL, which renders obvious “*a dynamic access control list.*”

182. **Second**, Gai discloses that the forwarding entity 404 forwards packets “*subject to access restrictions contained*” in the TCAM ACL. Gai explains that the “ACL represent the specific criteria with which network messages are compared” to determine if access should be restricted (e.g., “deny”) or if forwarding is permitted (“permit”). Ex.1006, 7:24-39; *see also*; 9:31-36 (“[The] statement of an ACL is basically a function written in an ‘if condition, then action’ format. The condition statement, moreover, corresponds to the particular criteria (e.g., IP SA=2.5.4.x and Source Port=100) and the action corresponds to the ACL output (e.g., permit, deny, etc.)”); Figs. 5A-E, 8; *see also* Ex.1006, 17:40-48.

183. During operation, the TCAM ACL is applied to packets.

The information from the message fields being tested may then be placed in a Comparand register (not shown) by the TCAM 410. **The TCAM 410 then proceeds to compare the contents** of the Comparand register with the cells of each row of the identified section simultaneously (i.e., with the ACE statements of the single, unified

ACL). When a matching row is detected, its address is supplied to memory 411, thereby identifying a corresponding space or location therein. **As described above, the corresponding space or location in memory 411 contains the action specified by the matching row of the TCAM 410, as loaded in memory 411 by the ACL converter 424. The specified action is then provided to the forwarding entity 404, which, in turn, implements it.** For example, based on the IP SA of the message, the action may be drop. In response, forwarding entity 404 simply discards the message. As described above, other possible actions include switch to destination interface, apply QoS and switch to destination interface, forward to CPU 406 for additional processing, forward to second forwarding or processing entity, etc.

Ex.1006, 17:28-48; *see also* 7:37-39 (“In the preferred embodiment, permissible actions include permit, deny, permit and log, and deny and log.”); 8:7-27 (“If a match is made with an ACE statement having a ‘permit’ action (e.g., ACE 502), the packet is forwarded. If a match is made with an ACE statement having a ‘deny’ action (e.g., ACE 506), the packet is dropped.”); 4:26-30 (“In particular, upon receipt of a packet at a first interface, a forwarding entity at the intermediate network device tests the packet against the single, unified ACL stored in the corresponding portion of the CAM. When a match is obtained, the corresponding decision is returned to the forwarding entity, which then takes the appropriate action

(e.g., forward, discard, log and forward, transfer to CPU for additional processing, etc.)”); Ex.1006, Figures 5A-E, 10A-11.

184. Each of Gai’s actions that restricts access (e.g., “forward,” “discard,” “permit,” “deny,” etc.), separately and in combination, correspond to “*access restrictions*” as claimed.

185. Furthermore, consistent with the analysis at limitation [1.3], Kwan discloses “**an access control list, which are data fields, included in a content addressable memory 240**, which is referred to as an ACL CAM. ...[to] **control access and switching through the ports.**” Ex.1007, 4:34-48. For example, the source IP address of received packets is compared to the ACL-CAM and if the IP address is found, the packets are forwarded to the destination. Ex.1007, 4:34-48, 8:14-29. On the other hand, packets with source IP addresses not in ACL-CAM are “blocked,” until the new IP address can be successfully learned. Ex.1007, 8:29-46, 9:4-16, Figs. 3, 4. The ACL’s action of blocking corresponds to “*access restrictions*” as claimed.

186. In the proposed combination, when packets are identified as having new IP address (per Kwan, see also [15.1.1]), it would have been obvious to a POSITA to learn the new IP address and dynamically update Gai’s TCAM ACL with the learned IP address. Ex.1006, 18:65-19:3; Ex.1007, 1:15-17, 6:66-7:2, 9:3-13. Kwan’s teachings as applied to Gai, for example, would allow for learning a

Ex.1003

new IP address while blocking packets originating from “false, or spoofed, source IP address[es],” thereby providing security. Ex.1007, 4:56-63, Abstract; *see also* Ex.1007, 7:61-66 (“[I]t is important...that the table containing the learned source IP addresses be dynamic, such that the table can be updated so that...new learned source IP addresses can be added to the table.”); Reasons to Combine Kwan with Gai.

187. Thus, Gai in combination with Yip and Kwan discloses that the forwarding is subject to access restrictions contained in a dynamically generated TCAM ACL, which renders obvious that the forwarding is “*subject to access restrictions contained in a dynamic access control list*,” as claimed.

g. [15.2.1] *a memory circuit coupled to the forwarding circuit*,

188. Gai renders obvious this limitation.

189. Gai discloses “*a memory circuit coupled to the forwarding circuit*” by disclosing NVRAM 408, dynamic memory 409, and TCAM 410 that are coupled to the forwarding entity 404 via CPU 406.

Device 316 further includes at least one forwarding entity 404, a central processing unit (CPU) 406, **non-volatile random access memory (NVRAM) 408, dynamic memory 409** and at least one content addressable or associative memory 410. The associative memory 410 is preferably a ternary content addressable memory (TCAM), and its

cells (not shown) may be associated with or assigned one of three values, such as “0”, “1” or “don't care”. Associated with and coupled to the **TCAM 410** is another memory device 411, such as a dynamic random access memory. Memory 411 preferably includes a plurality of spaces or locations, such that each entry or row of the TCAM 410 has a corresponding location or space in memory 411.

Ex.1006, 5:65-6:12. Gai's **“forwarding entity 404 is...coupled to CPU 406”**

which **“is coupled to NVRAM 408...and also to dynamic memory 409.... The forwarding entity 404 is also coupled to...the TCAM 410.”** Ex.1006, 6:13-30.

NVRAM 408, Dynamic Memory 409, and TCAM 410 are
coupled to the Forwarding Entity 404
“a memory circuit coupled to the forwarding circuit”

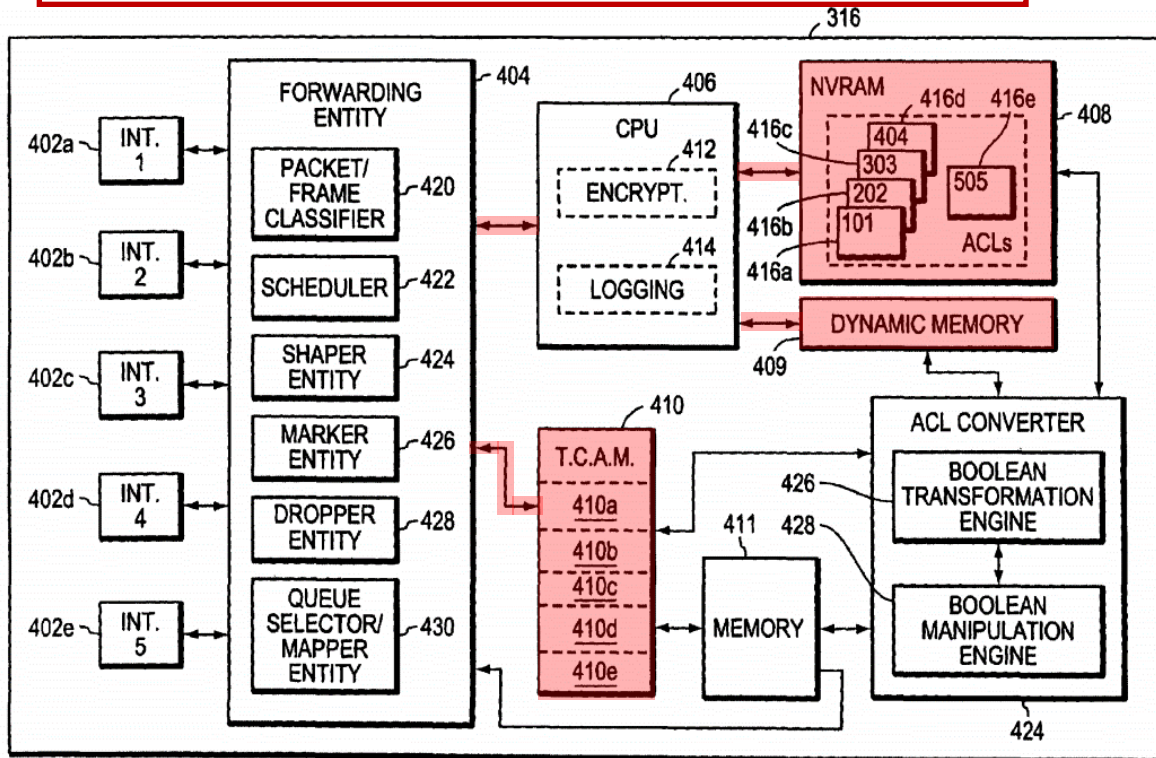


FIG. 4

Ex.1006, Fig. 4 (annotated).

190. A POSITA would have understood that because Gai's NVRAM 408 and dynamic memory 409 are coupled to CPU 406, which is coupled to the forwarding entity 404, both NVRAM 408 and dynamic memory 409 are coupled, via CPU 406, to the forwarding entity 404. I note that Gai's disclosure is within the scope of claim 15 of the '518 patent, as confirmed by dependent claim 16, which

recites that “*the processor further comprises a direct memory access controller coupled **between** the forwarding engine and the memory.*”

191. Thus, Gai discloses a memory circuit (e.g., NVRAM 408, dynamic memory 409, and TCAM 410) that is coupled to the forwarding entity 404, which renders obvious “*a memory circuit coupled to the forwarding circuit,*” as claimed.

h. [15.2.2] the memory circuit operable to store packets and

192. Gai in combination Georgiou renders obvious this limitation.

193. **First**, as discussed at limitation [15.2.1], Gai’s dynamic memory 409 corresponds to part of the claimed “*memory circuit.*”

194. **Second**, the further combination with Georgiou renders obvious that Gai’s dynamic memory is “*operable to store packets.*” For example, Georgiou teaches that typically IP packets are stored in memory while awaiting processing. According to Georgiou, “[t]ypical arrangements include...receive and temporarily store the incoming packets in a memory...until subsequent processing can access and retrieve the packets, as appropriate, for action on the packet contents.” Ex.1008, 1:22-27; *see also* Ex.1008, 3:17-22 (“...**the packet is typically...transferred to memory to await subsequent processing.**”); Ex.1008, 1:13-21 (“Internet Protocol (IP)...packets”). Georgiou’s memory is dynamic memory. Ex.1008, 5:51-67.

195. Consistent with the analysis at limitations [15.1.2]-[15.1.3], it would have been obvious for packets to be provided to the port of CPU 406 for processing. Ex.1006, 17:44-48 (“...forward to CPU 406 for addition processing...”). Further, it would have been obvious to a POSITA, in view of Georgiou, to store in Gai’s dynamic memory 409 (analogous to the dynamic memory of Georgiou) packets forwarded to CPU 406 for processing, until CPU 406 has sufficient processing bandwidth. Storing the packets in dynamic memory 409 would allow for CPU 406 to access and retrieve the stored packets, as appropriate, and take action on the packet contents (e.g., encrypt, learn new IP address, dynamically generate updated BDD ACLs, etc.), without dropping packets for lack of processing bandwidth. Ex.1007, 5:22-6:30; Ex.1008, 9:7-12; *see also* Reasons to Combine Georgiou with Gai.

196. Thus, Gai in combination with Georgiou renders obvious storing packets (e.g., packets that need to be processed and those that have a new IP address) in dynamic memory 409, which renders obvious that “*the memory circuit [is] operable to store packets,*” as claimed.

i. [15.2.3] [the memory circuit...] *operable to store an enhanced access control list and*

197. Gai in combination with Yip and Kwan renders obvious this limitation.

198. First, as discussed at limitation [1.1], Gai's one or more text-based ACLs, which include MAC addresses and seed IP address properties, per the combination with Yip and Kwan, renders obvious "*an enhanced access control list.*"

199. Second, as discussed at limitation [15.2.1], Gai's NVRAM 408 corresponds to part of the claimed "*memory circuit.*" Further, Gai discloses that the "**nonvolatile random access memory (NVRAM) [is used] for initially storing the one or more text-based ACLs.**" *See also* Ex.1006, 2:58-65, 4:8-13; *see also* 6:13-18, 7:15-8:6, 9:18-25.

200. Thus, Gai in combination with Yip and Kwan discloses that NVRAM 408 is operable to store administrator generated text-based ACLs (including MAC addresses and seed IP address properties), which renders obvious that the memory is "*operable to store an enhanced access control list,*" as claimed.

j. [15.2.4] [the memory circuit...operable to store]...a dynamic access control list; and

201. Gai in combination with Yip and Kwan renders obvious this limitation.

202. First, as discussed at limitations [1.0] and [1.3], Gai in combination with Yip and Kwan discloses dynamically generating an optimized TCAM ACL, which renders obvious "*a dynamic access control list.*"

203. Second, as discussed at limitation [15.2.1], Gai's TCAM 410 corresponds to part of the claimed "*memory circuit*." Further, Gai discloses that the "ACL converter 424 is operatively coupled to...TCAM 410 and its associated memory 411 for **storing modified versions of the ACLs**." *See also* Ex.1006, 4:26-29 ("With a single, unified ACL defined per interface per direction and **stored in a CAM-type memory**, the intermediate network device is able to rapidly evaluate network messages."); Ex.1006, 20:65-67 ("[S]toring the single, unified ACL at a portion of the CAM that corresponds to the given interface.").

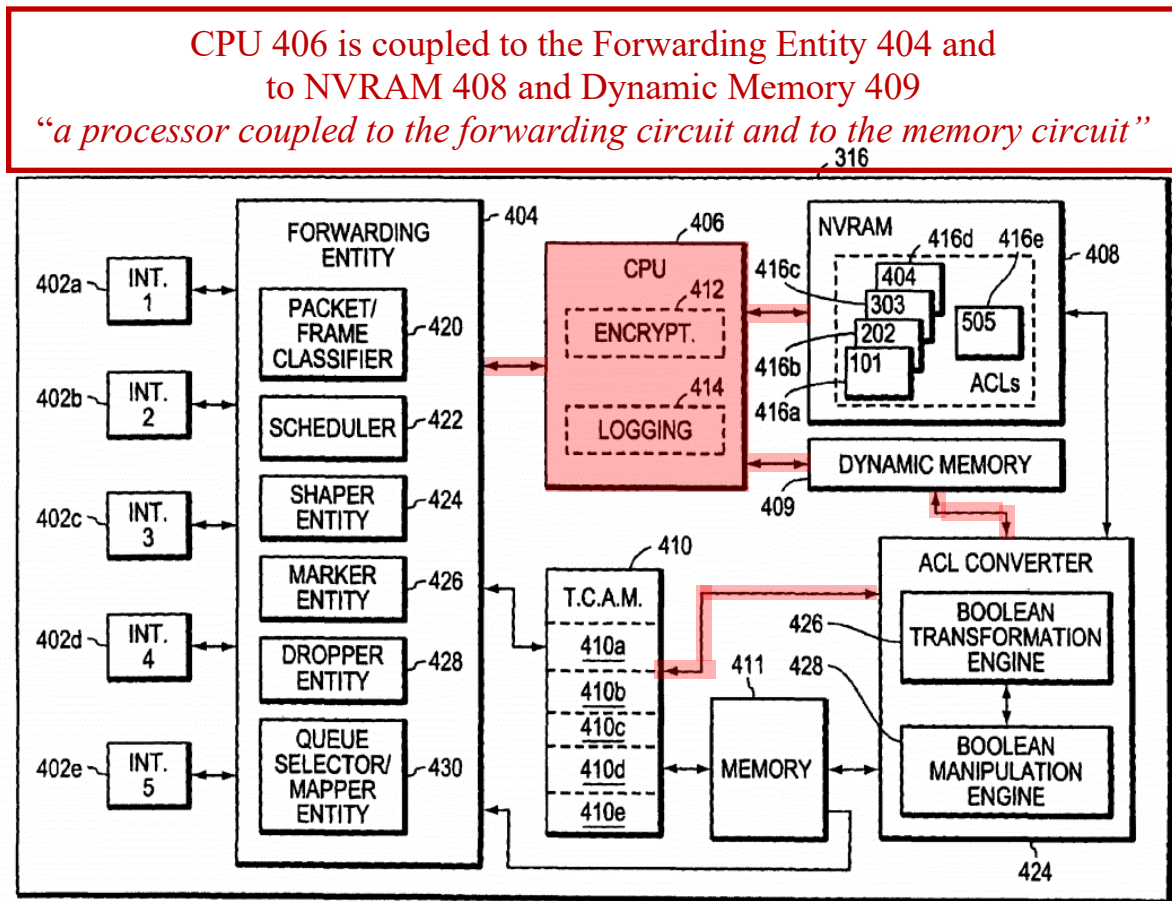
204. Thus, Gai in combination with Yip and Kwan discloses that the TCAM 410 is operable to store a dynamically generated optimized ACL, which renders obvious that the memory is operable to store "*a dynamic access control list*," as claimed.

k. [15.3.1] *a processor coupled to the forwarding circuit and to the memory circuit*,

205. Gai discloses this limitation.

206. As discussed at limitations [15.1.2] and [15.2.1], Gai's forwarding entity 404 ("*forwarding circuit*") is coupled to CPU 406 ("*a processor*") that is also coupled to NVRAM 408 and dynamic memory 409 ("*memory circuit*"). *See* Ex.1006, 6:13-27 ("CPU 406... is **coupled** to NVRAM 408... and also to dynamic memory 409.... The forwarding entity 404 is also **coupled** to the CPU 406.").

207. Additionally, CPU 406 is coupled to TCAM 410 via the ACL converter 424. Ex.1006, 6:36-41 (“Device 316 further includes an access control list (ACL) converter 424. **ACL converter 424 is operatively coupled to ...TCAM 410** and its associated memory 411 for storing modified versions of the ACLs 416.”). Gai’s “**ACL converter 424...may run on or otherwise be executed by CPU 406.**” Ex.1006, 6:52-55. It would have been obvious to a POSITA for the CPU 406 to be coupled to the TCAM 410, because the ACL converter (which is coupled to and stores ACLs in TCAM 410) runs on CPU 406.



Ex.1006, Fig. 4 (annotated).

208. Thus, Gai discloses that CPU 406 is coupled to the forwarding entity 404 and to NVRAM 408, dynamic memory 409, and TCAM 410, which renders obvious “a processor coupled to the forwarding circuit and to the memory circuit,” as claimed.

- 1.** [15.3.2] *the processor operable to define the specific packets detected by the forwarding circuit and*

209. Gai in combination with Kwan renders obvious this limitation.

210. First, as discussed at limitations [15.1.1] and [15.1.4], Gai in combination with Kwan discloses that forwarding entity 404 identifies specific received packets that need additional processing (e.g., encrypted, logged, etc., and packets that have a new IP address that has not been learned), which renders obvious “*the specific packets detected by the forwarding circuit.*”

211. Second, as discussed at limitations [1.0] and [1.3], Gai in combination with Yip and Kwan discloses that Gai’s CPU 406 executes ACL converter 424 software to dynamically generate an optimized TCAM ACL from one or more text-based ACLs. Specifically, Gai discloses that “ACL converter 424, including boolean transformation engine 426 and boolean manipulation engine 428, may run on or otherwise be executed by CPU 406.” Ex.1006, 6:52-55. The “ACL converter 424 transforms...text-based ACLs into corresponding boolean Binary Decision Diagram (‘BDD’) representations.” Ex.1006, 9:4-27, Fig. 4; *see also* Ex.1006, 9:42-10:26, Fig. 6. “**The ACL converter 424 then merges all of the ACLs assigned to each interface into a single, unified ACL and stores these single, unified ACLs in the TCAM 410.**” Ex.1006, 9:4-27; *see also* Ex.1006, 10:27-11:4, 12:20-16:17, Figs. 7, 9A-9B.

212. A POSITA would have understood that the generated unified ACL sets forth how packets should be processed by the forwarding engine 404. By executing ACL converter 424 software to dynamically generate the ACL for TCAM, the CPU

Ex.1003

406 “*define[s] the specific packets*” that the forwarding engine 404 identifies for additional processing. *See*, analysis at limitation [15.1.1]. Additionally, or alternatively, the CPU 406 “*define[s] the specific packets*” that the forwarding engine 404 identifies for additional processing by storing the unified ACL in the TCAM.

213. Thus, Gai in combination with Yip and Kwan discloses that CPU 406 is operable to sets forth the specific packets identified by the forwarding engine 404, by generating a unified ACL and storing the unified ACL in TCAM, which renders obvious that “*the processor [is] operable to define the specific packets detected by the forwarding circuit,*” as claimed.

- m. **[15.3.3] [the processor...] operable to process the specific packets stored in the memory circuit using the enhanced access control list to generate the dynamic access control list and**

214. Gai in combination with Yip, Kwan, and Georgiou renders obvious this limitation. **First**, as discussed at limitation [15.2.2] Gai in combination with Georgiou discloses storing packets for processing in dynamic memory 409, which renders obvious “*the specific packets stored in the memory circuit.*” Further, as discussed at limitation [1.1], Gai’s one or more text-based ACLs, which include MAC addresses and seed IP address properties, per the combination with Yip and Kwan, renders obvious “*an enhanced access control list.*” Furthermore, as

discussed at limitations [1.0] and [1.3], Gai in combination with Yip and Kwan discloses dynamically generating an optimized TCAM ACL from one or more text-based ACLs, which renders obvious “*a dynamic access control list.*”

215. Second, Georgiou discloses that “**process[or] can access and retrieve the packets [from memory], as appropriate, for action on the packet contents.**”

Ex.1008, 1:22-27; *see also* Ex.1008, 3:17-22 (“...the packet is typically... transferred to memory to await subsequent processing.”). Further, consistent with the analysis at [15.1.1]-[15.1.2], Kwan discloses that port security processor 242 is used to learn new IP addresses. Ex.1007, 4:56-58, 5:38-67, 8:17-47, Fig. 3.

216. In view of Georgiou’s teachings, it would have been obvious to a POSITA for Gai’s CPU 406 to retrieve the packets stored in its dynamic memory 409 that need additional processing, including packets that have a new IP address that has not been learned, per Kwan. See analysis at limitation [15.1.1]. That is, it would have been obvious to a POSITA for the CPU 406 to retrieve from dynamic memory 409 packets with a new IP address and process these packets per Kwan’s DHCP technique or RARP technique to learn the new IP address. *See also* analysis at limitation [1.2.4]. During the learning process, the CPU 406 would utilize seed IP address properties in one or more text-based ACLs and only match a new IP address with a MAC address that is consistent. See analysis at limitations [1.1] and [1.2.4]; Ex.1007, 6:14-25. In the instance where the new IP address matches the

Ex.1003

seed IP address properties, then the new IP address would be learned and stored in the one or more text-based ACLs. See analysis at limitation [1.1]. CPU 406 would then use the one or more text-based ACLs (including the newly learned IP address) to dynamically generate an optimized TCAM ACL. See analysis at limitations [1.0] and [1.3]. *See* Reasons to Combine Kwan with Gai.

217. Thus, Gai in combination with Yip, Kwan, and Georgiou discloses that CPU 406 is operable to process the specific packets (e.g., packets that have a new IP address) stored in dynamic memory 409 using one or more text-based ACLs (e.g., using MAC addresses and seed IP address properties) to dynamically generate an updated TCAM ACL, which renders obvious that the processor is “*operable to process the specific packets stored in the memory circuit using the enhanced access control list to generate the dynamic access control list,*” as claimed.

n. [15.3.4] [the processor... operable to...] *store the dynamic access control list in the memory circuit, and*

218. Gai renders obvious this limitation.

219. First, as discussed at limitations [15.2.1] and [15.2.4], Gai’s TCAM 410 stores the dynamically generated optimized ACL, which renders obvious “*store[ing] the dynamic access control list in the memory circuit.*”

220. Second, as discussed at limitations [15.1.3] and [15.3.3], Gai’s CPU 406 generates the TCAM ACL by executing software of ACL converter 424,

among other operations. Ex.1006, 6:52-59, 9:22-10:48. Gai further discloses “**executable program instructions...[for] storing the ACL corresponding to the second boolean representation at the memory device.**” Ex.1006, 22:15-29. In certain embodiments, “**the memory device is a content addressable memory (CAM).**” Ex.1006, 20:38-53. It would have been obvious to a POSITA for Gai’s CPU 406, which operates by executing program instructions, to execute the program instructions to store the dynamically generated optimized ACL in TCAM 410.

221. Thus, Gai discloses that CPU 406 is operable to store the dynamically generated ACL in TCAM 410, which renders obvious that the processor is operable to “*store the dynamic access control list in the memory circuit,*” as claimed.

- o.** [15.3.5] [the processor...] *further operable to provide the specific packets to the processor port of the forwarding circuit after processing the packets.*

222. Gai in combination with Kwan renders obvious this limitation.

223. First, as discussed at limitation [15.1.3], Gai in combination with Kwan discloses that the forwarding entity 404 is operable to receive packets on one of a plurality of ports including CPU 406 port. As also discussed at limitation [15.1.3], it would have been obvious to POSITA for Gai’s CPU 406 to provide the packets after encryption and learning a new IP addresses (among other packet processing operations) to its port so that the packet may be sent to the destination.

Gai's Figure 4 is consistent with this understanding, illustrating double-headed arrows between CPU 406 and the forwarding entity 404.

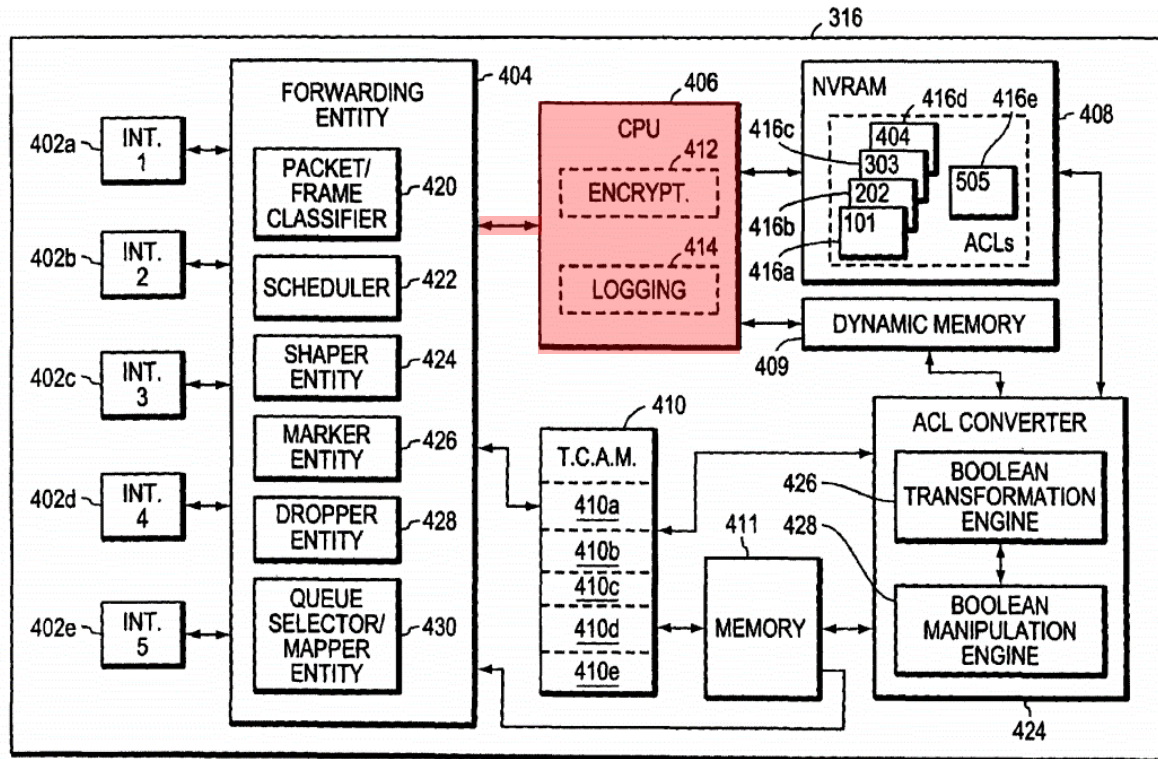
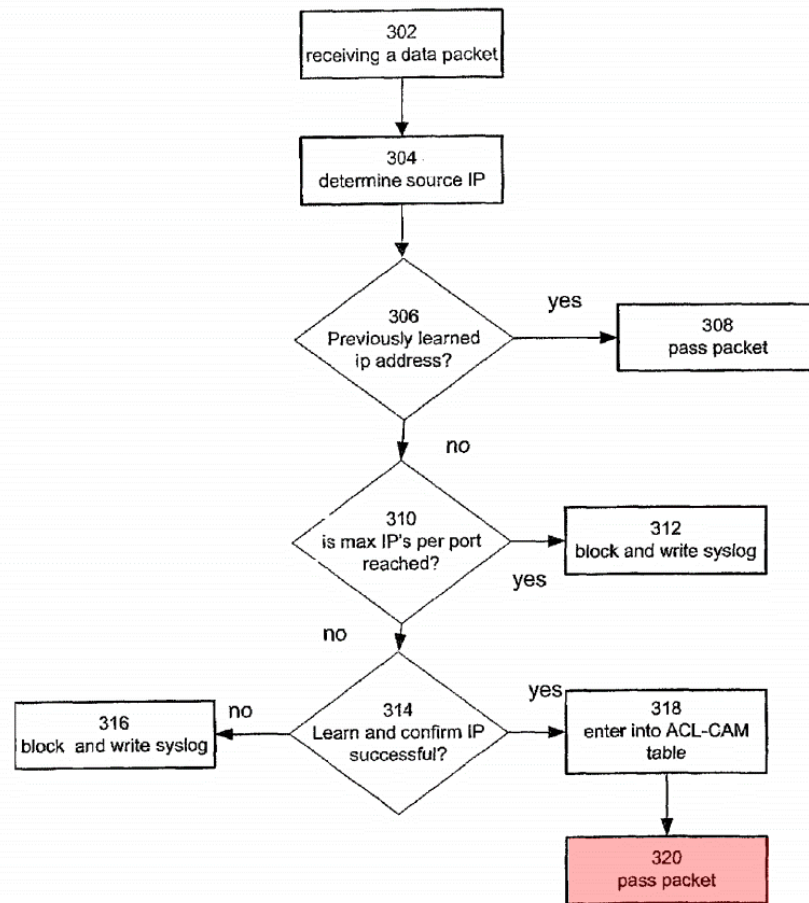


FIG. 4

Ex.1006, Fig. 4 (annotated).

224. **Second**, consistent with this understanding, Kwan in the context of Figure 3 teaches that packets are passed (320) to a destination port once the IP address has been learned (314).

Fig. 3



Ex.1007, Fig. 3 (annotated).

225. Accordingly, in implementing the combination with Kwan, it would have been obvious to a POSITA for Gai's CPU 406 to provide packets, after successfully learning a new IP address, to the forwarding entity 404 port so that the packets may be passed to the destination device. *See also* Reasons to Combine Kwan with Gai.

226. Thus, Gai in combination with Kwan discloses that CPU 406 is operable to provide the specific packets to port of CPU 406 after processing (e.g.,

after packets were encrypted and after packets were processed to learn the new IP address), which renders obvious that the processor is “*further operable to provide the specific packets to the processor port of the forwarding circuit after processing the packets,*” as claimed.

4. Claim 17

- a. [17.1] *The network switch of claim 15 wherein the switch comprises an Ethernet switch and wherein the packets comprise Ethernet packets.***

227. As discussed at limitations [15.0]-[15.1.1], the prior art discloses that the network device 316 supports the switching of Ethernet packets on computer network 300. *See* Ex.1006, 5:3-59; *see also* Ex.1006, 5:60-64 (“**Device 316...provide[s] connectivity... with LANs 302 and 304 and Internet 318**”); 2:14-16 (“**[S]witches may re-assemble or convert received data frames from one LAN standard (e.g., Ethernet) to another (e.g. Token Ring).**”). It would have been obvious to a POSITA that because Gai’s network device 316 supports processing of Ethernet LAN packets, it corresponds to “*an Ethernet switch.*” Also, it would have been obvious to a POSITA that the “received data frames,” which are formatted in accordance with the LAN Ethernet standard, correspond to “*Ethernet packets.*”

228. I note that Gai’s disclosure of Ethernet LAN is consistent with the ’518 patent’s embodiment. *See* Ex.1001, 1:5-24 (“In an Ethernet local area network (LAN), computers or hosts are attached to the network and each host is uniquely

identified by a physical address which is designated a media access control (MAC) address in an Ethernet network...a switch receives an Ethernet packet from a source host.”).

229. Thus, Gai’s disclosure that the network device 316 comprises an Ethernet LAN switch that processes Ethernet data frames, renders obvious that “*the switch comprises an Ethernet switch and wherein the packets comprise Ethernet packets.*”

5. Claim 19

a. [19.0] *A computer network, comprising:*

230. To the extent the preamble is limiting, as discussed at limitation [15.0], Gai discloses a computer network 300, which discloses “[*a*] computer network.”

b. [19.1.1] *a network switch, including,*

231. As discussed at limitations [1.2.1] and [15.0], Gai discloses a network device 316 that includes circuitry for switching, which renders obvious “*a network switch.*”

c. [19.1.2] *a forwarding circuit operable to detect specific received packets and*

232. See analysis at limitation [15.1.1].

a. [19.1.3] *to provide the specific packets on a processor port, and*

233. See analysis at limitation [15.1.2].

- b. **[19.1.4] *further operable to receive packets on one of a plurality of ports including the processor port and***
234. See analysis at limitation [15.1.3].
- c. **[19.1.5] *to forward each received packet to a port corresponding to a destination address contained in the packet***
235. See analysis at limitation [15.1.4].
- d. **[19.1.6] *subject to access restrictions contained in a dynamic access control list;***
236. See analysis at limitation [15.1.5].
- e. **[19.2.1] *a memory circuit coupled to the forwarding circuit,***
237. See analysis at limitation [15.2.1].
- f. **[19.2.2] *the memory circuit operable to store packets and***
238. See analysis at limitation [15.2.2].
- g. **[19.2.3] *[the memory circuit]...operable to store an enhanced access control list and***
239. See analysis at limitation [15.2.3].
- h. **[19.2.4] *[the memory circuit...operable to store]...a dynamic access control list; and***
240. See analysis at limitation [15.2.4].
- i. **[19.3.1] *a processor coupled to the forwarding circuit and to the memory circuit,***
241. See analysis at limitation [15.3.1].

- j. [19.3.2] *the processor operable to define the specific packets detected by the forwarding circuit and*

242. See analysis at limitation [15.3.2].

- k. [19.3.3] *operable to process the specific packets stored in the memory circuit using the enhanced access control list to generate the dynamic access control list and*

243. See analysis at limitation [15.3.3].

- l. [19.3.4] [the processor... *operable to...*] *store the dynamic access control list in the memory circuit, and*

244. See analysis at limitation [15.3.4].

- m. [19.3.5] [the processor...] *further operable to provide the specific packets to the processor port of the forwarding circuit after processing the packets; and*

245. See analysis at limitation [15.3.5].

- n. [19.4] *a plurality of hosts, each host coupled to a respective port of the network switch.*

246. Gai renders obvious this limitation.

247. As an initial matter, a “host” in the ’518 patent “includes any type of electronic device that may be coupled to the network, such as a computer system, IP telephone, or personal digital assistant (PDA).” Ex.1001, 2:37-40.

248. **First**, as discussed at limitations [15.0] and [15.1.4], Gai’s network device 316 includes ports, which discloses a “*a respective port of the network switch.*”

249. **Second**, Gai discloses that the “[a] plurality of end stations [] 306-312, and servers, such as servers 313 and 314, may be coupled to LANs 302, 304” and that “LANs 302 and 304 may also be interconnected by an intermediate **network device 316.**” Ex.1006, 5:3-13. Gai’s network device 316 utilizes ports for receiving data from the external end stations. Ex.1006, 2:1-3 (“The switching function includes receiving message frames at a **source port** and transferring them to at least one **destination port** for receipt by another entity.”), 17:4-6 (“...the forwarding entity 404 identifies the port (e.g., physical port number) on which the message 100 was received.”); 5:60-66 (“...interfaces 402*a*-402*e* are in communication with LANs 302 and 304 and Internet 318. Each interface 402*a-e*, moreover, may be associated with one or more physical ports...”).

250. As illustrated below at Figure 3, each of Gai’s end stations 306-312 are coupled, via LANs 320, 304, to the network device 316. It would have been obvious to a POSITA for each end station to be coupled to one of the physical interfaces 402*a*-402*e* of the network device 316 because that is how computers were most commonly connected to a network switch. That understanding is also consistent with the disclosure in Kwan, which describes how “[e]nd user host devices...such as personal computers can be coupled to these ports.” Ex.1007, 3:46-50.

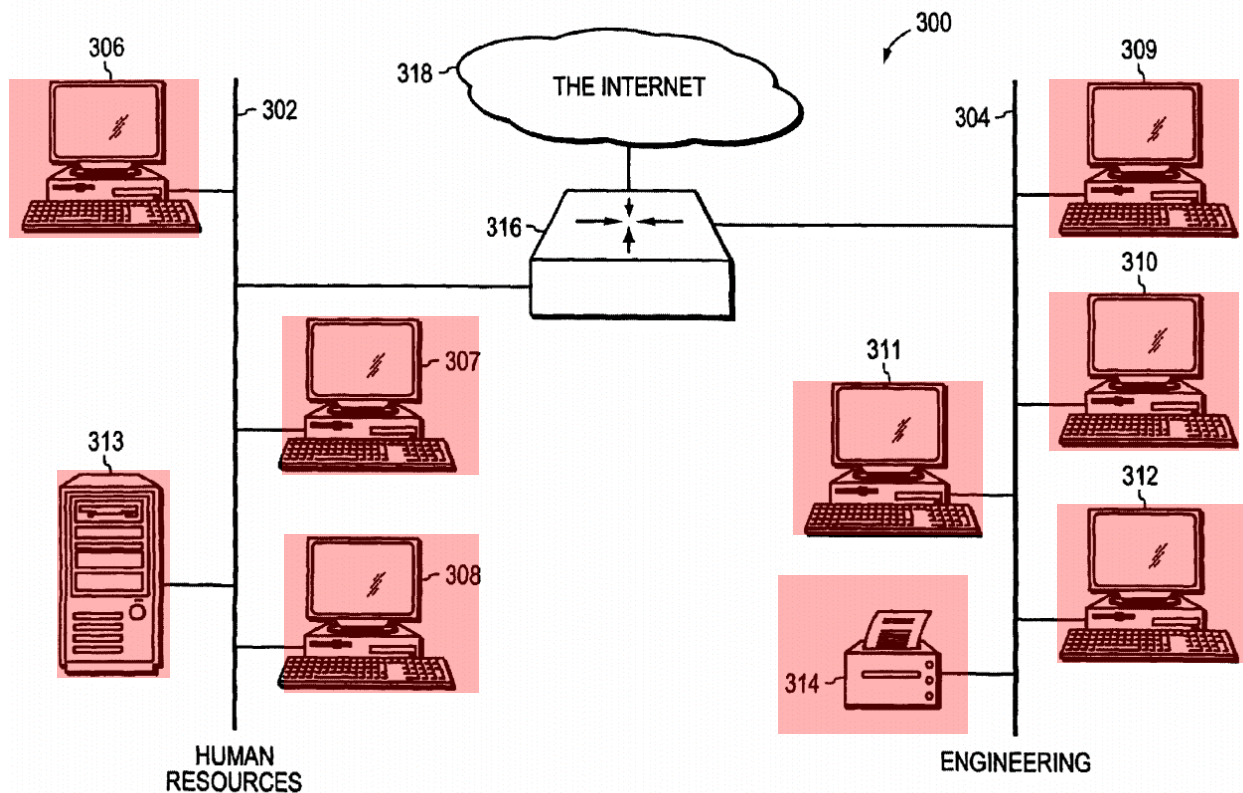


FIG. 3

Ex.1006, Fig. 3 (annotated).

251. A POSITA would have understood that Gai's end stations, illustrated as personal computers 306-312, server 313, and printer 314, correspond to "*hosts*" as that term is used in the '518 patent. *See e.g.*, Ex.1001, 2:37-40. Also, a POSITA would have understood that coupling via a LAN is consistent with the '518 patent's disclosure. *See* Ex.1001, 1:5-21 ("**In an Ethernet local area network (LAN), computers or hosts are attached to the network** and each host is uniquely identified by a physical address which is designated a media access control (MAC)

address in an Ethernet network...**a switch receives an Ethernet packet from a source host.**”).

252. Thus, Gai (alone or in combination with Kwan) discloses a plurality of end stations (e.g., any combination of 306-314), each coupled to a respective port of the network device 316, which renders obvious “*a plurality of hosts, each host coupled to a respective port of the network switch.*”

6. Claim 20

- a.** [20.1] *The computer network of claim 19 wherein at least some of the hosts comprise personal computer systems.*

253. Gai alone and in combination with Kwan renders obvious that at least some of the hosts comprise personal computer systems.

254. First, as discussed at limitation [19.4], at least some of Gai’s end stations, namely 306-312, are illustrated as personal computers, which renders obvious that “*at least some of the hosts comprise personal computer systems.*”

255. Second, to the extent argued that Gai’s end stations 306-312 do not sufficiently disclose “*personal computer systems,*” Kwan expressly discloses that “[e]nd user host devices...such as **personal computers** can be coupled to these ports.” Ex.1007, 3:46-50. It would have been obvious to a POSITA to implement at least some of Gai’s end stations as personal computers, in view of Kwan, because such an implementation would allow for users to use their personal computers and

access or exchange content with other users of other personal computers or with other network devices, e.g., send a print request to end station 314, illustrated as a printer.

256. Thus, Gai in combination with Kwan discloses that at least some of Gai's end stations comprise personal computers, which renders obvious that "*at least some of the hosts comprise personal computer systems.*"

7. Claim 21

- a.** [21.1] *The computer network of claim 19 wherein the network comprises an Ethernet network, and wherein the switch comprises an Ethernet switch and the packets comprise Ethernet packets.*

257. As discussed at limitations [15.0]-[15.1.1], [17.1], Gai discloses that the computer network 300 comprises an Ethernet LAN and that network device 130 comprises an Ethernet LAN switch that processes Ethernet data frames, which renders obvious "*wherein the network comprises an Ethernet network, and wherein the switch comprises an Ethernet switch and the packets comprise Ethernet packets.*"

VIII. DECLARATION

258. I declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true, and that these statements were made with knowledge that willful false statements and

Declaration of Mark E. Crovella, Ph.D.
Inter Partes Review of U.S. 7,623,518

the like so made are punishable by fine or imprisonment, or both, under section
1001 of Title 18 of the United States Code.



Dated: April 1, 2024

Mark E. Crovella, Ph.D.