UNITED STATES PATENT AND TRADEMARK OFFICE

_____

BEFORE THE PATENT TRIAL AND APPEAL BOARD

_____

CISCO SYSTEMS, INC.,
Petitioner

_____

IPR2024-00734
U.S. Patent No. 7,623,518

PETITION FOR *INTER PARTES* REVIEW UNDER 35 U.S.C. § 312 AND 37 C.F.R. § 42.104

TABLE OF CONTENTS

PETITIONER'S EXHIBIT LIST 5

I. INTRODUCTION 6

II. GROUNDS FOR STANDING 6

III. NOTE 6

IV. THE '518 PATENT 6

V. PROSECUTION HISTORY 7

VI. LEVEL OF ORDINARY SKILL IN THE ART 7

VII. CLAIM CONSTRUCTION 7

A. "*at least one of*" and "*each of*" 7

VIII. RELIEF REQUESTED AND REASONS THEREFOR 8

IX. IDENTIFICATION OF HOW THE CLAIMS ARE UNPATENTABLE 9

A. Challenged Claims and Statutory Grounds 9

B. Ground 1 10

1. Gai 10

2. Yip 12

3. Kwan 12

4. Reasons to Combine Yip with Gai 14

5. Reasons to Combine Kwan with Gai 17

6. Claim 1 22

7. Claim 3 42

C. Ground 2: Claims 15, 17, 19-21 are Obvious over Gai, Kwan, and Georgiou 43

1. Georgiou 43

2. Reasons to Combine Georgiou with Gai 44

3. Claim 15 48

4. Claim 17 79

5. Claim 19 79

6. Claim 20 84

7. Claim 21 85

X. DISCRETIONARY DENIAL IS INAPPROPRIATE 85

A. No *Fintiv* denial 85

1. No evidence regarding a stay 85

2. Parallel trial date 85

3. Investment in the parallel proceeding 86

4. Issue overlap 86

5. Parties 87

6. Other circumstances 87

B. No §325(d) denial 87

XI. CONCLUSION 87

XII. MANDATORY NOTICES 88

A. Real Party-in-Interest 88

B. Related Matters..... 88

C. Lead and Back-up Counsel and Service Information 89

CERTIFICATE OF WORD COUNT.....90

CERTIFICATE OF SERVICE 91

PETITIONER'S EXHIBIT LIST

Ex.1001	U.S. 7,623,518
Ex.1002	Prosecution History of U.S. 7,623,518
Ex.1003	Declaration of Mark E. Crovella, Ph.D. under 37 C.F.R. § 1.68
Ex.1004	*Curriculum Vitae* of Mark E. Crovella, Ph.D.
Ex.1005	Complaint filed on May 9, 2024
Ex.1006	U.S. Patent No. 6,651,096 to Gai et al. ("Gai")
Ex.1007	U.S. Patent No. 7,523,485 to Kwan ("Kwan")
Ex.1008	U.S. Patent No. 7,003,597 to Georgiou et al. ("Georgiou")
Ex.1009	U.S. Patent No. 6,105,086
Ex.1010	U.S. Patent Publication No. 2001/0012294A1
Ex.1011	U.S. Patent No. 5,720,032
Ex.1012	U.S. Patent No. 5,797,033
Ex.1013	U.S. Patent No. 8,751,647
Ex.1014	U.S. Patent No. 7,343,485
Ex.1015	Median time-to-trial in the Eastern District of Texas
Ex.1016	U.S. Patent Publication No. 2005/0055570A1
Ex.1017	Intellinet User Manual
Ex.1018	Netgear WG602 Users Guide
Ex.1019	Email from Seth Hasenour to Emily Whitcher, "Re: Lionra vs. Cisco (Lionra II) – Letter," December 4, 2023.
Ex.1020	Docket Control Order, *Lionra Techs. Ltd. v. Cisco Systems, Inc.*, No. 2:23-cv-00206, Dkt. 30 (E.D. Tex. Aug. 9, 2023).

I. INTRODUCTION

Pursuant to 35 U.S.C. §§ 311, 314(a), and 37 C.F.R. § 42.100, Cisco Systems, Inc. ("Petitioner") respectfully requests that the Board institute *inter partes* review and cancel claims 1, 3, 15, 17, and 19-21 ("Challenged Claims") of U.S. 7,623,518 ("'518 patent") as unpatentable under pre-AIA 35 U.S.C. § 103.

II. GROUNDS FOR STANDING

Petitioner certifies that the '518 patent is eligible for IPR. Petitioner is not barred or estopped from requesting this IPR. 37 C.F.R. § 42.104(a).

III. NOTE

Petitioner cites to exhibits' original page or column numbers where available. Emphasis in quoted material has been added.

IV. THE '518 PATENT

The '518 patent describes "control[ing] access of a user to a network." Ex.1001, 3:20-22. The '518 patent uses an enhanced access control list with "data related to at least one of user names, DNS names, domain names, and physical addresses." Ex.1001, 3:22-29. The enhanced access control list is used to generate a "dynamic access control list." Ex.1001, 3:22-29; Ex.1003, ¶¶23-26.

V. PROSECUTION HISTORY

Claim 1 was allowed after incorporating details on "*converting user names into corresponding IP and physical addresses*." Claim 15 (originally numbered 18) was allowed without amendment. *See* Ex.1002, 21-24, 8-13; Ex.1003, ¶¶27-30.

VI. LEVEL OF ORDINARY SKILL IN THE ART

A Person of Ordinary Skill in The Art ("POSITA") on April 8, 2004, would have been someone knowledgeable with the computer networking 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. Ex.1003, ¶¶15-22.

VII. CLAIM CONSTRUCTION

Claim terms in IPR are construed according to their "ordinary and customary meaning." 37 C.F.R. § 42.100(b). Ex.1003, ¶¶31-34.

In co-pending litigation, Petitioner may argue that some claims are indefinite under 35 U.S.C. § 112. Such issues are outside the scope of this proceeding and do not preclude the Board's evaluation of grounds presented below.

A. "*at least one of*" and "*each of*"

Claim 1 uses the phrase "*at least one of*" in both the "*developing*" step [1.1]

and "*converting*" step [1.2.1]. Additionally, claim 1 uses the phrase "*each of*" in the "*developing*" step [1.3].

In the co-pending litigation, Patent Owner "disagree[s]" that the term "*at least one of*" requires the claimed "*enhanced access control list*" include "data related to all four of the following: user names, DNS names, Windows domain names, and physical addresses." Ex.1019. Patent Owner declined to explain its proposed interpretation until claim construction, which begins May 2024. Ex.1019; Ex.1020.

Accordingly, the Petition applies 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. Ex.1003, ¶¶33-34; *see Western Digital Corp. v. SPEX Tech. Inc*., IPR2018-00084, Paper 14, 10-11 (P.T.A.B. Apr. 25, 2018).

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]), the Petition applies Patent Owner's apparent district court interpretation to mean developing from the at least one of the listed items of converting.

VIII. RELIEF REQUESTED AND REASONS THEREFOR

Petitioner asks that the Board institute a trial for IPR and cancel the Challenged Claims. Petitioner challenges the claims of the '518 patent because they are asserted in co-pending litigation. Their cancellation will reduce the time

and expense of the litigation for all parties.

IX. IDENTIFICATION OF HOW THE CLAIMS ARE UNPATENTABLE

A. Challenged Claims and Statutory Grounds[1]

Ground	Claims	Basis
#1	1, 3	§ 103 (Pre-AIA) over Gai, Yip, and Kwan
#2	15, 17, 19-21	§ 103 (Pre-AIA) over Gai, Yip, Kwan, and Georgiou

U.S. 6,651,096 (Ex.1006, "Gai"), was filed by Cisco Systems, Inc. on 4/20/1999, and issued 11/18/2003.

U.S. 8,751,647 (Ex.1007, "Yip") was filed on 6/30/2001, and issued 6/10/2014.

U.S. 7,523,485 (Ex.1007, "Kwan") was filed on 7/31/2003 and issued 4/21/2009. Kwan claims priority to provisional applications filed on 5/21/2003.

U.S. 7,003,597 (Ex.1008, "Georgiou") was filed on 7/9/2003 and issued 2/21/2006.

These references are prior art under Pre-AIA 35 U.S.C. §§ 102(a) and

[1] For each combination, Petitioner relies on the teachings, and not on a physical incorporation of elements. *See In re Mouttet*, 686 F.3d 1322, 1332 (Fed. Cir. 2012); Ex.1003, ¶¶104, 202, 229.

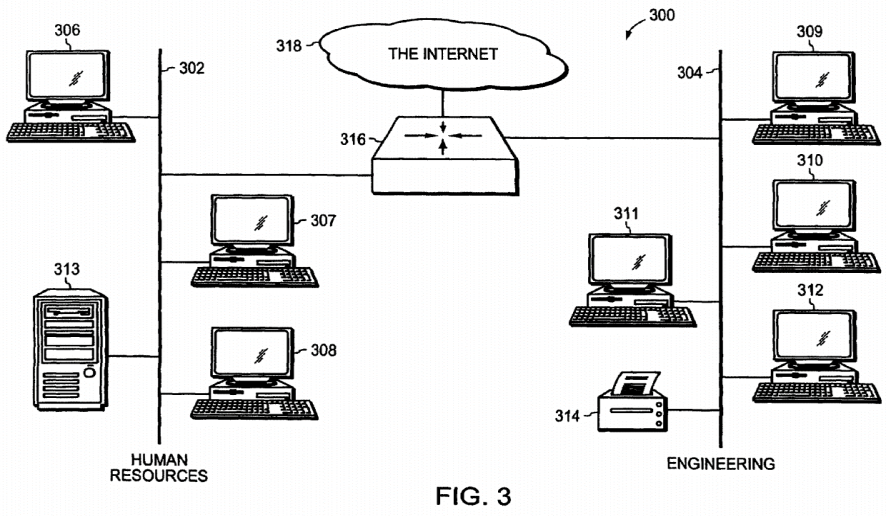
102(e).

Petitioner cites additional prior art demonstrating knowledge in the art and providing contemporaneous context regarding what a POSITA would have understood from the asserted prior art. *See Yeda Research v. Mylan Pharm. Inc.*, 906 F.3d 1031, 1041-1042 (Fed. Cir. 2018); 37 C.F.R. § 42.104(b).

B. Ground 1

1. Gai

Gai generally discloses a network device 316 that operates in a packet switch network 300. Ex.1006, 1:14-18, 1:59-2:4, 5:23-34, Figs. 3-4. Figure 3 illustrates network 300:



Ex.1006, Fig. 3.

Figure 4 illustrates network device 316:

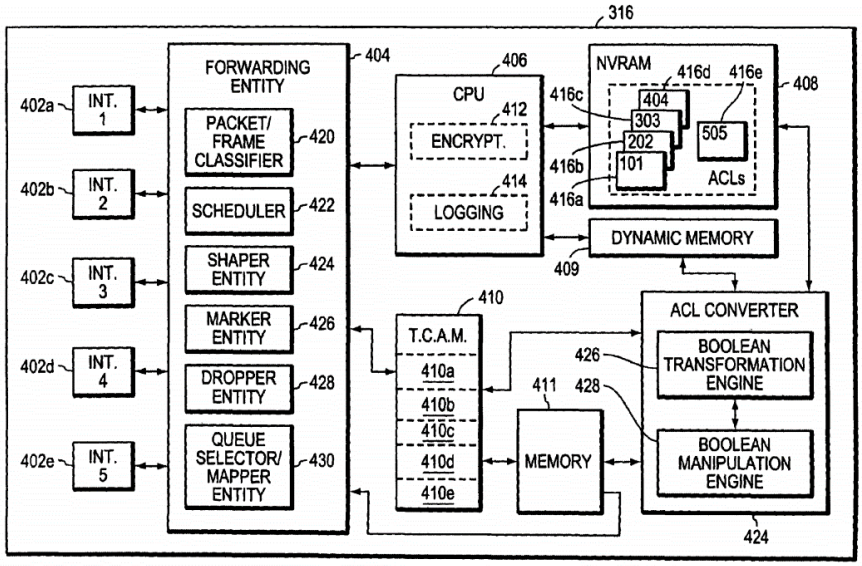


FIG. 4

Ex. 1006, Fig. 4.

Network device 316 "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." Ex.1006, 5:66-6:7. NVRAM 408 includes "one or more text-based access control lists (ACLs) 416a-416e" created by an administrator. Ex.1006, 2:58-60, 6:13-18, 7:60-66. An ACL converter 424 optimizes the text-based ACLs into boolean Binary Decision Diagram ('BDD') representations. Ex.1006, 9:22-11:7, Fig. 6, Fig. 7. The BDD ACL is then stored into TCAM 410 and applied to packets. Ex.1006, 14:57-62; Ex.1003, ¶¶37-43.

2. Yip

Yip describes "a network security system that controls access to a network at the edges of the network." Ex.1013, 1:6-1:10. Yip security system includes "an administrator-controlled list of Internet Protocol (IP) addresses or media access control (MAC) addresses… authorized to access the network." Ex.1013, 2:43-47. Ex.1003, ¶¶44-45.

3. Kwan

Kwan describes improving network security by reducing risks from "the spoofing of IP addresses." Ex.1007, 1:15-17; *see also* Ex.1007, 1:20-31. Kwan illustrates a network device 200 in Figure 2 (below). Ex.1007, 3:42-46, 4:10-55. Within network device 200, a port security processor 242 "provides for dynamically adjusting inbound source IP address anti-spoofing blocking criteria for each port." Ex.1007, 4:56-63. A source IP address detector 236 learns new IP addresses in multiple ways, including by monitoring "reverse address resolution protocol (RARP)" and dynamic host control protocol (DHCP) traffic. Ex.1007, 5:38-67.

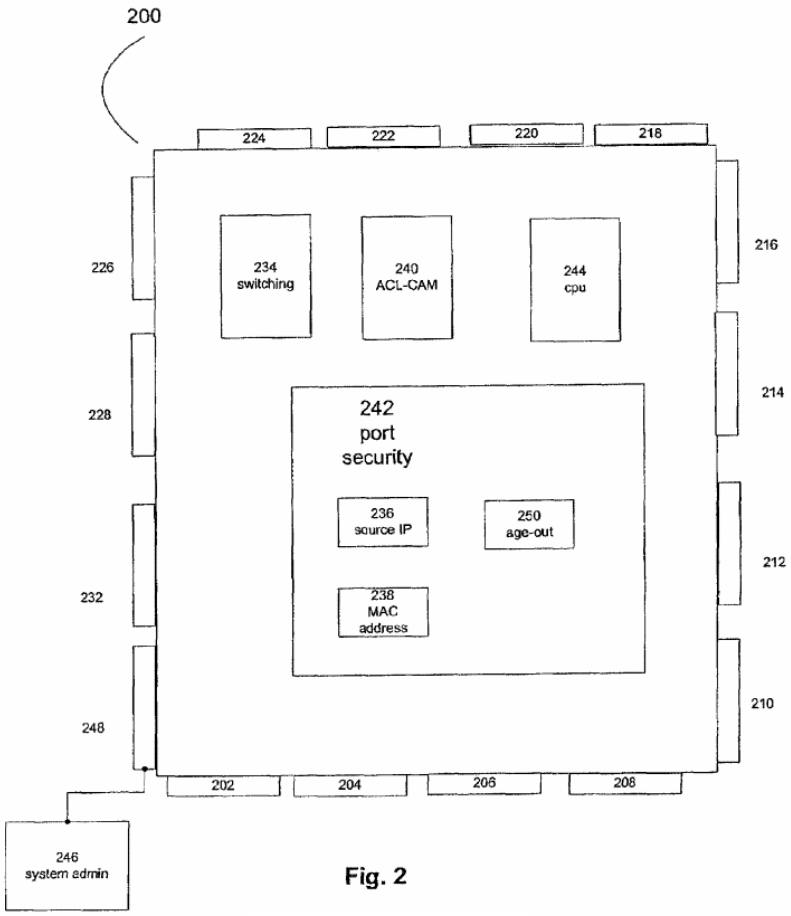
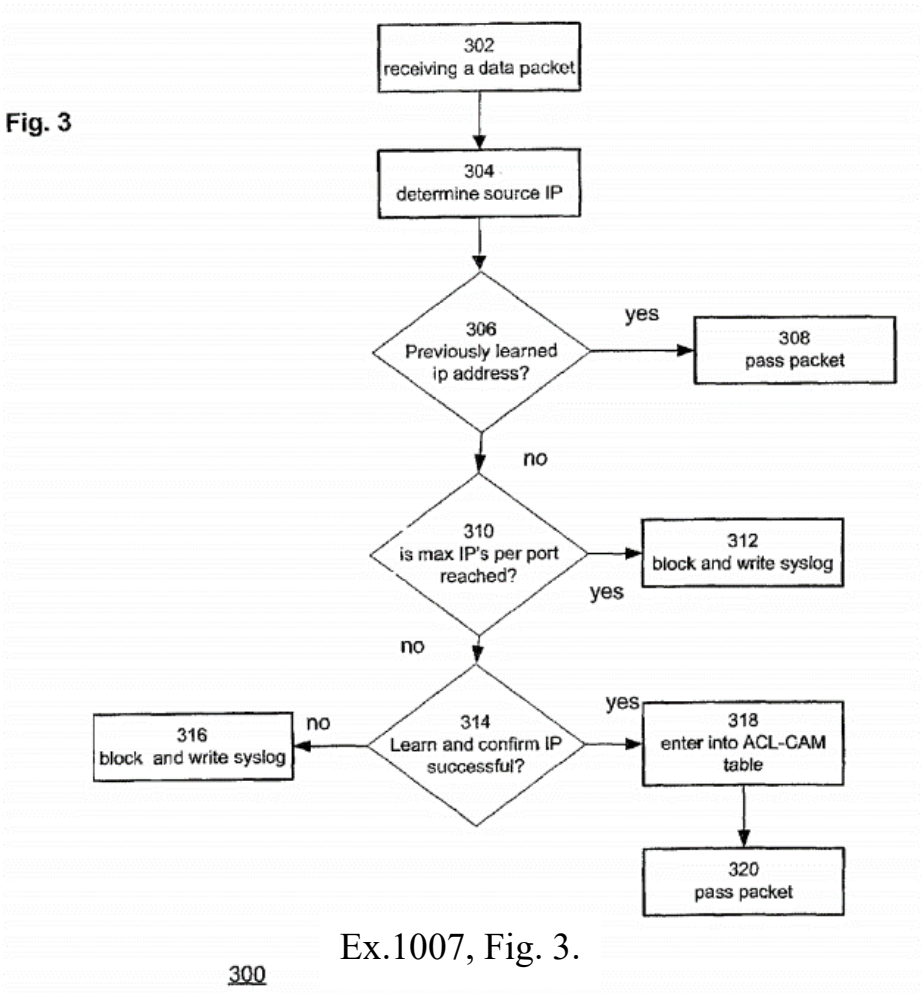


Fig. 2

Ex.1007, Fig. 2.

An administrator can prevent potentially learning incorrect IP addresses by specifying "IP address properties." Ex.1007, 6:14-22. Providing these IP address properties "will eliminate false learning of a default" IP address. Ex.1007, 6:22-25. An ACL-CAM is updated with newly learned IP addresses. Ex.1007, 6:28-30. Figure 3 illustrates Kwan's learning process (*see* Ex.1007, 8:17-48, 7:60-66):



Ex.1007, Fig. 3.

After a new IP address is learned from a packet, the packet is passed to its destination. Ex.1007, 8:48-51; Ex.1003, ¶¶46-50.

4. Reasons to Combine Yip with Gai

A POSITA would have been motivated to combine the teachings of Yip with Gai, as discussed below. Ex.1003, ¶51.

a. ***Analogous Art***

Gai and Yip are analogous art to the ’518 patent. 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 Kwan, which provides additional access control list teachings applicable to Gai. Ex.1003, ¶52.

b. *Motivation to Combine*

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

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; *see also* Ex.1006, 2:46-57, 3:26-30, 5:3-21, 6:13-17, 21:1-7, Fig. 3, 5A-5E; Ex.1003, ¶¶53-54.

Gai's ACLs are designed to provide "security…[and] implement additional features." Ex.1006, 8:3-6. It was known for ACLs to include MAC addresses to improve security by 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; Ex.1017, 55; Ex.1018, 13. Consistent with that knowledge, Yip describes an administrator including MAC addresses in ACLs. Ex.1013, 2:43-47; Ex.1003, ¶¶55-56.

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 MAC addresses 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 packet filtering 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; Ex.1003, ¶57.

The proposed combination merely represents a combination of prior art elements (MAC addresses, per Yip, with Gai's ACL) according to known methods 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). Ex.1003, ¶58.

c. *Reasonable Expectation of Success*

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* Ex.1006, 2:58-60. The expectation

of success is also shown by other implementations of ACLs with MAC addresses. *See* Ex.1014, 6:30-32, 8:27-47, 11:57-62; Ex.1017, 55; Ex.1018, 13; Ex.1003, ¶¶59-61.

5. Reasons to Combine Kwan with Gai

A POSITA would have also been motivated to combine the teachings in Kwan with Gai (as modified per Yip), as discussed below. Ex.1003, ¶62.

a. ***Analogous Art***

Kwan is analogous art to the '518 patent. Kwan (like Gai) is in the same field of endeavor of computer networks and addresses 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. Ex.1003, ¶63.

b. ***Motivation to Combine***

A POSITA would have been motivated to combine the teachings of Kwan with Gai to produce predictable and beneficial results as discussed below. Ex.1003, ¶64.

Gai's network device 316 uses 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 are initially set up by an administrator. Ex.1006, 7:16-29; Ex.1003, ¶65. 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. 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 access criteria when processing packets. Ex.1007, 4:52-55 ("One limitation of utilizing a prior switch was that… 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 dynamically assigned. Ex.1003, ¶66.

Kwan beneficially discloses how to learn dynamically assigned IP addresses . *See* Ex.1007, 2:7-3:4. Kwan's network device 200 provides switching functions (analogous to Gai's network device 316). Ex.1007, 3:43-58. 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; Ex.1003, ¶67.

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." Ex.1007, 8:25-38; *see also* Ex.1007, 5:38-67 (disclosing ways to learn an IP address). Kwan describes in detail, for example, an IP address "learned by using a reverse address resolution protocol (RARP)" (Ex.1007, 5:38-46) or "by listening to [a] DHCP response packet being returned to the host…[which] contains the source IP Address for the host." 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; Ex.1003, ¶68.

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. This process "will eliminate false learning." Ex.1007, 6:18-25. A POSITA would have recognized that the administrator-provided seed IP address properties allow a new IP address consistent with the seed IP address properties 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 so that only

consistent IP addresses will be learned and permitted to communicate on the network. Ex.1003, ¶69.

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-created text-based ACLs. In further view of Kwan, it would have been obvious to a POSITA to implement Gai's administrator-created text-based ACLs to include MAC addresses and seed IP address properties and to use this information to learn new IP addresses. Ex.1003, ¶70. 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. Such an implementation would obviate the need for Gai's administrator to manually update text-based ACLs when a host's IP address changes or 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 computer's dynamically assigned IP address. Ex.1003, ¶70.

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. It would have also been obvious to limit IP address learning to IP addresses that are consistent with the seed IP address properties in the ACLs, thereby lowering the "risk of 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. Kwan's teachings would also beneficially help combat source IP spoofing. Ex.1007, 3:1-4; Ex.1003, ¶71.

The proposed combination merely represents a combination of prior art elements (MAC addresses, 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 by (1) providing 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. Ex.1003, ¶72.

c. *Reasonable Expectation of Success*

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. Modifying Gai's administrator-created text-based ACLs to include MAC addresses and seed IP address properties would have been "conventional" and within the level of ordinary skill. *See* Ex.1006, 2:58-60. Ex.1003, ¶73.

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 using RARP and DHCP techniques. Ex.1003, ¶¶74-75.

6. Claim 1

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

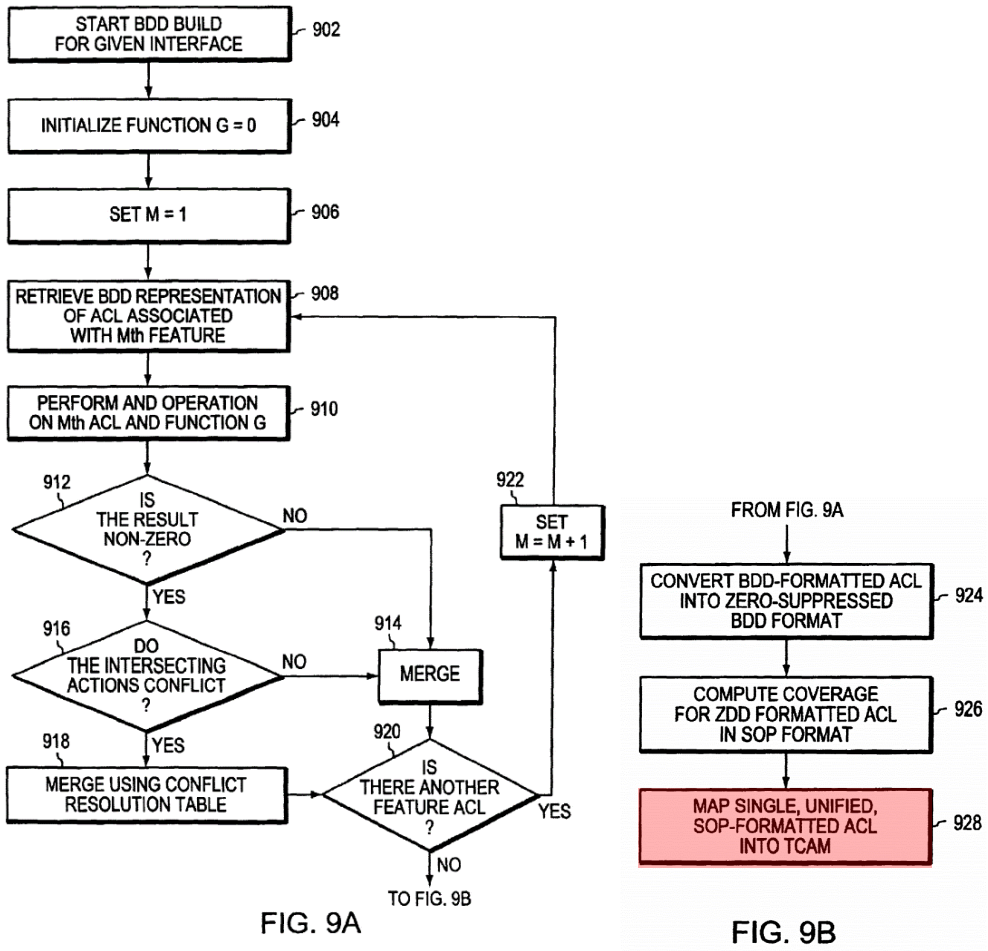
To the extent limiting, the prior art renders obvious the preamble. Ex.1003, ¶76.

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; Ex.1003, ¶77.

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 416*a*) 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)."); Ex.1003, ¶78.

Gai's Figures 9A and 9B are provided below, illustrating a method for dynamically generating optimized ACLs that are stored in TCAM. Ex.1003, ¶79.



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

Gai's dynamically generated TCAM ACL corresponds to an "*access control list*" as recited in the preamble. Ex.1003, ¶80.

Second, as analyzed below at [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. Ex.1003, ¶81.

Thus, Gai in combination with Yip and Kwan discloses a method of dynamically generating an optimized TCAM ACL, which renders obvious the

preamble. Ex.1003, ¶82.

[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;***

As discussed in the claim construction section, the Petition applies Patent Owner's apparent district court interpretation of the claim term "*at least one of*" as requiring only one of the listed items. The below analysis demonstrates that the prior art renders obvious "*developing an enhanced access control list including data related to…physical addresses.*" Ex.1003, ¶¶83-84.

First, Gai teaches "*developing an enhanced access control list"* by disclosing that the administrator creates one or more "text-based ACLs." Ex.1006, 7:15-32, Fig. 1. Gai's one or more text-based ACLs are used "for security" and to "implement 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, 2:46-57, 3:24-30, 7:29-9:21, Figs. 5A-5E, 10A-10B; Ex.1003, ¶85.

To the extent it is argued that the claimed "*enhanced access control list*" requires a single access control list, such would have been obvious. Although Gai contemplates a plurality of text-based ACLs (e.g., 416*a*-416*e*), Gai expressly discloses "**one** or more text-based ACLs." Ex.1006, 3:57-61; *see also* Ex.1006, 20:25-29. 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. Ex.1003, ¶¶86-87.

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*." Ex.1003, ¶88.

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; Ex.1016, claim 2; Ex.1017, 55, 84; Ex.1018, 13. 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."* Specifically, Yip discloses that "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; Ex.1003, ¶¶89-90.

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. 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. 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 a 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; Ex.1003, ¶91.

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 in 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 the network based on MAC address. Ex.1003, ¶92.

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; Ex.1017, 84; Ex.1016, claim 2; Ex.1003, ¶93.

Thus, Gai in combination with Yip discloses creating text-based ACLs that include MAC addresses, which renders obvious this limitation. Ex.1003, ¶94.

Third, Gai and Yip in further combination with Kwan also renders obvious "*developing an enhanced access control list including data related to…physical addresses*." Ex.1003, ¶95.

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; Ex.1003, ¶96.

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]; Ex.1003, ¶97.

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. As Kwan recognizes, "it 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." Ex.1007, 7:61-66. As such, Kwan's teachings 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 addresses. Also, 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. *See also* Reasons to Combine Kwan with Gai; Ex.1003, ¶98.

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; Ex.1003, ¶99.

Additionally, to reduce the risk of learning wrong IP addresses, Kwan discloses that the system administrator provides seed IP address properties. 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 IP addresses consistent with the seed IP address properties 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. Ex.1003, ¶100.

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. As an additional reason, Kwan's teachings would beneficially help avoid spoofing. Ex.1007, Abstract. The proposed combination 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; Ex.1003, ¶¶101-102.

As discussed 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; Ex.1003, ¶103.

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, which renders obvious this limitation. Ex.1003, ¶104.

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

First, Gai discloses a "*network switch"* by teaching an "network device 316" that includes circuity for performing "switching functions" in "network 300." Ex.1006, 5:22-34; *see also* Ex.1006, 6:62-65. 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.1003, ¶¶105-106.

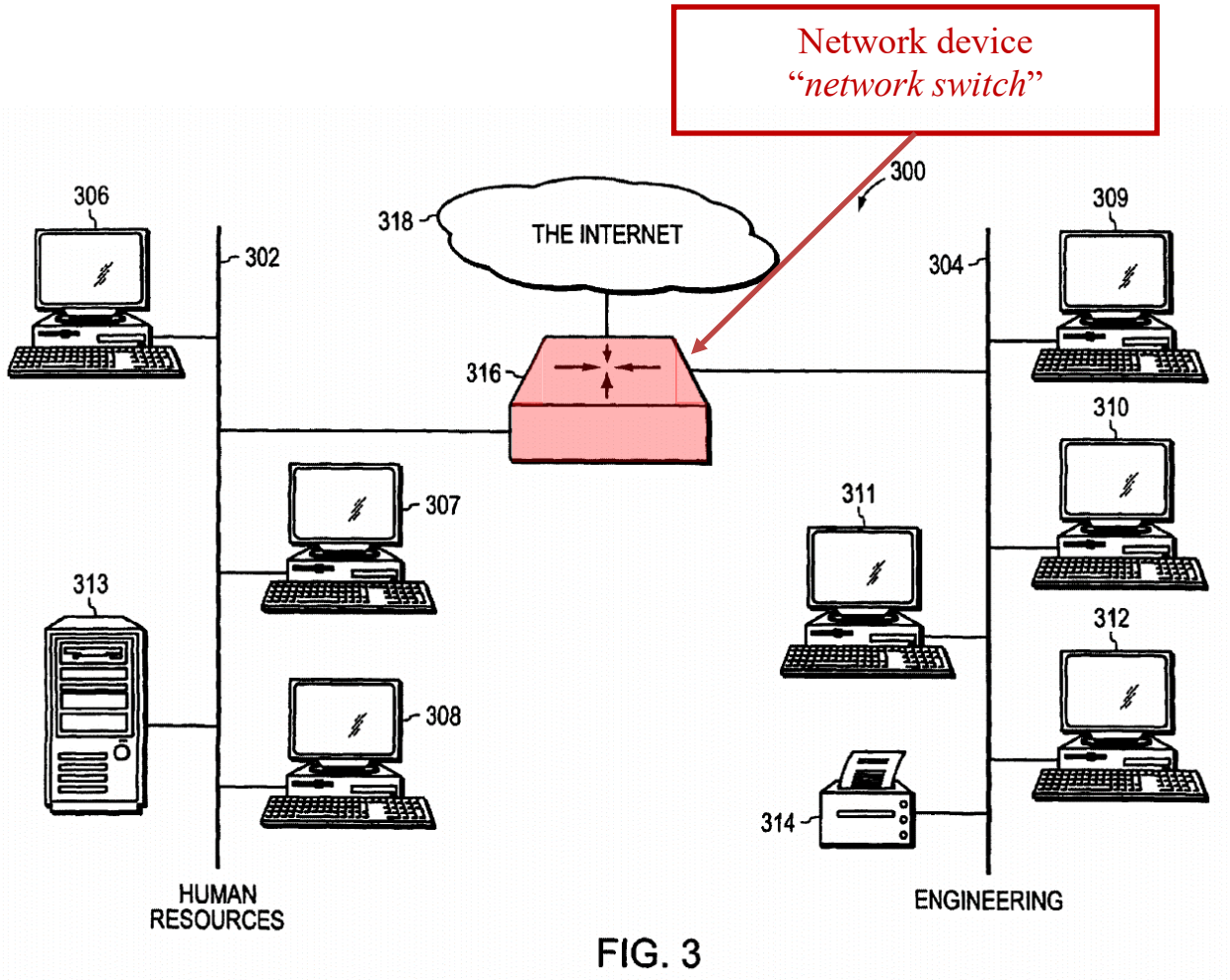


FIG. 3

Ex.1006, Fig. 3 (annotated).

Thus, Gai's network device 316 that performs switching functions renders obvious "*a network switch*," as claimed. Ex.1003, ¶¶107-108.

Second, as discussed in the claim construction section, the Petition applies Patent Owner's apparent district court interpretation of the claim term "*at least one of*" to require only one of the listed "*converting*" steps recited at [1.2.2]-[1.2.4]. The below analysis demonstrates 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]. Ex.1003, ¶109.

[1.2.2] ***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*****;***

As analyzed in the claim construction section, the Petition applies Patent Owner's apparent district court interpretation of "*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. Ex.1003, ¶110.

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

As analyzed in the claim construction section, the Petition applies Patent Owner's apparent district court interpretation of the claim's recitation of "*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. Ex.1003, ¶111.

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

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.*" Ex.1003, ¶¶112-113.

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; Ex.1003, ¶114.

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; Ex.1003, ¶114.

(1) Converting using DHCP technique

In a first example, Kwan discloses listening to DHCP response packets to learn new IP addresses according to MAC addresses. In more detail, Kwan "learn[s] the IP address for a host by listening to the DHCP response packet being returned to the host." Ex.1007, 5:44-46. "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, 5:46-50. In Kwan, "storing of the new source IP address in a table, can include storing the information in an ACL-CAM." Ex.1007, 8:46-48. Ex.1003, ¶115.

Kwan discloses that the new IP address is only matched to a known MAC address if it is consistent with seed IP properties. Ex.1007, 6:14-27. 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 only 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. Ex.1003, ¶¶116-117.

Notably, Kwan's disclosure of listening to DHCP return packets is substantially similar to the '518 patent's converting technique of "monitor[ing] DHCP return packets…and from these return packets determin[ing] the IP address assigned by the DHCP server to a given MAC address." *See E.g.,* Ex.1001, 7:51-67; *see also* Ex.1001, claim 7 ("…*converting physical addresses into IP addresses…comprises…monitoring DHCP packets*…"); Ex.1003, ¶118.

(2) Converting using RARP technique

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. Ex.1003, ¶119.

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.*" Ex.1003, ¶120.

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 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 (e.g., using DHCP), whose use was prevalent. *See* Ex.1014, 1:16-19; Ex.1018, 10. 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; Ex.1003, ¶121.

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 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]). Ex.1003, ¶122.

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 to data (e.g., known MAC addresses and seed IP address) in one or more text-based ACLs, which renders obvious this limitation. Ex.1003, ¶123.

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

First, as analyzed in the claim construction section, the Petition applies Patent Owner's apparent district court interpretation of the claim's recitation of "*at least one of*" at limitation [1.2.1] to require only one of the listed converting steps to be performed, and applies Patent Owner's apparent district court interpretation of "*each of"* the operations of converting to require only one converting operation recited at [1.2.2]-[1.2.4]. Ex.1003, ¶¶124-125.

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*"). Ex.1003, ¶126.

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; Ex.1003, ¶127.

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. Ex.1003, ¶128.

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 the new IP address. *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; Ex.1003, ¶129.

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 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 the new IP addresses. This processing sequence would allow for Gai's network device 316 to apply optimized TACM ACLS with the 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; Ex.1003, ¶130.

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 this limitation. Ex.1003, ¶131.

7. Claim 3

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

As discussed at [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 this limitation. Ex.1003, ¶132.

C. Ground 2: Claims 15, 17, 19-21 are Obvious over Gai, Kwan, and Georgiou

1. Georgiou

Georgiou "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 to store Internet Protocol (IP) packets in memory while awaiting processing. Ex.1008, 3:36-51; *see also* Ex.1008, 1:22-27, 3:17-4:26.

Figure 1 (below) illustrates a direct memory access controller DMA 140 that stores packets in memory (buffers 200 and 250) while awaiting processing by the host processor 110. Ex.1003, ¶¶133-134.

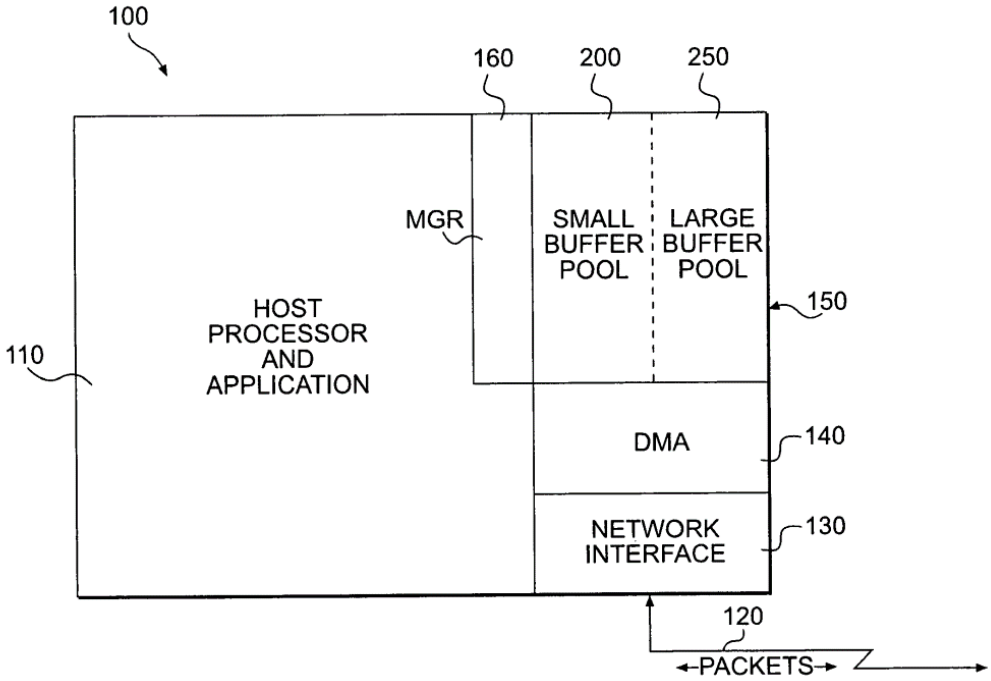


FIG. 1

Ex.1008, Fig. 1.

2. Reasons to Combine Georgiou with Gai

A POSITA would have been motivated to combine the teachings of Georgiou with Gai (as modified per Kwan), as discussed below. Ex.1003, ¶135.

a. *Analogous Art*

Georgiou is analogous art. Like the '518 patent, Georgiou pertains to computer networks and addresses 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, ¶136.

b. *Motivation to Combine*

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. Ex.1003, ¶137.

Gai discloses that packets are "forward[ed] to CPU for additional processing." Ex.1006, 13:41-45; *see also* Ex.1006, 12:2-7. 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, and to forward such packets 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 a significant number of packets that need processing, the CPU may not be able to keep up with the packet flow. Ex.1003, ¶138.

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; Ex.1003, ¶139.

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.). Ex.1003, ¶140.

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. Ex.1003, ¶141.

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. Ex.1003, ¶142.

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 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. Ex.1003, ¶143.

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. Ex.1003, ¶144.

c. *Reasonable Expectation of Success*

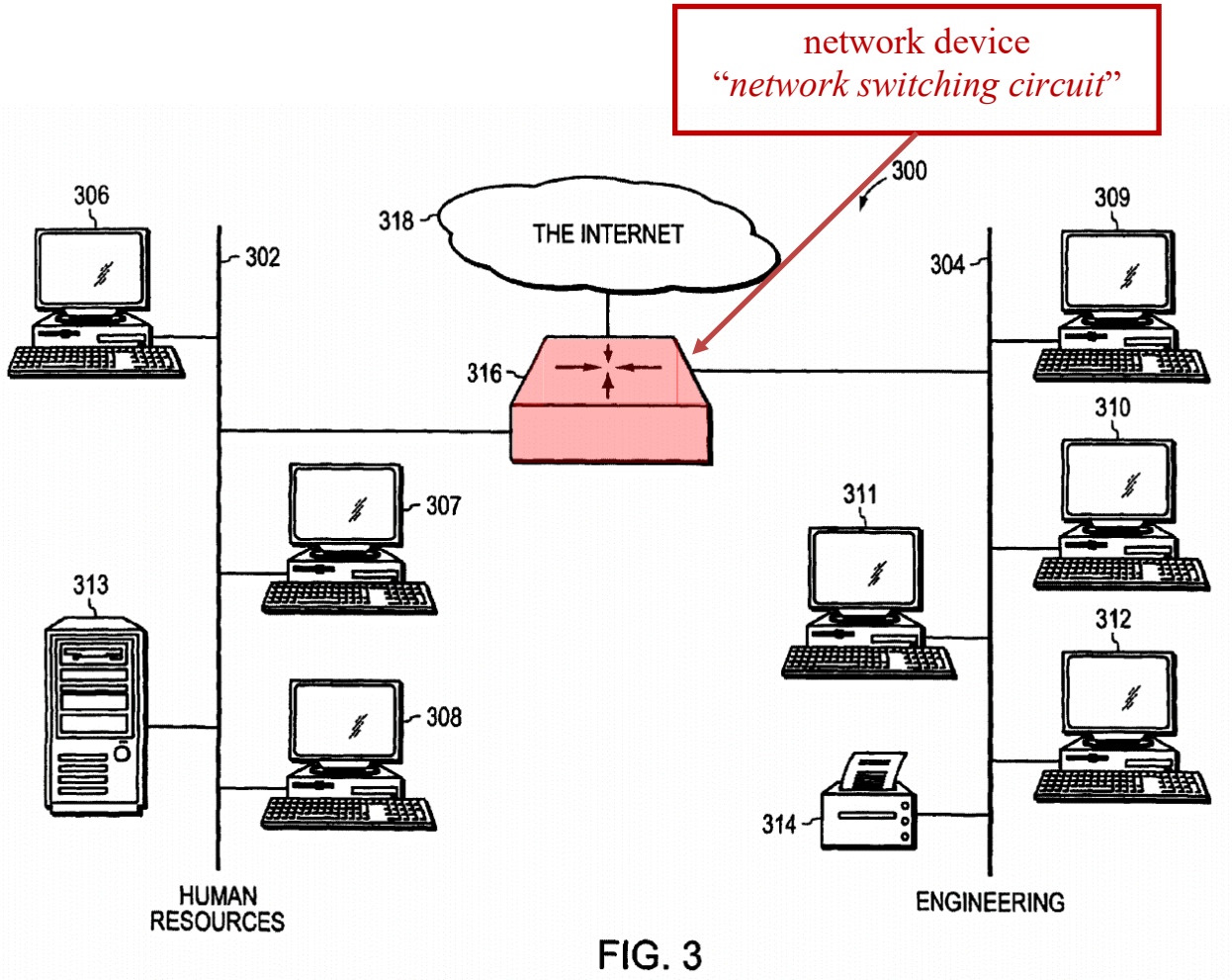
The results would have been predictable and there would have been a reasonable expectation of success in the combination given the similarity in Georgiou's and Gai's systems. Any modification needed to Gai, in order to accommodate the teachings of Georgiou, including implementing a DMA circuit to store 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; Ex.1009, 1:32-34; Ex.1010, [0216]; Ex.1011, 11:43-45; Ex.1012, Abstract; Ex.1003, ¶¶145-146.

3. Claim 15

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

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.1003, ¶147.



Ex.1006, Fig. 3 (annotated).

Thus, Gai's network device 316 that includes switching circuitry renders obvious "[*a] network switching circuit,*" as recited in the preamble. Ex.1003, ¶148.

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]. Ex.1003, ¶149.

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

First, Gai discloses "*a forwarding circuit*" by disclosing a "**forwarding entity 404**," Ex.1006, 8:26-27. The "[f]orwarding entity 404 may include a plurality of **conventional sub- components"** 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."). Ex.1003, ¶¶150-151

Gai's forwarding entity 404 is illustrated at Figure 4 reproduced below.

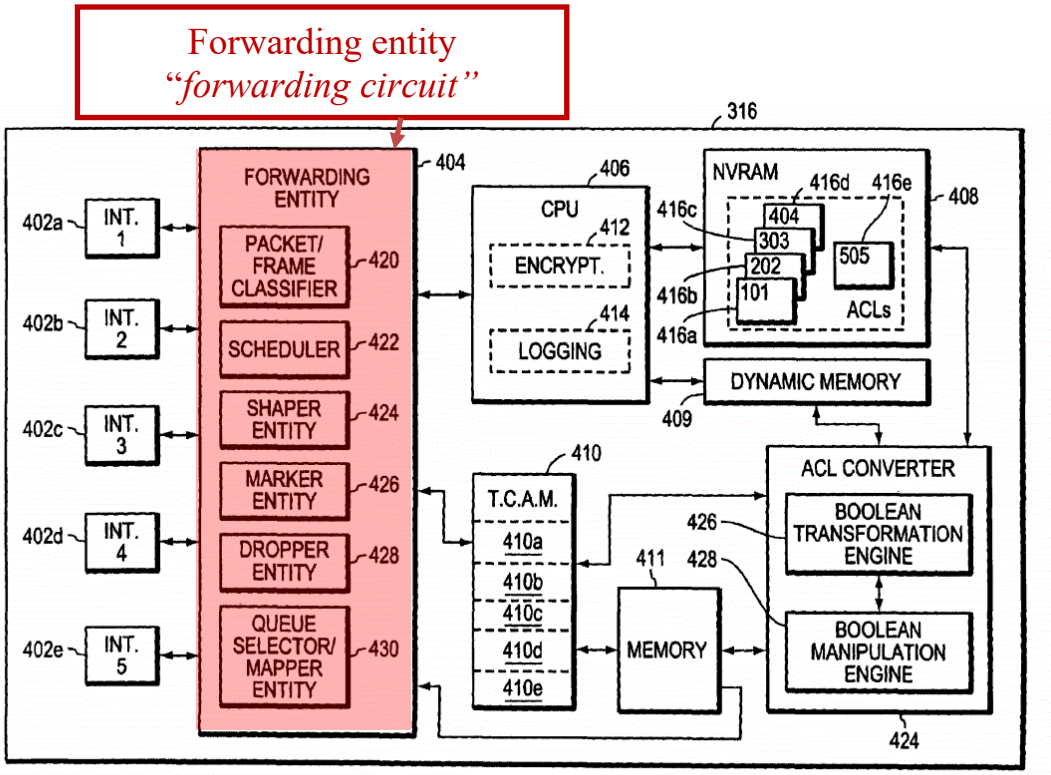


FIG. 4

Ex.1006, Fig. 4 (annotated).

Accordingly, Gai's "forwarding entity 404" renders obvious the claimed "*forwarding circuit.*" Ex.1003, ¶151.

Second, Gai in combination with Kwan, in two different ways, renders obvious "*detect[ing] specific received packets.*" Ex.1003, ¶152.

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

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; Ex.1003, ¶153.

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.). Ex.1003, ¶154. "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. In the present example, the TCAM ACL is used 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.). Ex.1003, ¶154.

In another example, Gai teaches that rather than using TCAM, "the ACLs may be left in EDD or ZDD format, stored at the dynamic memory 409 and evaluated, essentially through software, by the forwarding entity 404." Ex.1006, 18:65-19:3. The Petition will generally refer to Gai's TCAM ACL disclosure, with the understanding that the analysis below equally applies when the ACL is stored in dynamic memory 409.

2) Detecting specific received packets with new IP addresses[2]

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; Ex.1003, ¶155.

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 [240]." Ex.1007, 4:34-39; *see also* Ex.1007, 4:39-48. The access control list (called ACL-CAM 240) 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. Ex.1007, 5:25-30. If a packet is detected with a new IP address, a dynamic "learn source IP routine" is performed. Ex.1007, 8:17-47, Fig. 3; *see also* Ex.1007, 5:38-67, 7:61-66. 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. Ex.1003, ¶156.

[2] To provide antecedent basis for later limitations, the Petition further applies the combination of Gai and Kwan.

Gai's ACL includes source IP addresses. Ex.1006, 7:29-39, 21:1-7. 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; *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; *see also* Reasons to Combine Kwan with Gai; Ex.1003, ¶157.

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 this limitation. Ex.1003, ¶158.

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

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*." Ex.1003, ¶¶159-160.

Second, Gai teaches providing packets to a port of CPU 406 for additional processing. "In particular, upon receipt of a packet at a first 30 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." Ex.1006, 4:29-33. "When a match is obtained, the corresponding decision is returned to the forwarding entity, which then takes the appropriate action 35 (e.g., forward, discard, log and forward, **transfer to CPU for additional processing**, etc.)" Ex.1006, 4:29-37; *see also* Ex.1006, 11:60-12:7 ("[T]he message [may] be **forwarded to the CPU 406** for encryption…"), 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 packets match ACL rules) the action would include forwarding or transferring the specific packets to the CPU for further processing. Ex.1003, ¶161.

Gai's Figure 4 below illustrates with double-headed arrows that the forwarding entity 404 provides packets to a port of the CPU 406. Ex.1003, ¶162.

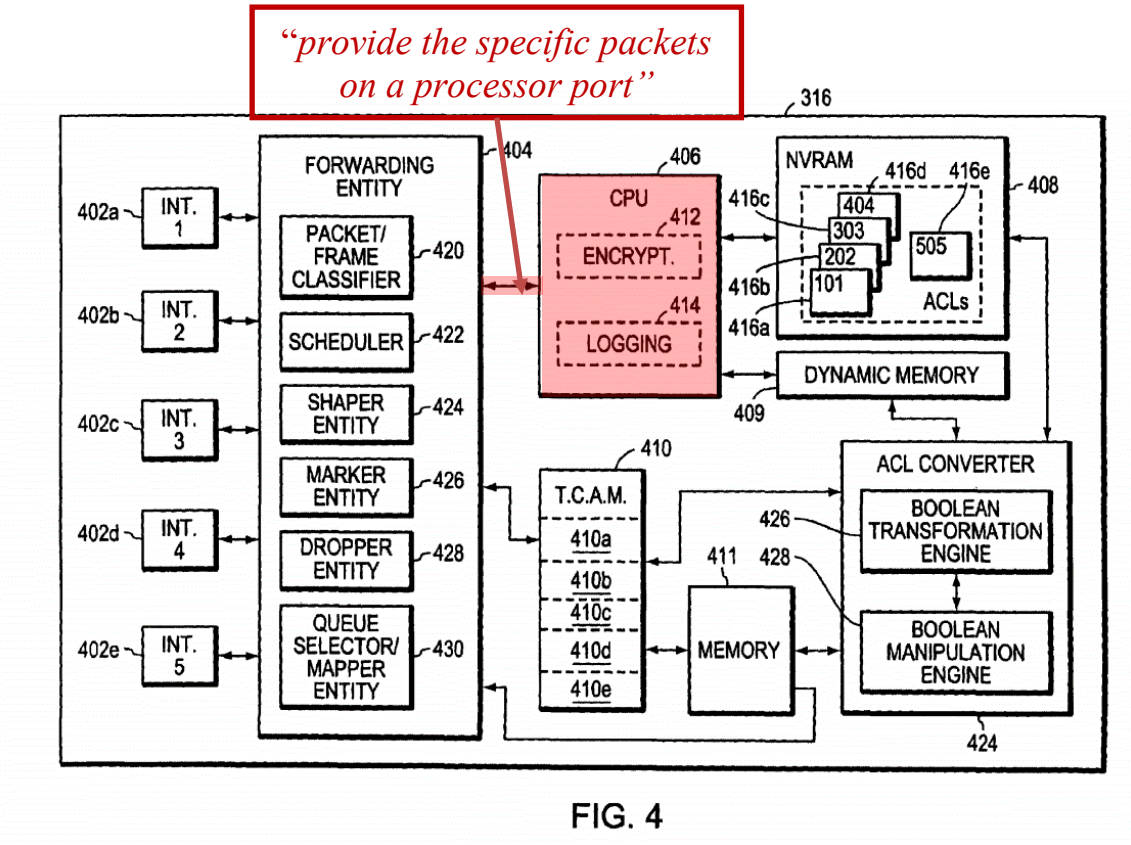


FIG. 4

Ex.1006, Fig. 4 (annotated).

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. Ex.1007, 8:17-47, Fig. 3; *see also* Ex.1007, 5:38-67 (disclosing various techniques that port security processor 242 uses to learn IP addresses); Ex.1003, ¶163.

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 the CPU 406 to learn the new IP address and dynamically update the ACL. Ex.1007, 9:3-13 ("After the source IP address has been learned…then the [ACL] table is programmed…and the packet is passed 422. If the reverse IP check is not successful then the received data packet is blocked 424, or dropped at the port."). 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; Ex.1003, ¶164.

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 this limitation. Ex.1003, ¶165.

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

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; Ex.1003, ¶¶166-197.

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; *see also* Ex.1006, 2:1-14 ("…**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."), 6:27-30 ("…the forwarding entity 404…[switches] network messages among the various interfaces 402a-e of device 316."); 7:66-8:6 ("…each interface 402a-e per direction (e.g., inbound or outbound)…"); Ex.1003, ¶168.

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 402*a-e* that correspond to the forwarding entity 404's ports ("*one of a plurality of ports*"). Ex.1003, ¶169.

"*receive packets on one of a plurality of ports*"

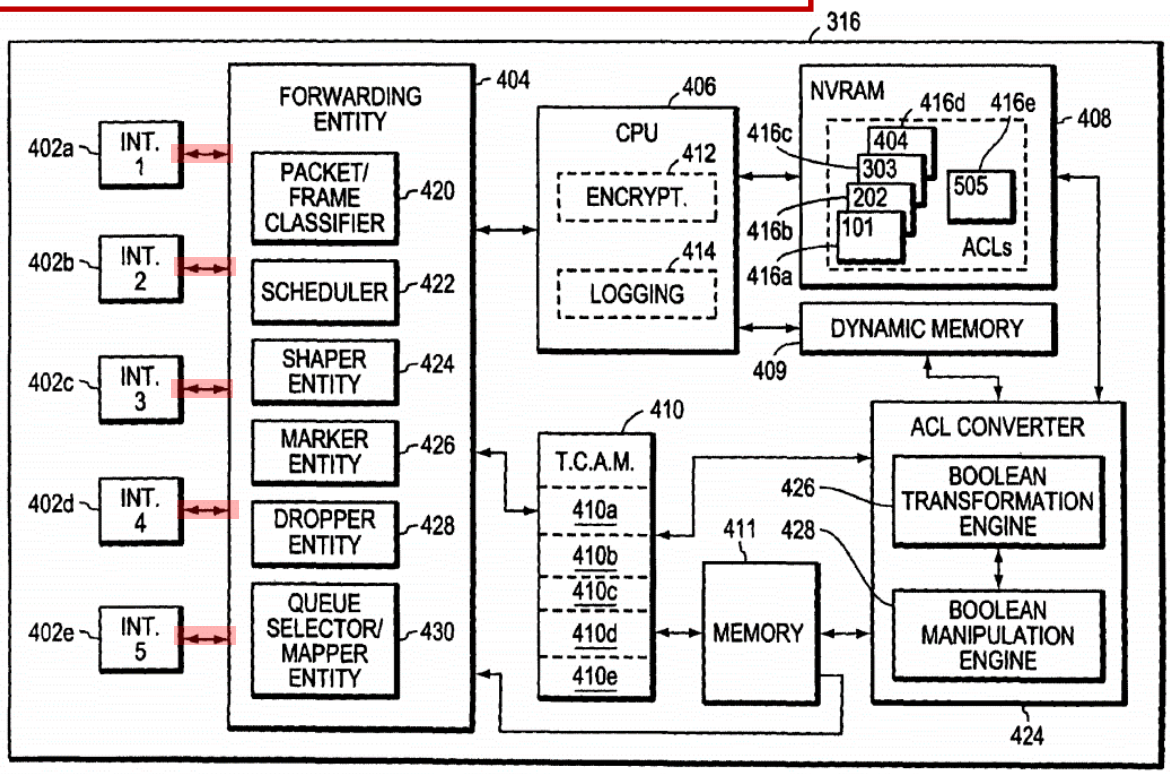


FIG. 4

Ex.1006, Fig. 4 (annotated).

Third, Gai in combination with Kwan renders obvious that the forwarding entity 404 ("*forwarding circuit*") receives packets on "*the processor port.*" As discussed at [15.1.1], in the combination of Gai and Kwan the forwarding entity 404 forwards to the CPU 406 specific packets. Ex.1006, 6:13-30, 11:60-12:7; Ex.1007, 5:30-36, Fig. 3; Ex.1003, ¶170.

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

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. Ex.1003, ¶171.

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 the CPU 406 ("*operable to receive packets on … the processor port*."). Ex.1003, ¶172.

"receive packets on… the processor port"

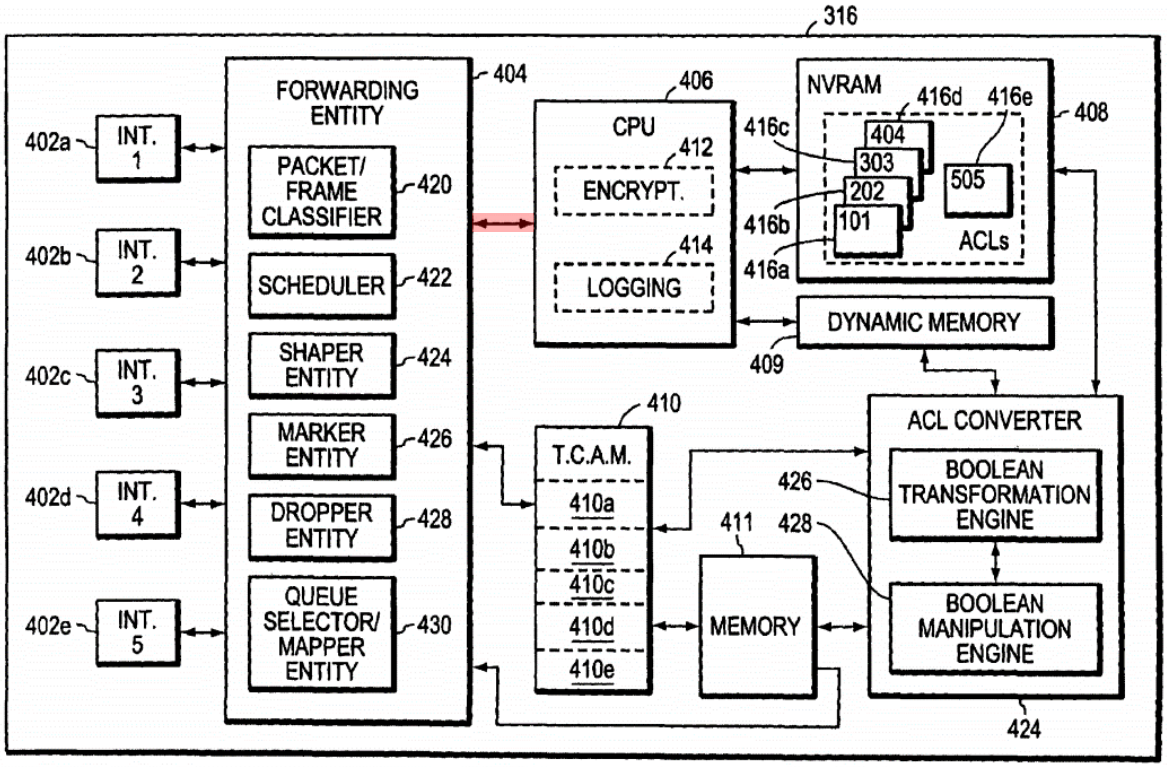


FIG. 4

Ex.1006, Fig. 4 (annotated).

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 the CPU 406 port, which renders obvious this limitation. Ex.1003, ¶173.

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

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. For example, Gai discloses that "the **forwarding entity 404 is basically**

configured to forward or switch network messages among the various interfaces 402a-e of device 316" and that "[e]**ach interface 402a-e, moreover, may be associated with one or more physical ports**." Ex.1006, 5:60-66, 6:27-30. To facilitate switching, the forwarding entity uses the IP destination address contained in the received message. Ex.1006, 2:1-51 ("The switching function includes receiving message…and transferring them to at least one destination port for receipt by another entity…**messages should be forwarded…based on certain predefined criteria… IP destination address**…."); Ex.1006, 19:6-14 ("Forwarding entity, 404 also extracts the necessary **information from the message (such as IP ... destination addresses**, etc.)"); Ex.1003, ¶¶174-175.

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 message (e.g., 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; Ex.1003, ¶176.

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 402a-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*"). Ex.1003, ¶114.

"forward each received packet to a port"

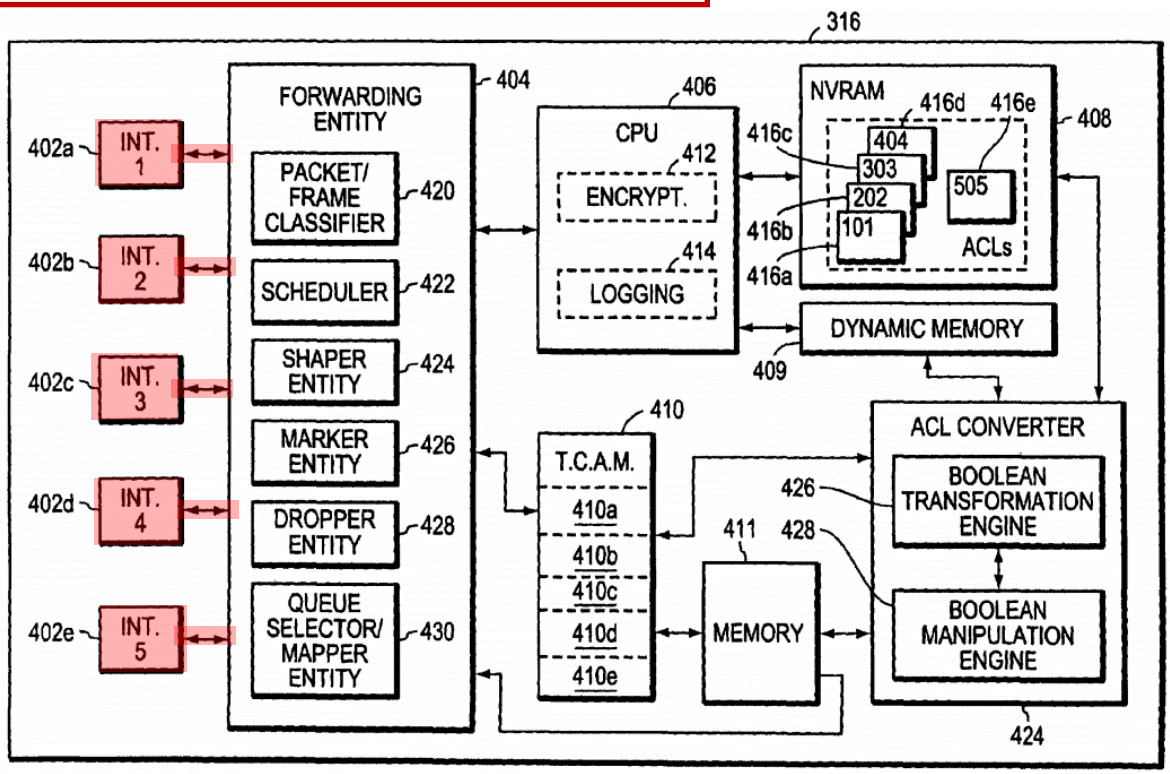


FIG. 4

Ex.1006, Fig. 4 (annotated).

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. Ex.1003, ¶¶177-178.

Thus, Gai renders obvious that the forwarding entity 404 is operable to forward each incoming packet (e.g., from the input ports or the CPU port) to an output port corresponding to an IP destination address extracted from the packet, which renders obvious this limitation. Ex.1003, ¶179.

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

First, as discussed at [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*." Ex.1003, ¶¶180-181.

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=l00) and the action corresponds to the ACL output (e.g., permit, deny, etc.)"); Figs. 5A-E, 8; *see also* Ex.1006, 17:40-48; Ex.1003, ¶182.

During operation, the TCAM ACL is applied to packets and the specified action (e.g., "forward," "discard," "permit," "deny," etc.) is implemented by the forwarding entity 404. Ex.1006, 7:37-39, 17:28-48; *see also* Ex.1006, 4:26-30, 8:7-27, Figs. 5A-E, 10A-11. Each of Gai's actions that restricts access, separately and in combination, correspond to "*access restrictions*" as claimed. Ex.1003, ¶¶183-184.

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, in Kwan, 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. Ex.1003, ¶185.

In the proposed combination, when packets are identified as having a 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 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; Reasons to Combine Kwan with Gai; Ex.1003, ¶186.

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 this limitation. Ex.1003, ¶187.

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

Gai discloses "*a memory circuit coupled to the forwarding circuit*" by disclosing "**non-volatile random access memory (NVRAM) 408**" and "**dynamic memory 409**" and "**TCAM 410**" that are coupled to the forwarding entity 404 via the CPU 406. 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; Ex.1003, ¶¶188-189.

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

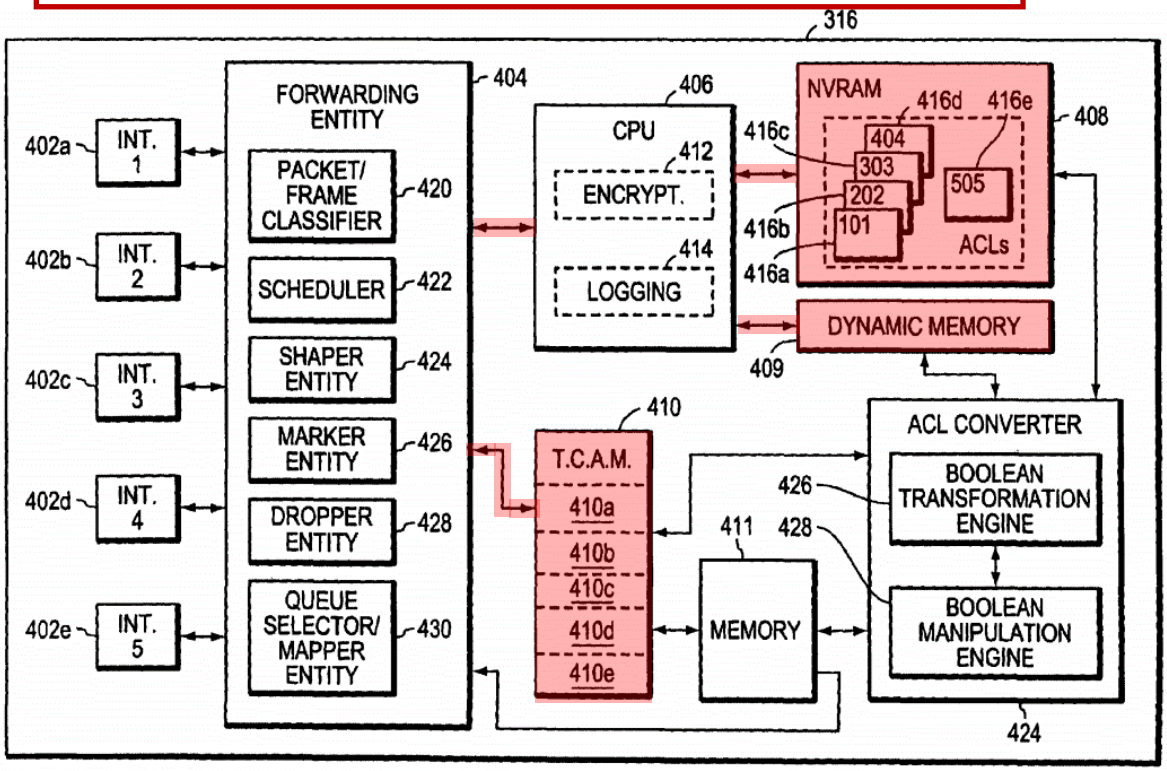


FIG. 4

Ex.1006, Fig. 4 (annotated).

A POSITA would have understood that because Gai's NVRAM 406 and dynamic memory 409 are coupled to the CPU 406, which is coupled to the forwarding entity 404, both NVRAM 406 and dynamic memory 409 are coupled, via CPU 406, to the forwarding entity 404. Accordingly, Gai's NVRAM 406 and dynamic memory 409, collectively, correspond to "*a memory circuit coupled to the forwarding circuit*," as claimed. Ex.1003, ¶¶190-191.

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*." Ex.1003, ¶190.

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 this limitation. Ex.1003, ¶191.

[15.2.2] ***the memory circuit operable to store packets and***

First, as discussed at limitation [15.2.1], Gai's dynamic memory 409 corresponds to part of the claimed "*memory circuit.*" Ex.1003, ¶¶192-193.

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, 1:13-21, 3:17-22. Georgiou's memory is dynamic memory. Ex.1008, 5:51-67; Ex.1003, ¶194.

Consistent with the analysis at [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; Ex.1003, ¶195.

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 this limitation. Ex.1003, ¶196.

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

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*." Ex.1003, ¶¶197-198.

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; Ex.1003, ¶199.

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 this limitation. Ex.1003, ¶200.

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

First, as discussed at [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*." Ex.1003, ¶¶201-202.

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, 20:65-67; Ex.1003, ¶203.

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 this limitation. Ex.1003, ¶204.

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

As discussed at [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."); Ex.1003, ¶¶205-206.

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.1003, ¶207.

CPU 406 is coupled to the Forwarding Entity 404 and to NVRAM 408 and Dynamic Memory 409
"a processor coupled to the forwarding circuit and to the memory circuit"

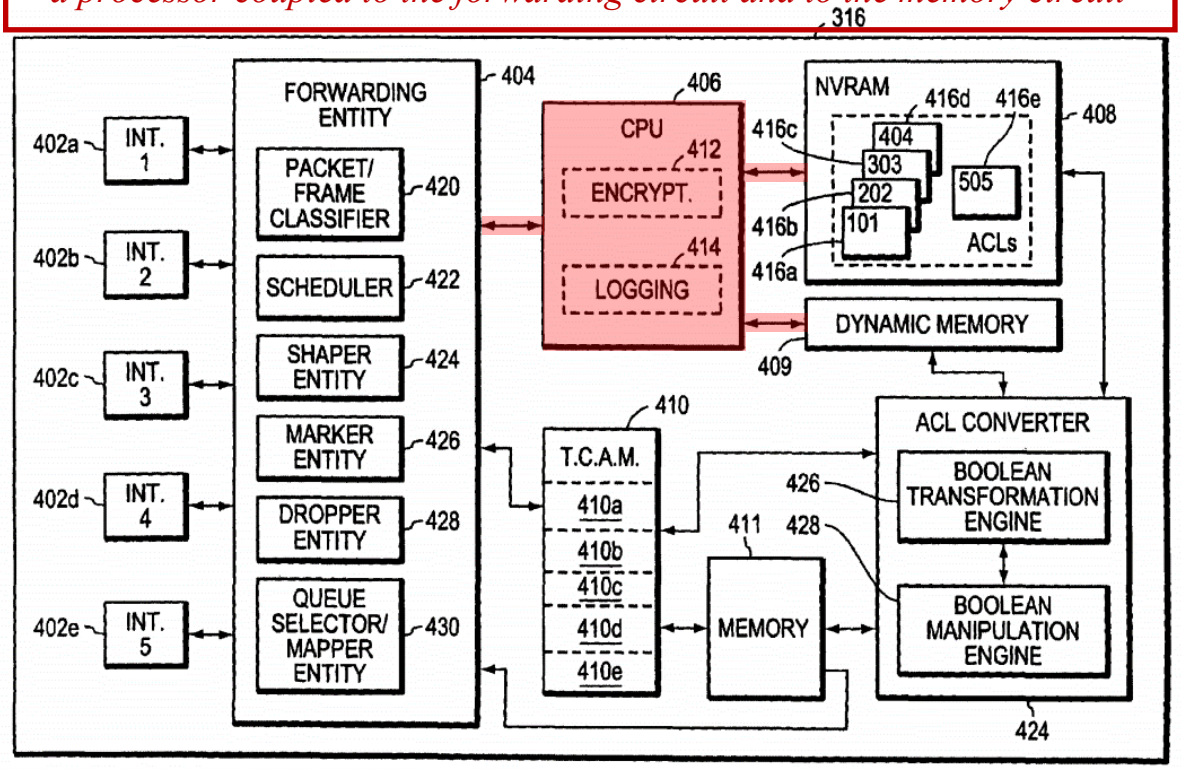


FIG. 4

Ex.1006, Fig. 4 (annotated).

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 this limitation. Ex.1003, ¶208.

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

First, as discussed at [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*." Ex.1003, ¶¶209-210.

Second, as discussed at [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. Ex.1003, ¶211.

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 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. Ex.1003, ¶212.

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 this limitation. Ex.1003, ¶213.

[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***

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 [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*." Ex.1003, ¶214.

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."). Ex.1003, ¶215. Further, consistent with the analysis at [15.1.1]-[15.1.2], Kwan discloses that port security processor 242 is used to learn the new IP addresses. Ex.1007, 4:56-58, 5:38-67, 8:17-47, Fig. 3.

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* [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 [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 [1.1] and [1.2.4]; Ex.1007, 6:14-25. In the instance where the new IP address is consistent with the seed IP address properties, then the new IP address would be learned and stored in the one or more text-based ACLs. *See* [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 [1.0] and [1.3]. *See* Reasons to Combine Kwan with Gai; Ex.1003, ¶216.

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 this limitation. Ex.1003, ¶217.

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

First, as discussed at [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*." Ex.1003, ¶¶218-219.

Second, as discussed at [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. Ex.1003, ¶220.

Thus, Gai discloses that CPU 406 is operable to store the dynamically generated ACL in TCAM 410, which renders obvious this limitation. Ex.1003, ¶221.

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

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. Ex.1003, ¶¶222-223.

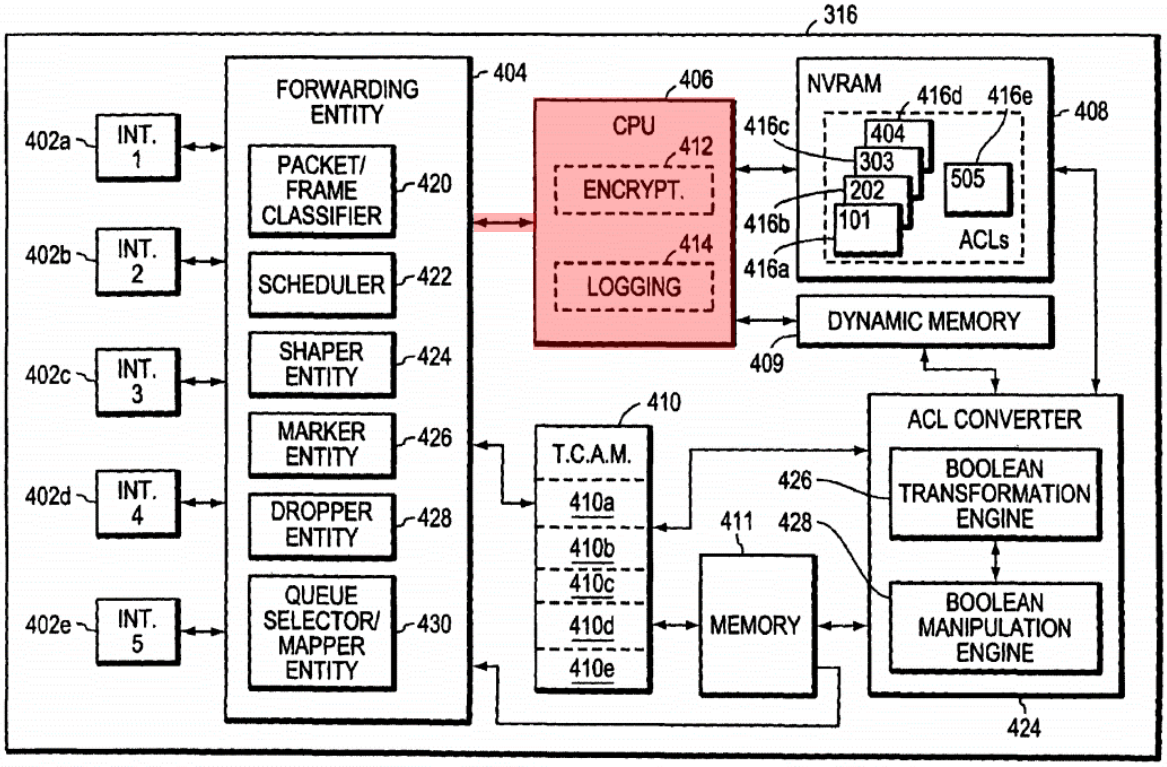
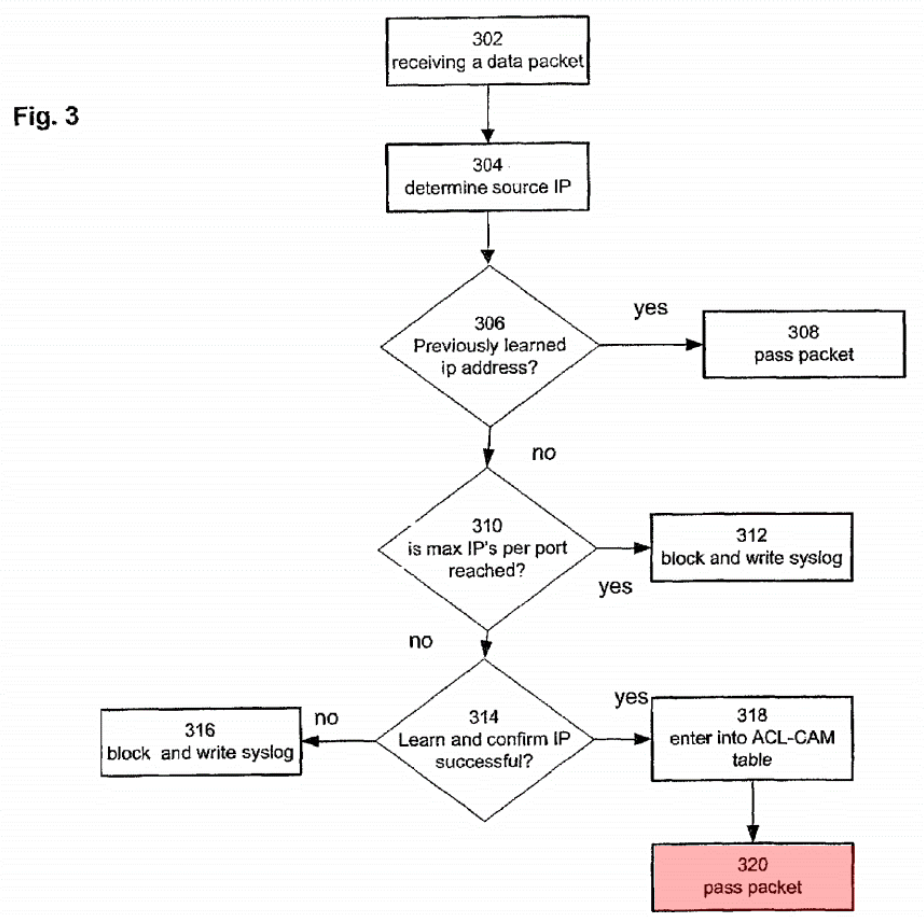


FIG. 4

Ex.1006, Fig. 4 (annotated).

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). Ex.1003, ¶224.



Ex.1007, Fig. 3 (annotated).

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; Ex.1003, ¶225.

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 this limitation. Ex.1003, ¶226.

4. Claim 17

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

Consistent with the analysis at [15.0]-[15.1.1], the prior art discloses that the network device 316 switches Ethernet packets on computer network 300 that uses a "LAN standard (e.g., **Ethernet**)." Ex.1006, 2:14-16, 5:3-64; *see also* Ex.1007, 1:32-43; Ex.1003, ¶227. 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*." Gai's disclosure of Ethernet LAN is consistent with the '518 patent's embodiment. *See* Ex.1001, 1:5-21; Ex.1003, ¶¶228-229.

5. Claim 19

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

To the extent the preamble is limiting, as discussed at limitation [15.0], Gai discloses a computer network 300. Ex.1003, ¶230.

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

As discussed at [1.2.1] and [15.0], Gai discloses a network device 316 that

includes circuitry for switching. Ex.1003, ¶231.

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

See [15.1.1]. Ex.1003, ¶232.

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

See [15.1.2]. Ex.1003, ¶233.

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

See [15.1.3]. Ex.1003, ¶234.

[19.1.5] ***to forward each received packet to a port corresponding to a destination address contained in the packet***

See [15.1.4]. Ex.1003, ¶235.

[19.1.6] ***subject to access restrictions contained in a dynamic access control list;***

See [15.1.5]. Ex.1003, ¶236.

[19.2.1] ***a memory circuit coupled to the forwarding circuit,***

See [15.2.1]. Ex.1003, ¶237.

[19.2.2] ***the memory circuit operable to store packets and***

See [15.2.2]. Ex.1003, ¶238.

[19.2.3] [the memory circuit]…*operable to store an enhanced access control list and*

See [15.2.3]. Ex.1003, ¶239.

[19.2.4] [the memory circuit…operable to store]…*a dynamic access control list; and*

See [15.2.4]. Ex.1003, ¶240.

[19.3.1] ***a processor coupled to the forwarding circuit and to the memory circuit,***

See [15.3.1]. Ex.1003, ¶241.

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

See [15.3.2]. Ex.1003, ¶242.

[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***

See [15.3.3]. Ex.1003, ¶243.

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

See [15.3.4]. Ex.1003, ¶244.

[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***

See [15.3.5]. Ex.1003, ¶245.

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

First, as discussed at [15.0] and [15.1.4], Gai's network device 316 includes ports, which discloses a "*a respective port of the network switch*." Ex.1003, ¶¶246-248.

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, 5:60-66, 17:4-6; 5:60-66; Ex.1003, ¶249.

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 402a-402e 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; Ex.1003, ¶250.

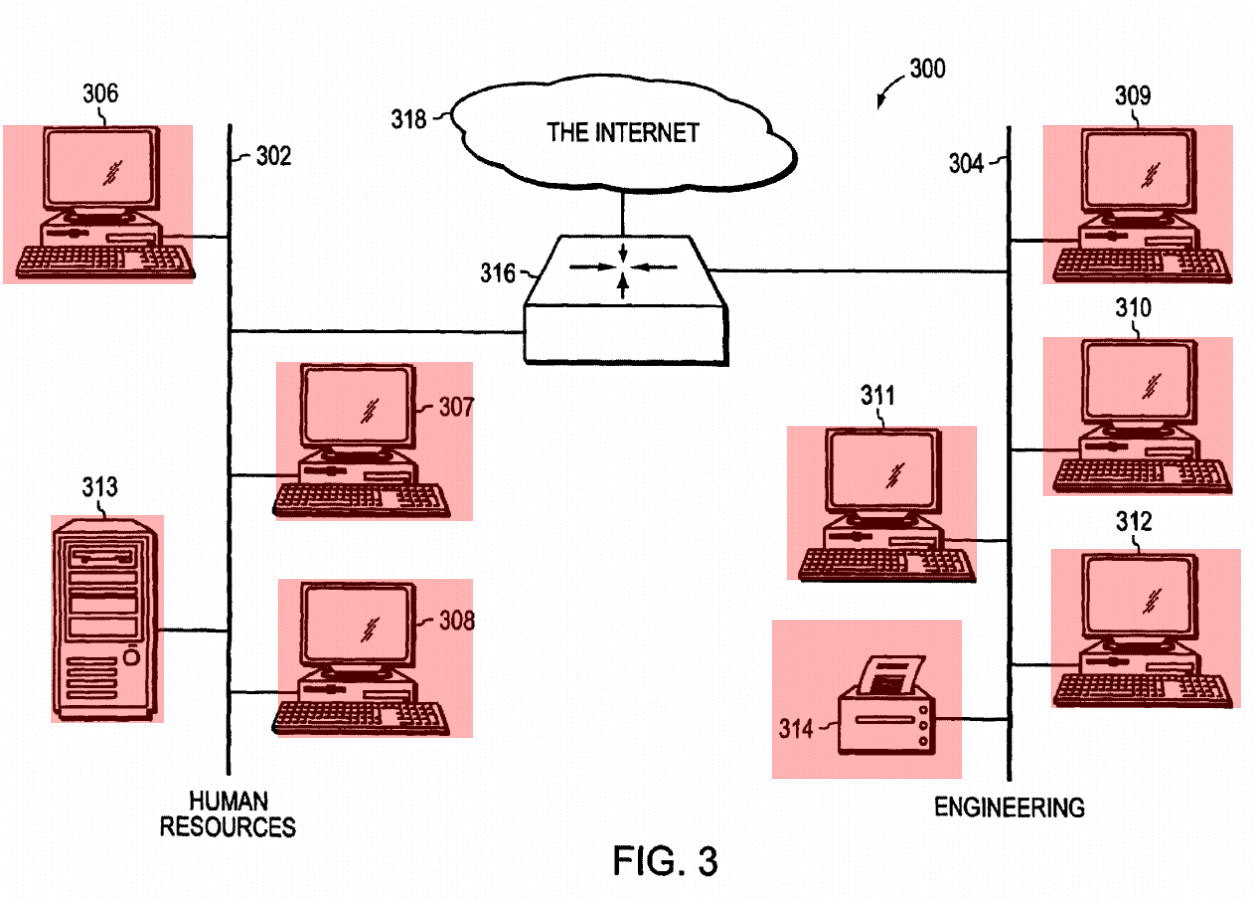


FIG. 3

Ex.1006, Fig. 3 (annotated).

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; Ex.1003, ¶251.

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 this limitation. Ex.1003, ¶252.

6. Claim 20

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

First, as discussed at [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."* Ex.1003, ¶¶2253-254.

Second, to the extent argued that Gai's end stations 306-312 do not sufficiently disclose "*personal computer systems*," Kwan expressly discloses that "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. Ex.1003, ¶255.

Thus, Gai in combination with Kwan discloses that at least some of Gai's end stations comprise personal computers, which renders obvious this limitation. Ex.1003, ¶256.

7. Claim 21

[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.***

As discussed at [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 this limitation. Ex.1003, ¶257.

X. DISCRETIONARY DENIAL IS INAPPROPRIATE

A. No *Fintiv* denial

The six factors considered for § 314 denial strongly favor institution. *See Apple Inc. v. Fintiv, Inc*., IPR2020-00019, Paper 11 (PTAB Mar. 20, 2020) (precedential).

1. No evidence regarding a stay

No motion to stay has been filed, so the Board should not infer the outcome of such a motion. This factor is neutral.

2. Parallel trial date

The co-pending district court case was filed May 9, 2023. Ex.1005. The current median time-to-trial in the Eastern District of Texas is 16.7 months. Ex.1015, 35. Accordingly, the projected trial date for *Fintiv* purposes is October of 2024—approximately 17 months after May 2023. Trial is currently scheduled,

however, for March 2025. Ex.1020, 1. The Board recognizes "that scheduled trial dates are unreliable and often change." *See* Director's June 21, 2022 Memorandum on Discretionary Denials ("Memo"), 8. And the Board has instituted cases with similar time differentials between trial date and final written decision, based on strong showing on the merits. *See Microsoft Corp. v. WSOU Investments, LLC*, IPR2021-00930, Paper 8, 6 (Dec. 2021) (instituting trial with a nine-month differential); *PEAG LLC v. Varta Microbattery GmBH*¸ IPR2020-01214, Paper 8, 19-20 (Jan. 2021) (instituting trial despite a seven-month time differential). Considering the Petition's strong showing on the merits, the Board should not discretionarily deny institution.

3. Investment in the parallel proceeding

The co-pending litigation is in its early stages, and the investment in it has been minimal. The parties have not exchanged preliminary positions on claim construction or expert reports. Ex.1020, 3-4.

4. Issue overlap

The district court litigation remains in its early stages, and Petitioner has not decided on the prior art or invalidity defenses it will present via motion or at trial. Thus, there is no present overlap of prior art issues, and this factor favors institution.

5. Parties

Petitioner is a defendant in the litigation. Ex.1005. This common circumstance is not a basis for denying institution.

6. Other circumstances

The merits of Petitioner's arguments are compelling, and this factor weighs against discretionary denial. *See* Memo, 4.

B. No §325(d) denial

The challenges presented are neither cumulative nor redundant to the '518 patent prosecution. The Board should not deny this petition under §325(d).

XI. CONCLUSION

Petitioner has established a reasonable likelihood that the Challenged Claims are unpatentable.

Respectfully submitted,

Dated: April 2, 2024
HAYNES AND BOONE, LLP
2801 N. Harwood St., Suite 2300
Dallas, Texas 75201
Customer No. 27683

/Theodore M. Foster/
Theodore M. Foster
Lead Counsel for Petitioner
Registration No. 57,456

XII. MANDATORY NOTICES

A. Real Party-in-Interest

Pursuant to 37 C.F.R. § 42.8(b)(1), Petitioner certifies that the real party-in-interest is Cisco Systems, Inc.

B. Related Matters

Pursuant to 37 C.F.R. § 42.8(b)(2), to the best knowledge of the Petitioner, the '518 patent is or was involved in the following case(s):

Case Heading	Number	Court	Filed
Lionra Technologies Limited v. CrowdStrike Holdings, Inc. et al	*1-23-cv-00928*	WDTX	Aug. 7, 2023 (terminated Nov. 6, 2023)
Lionra Technologies Limited v. VMware, Inc.	*1-23-cv-00929*	WDTX	Aug. 7, 2023
Lionra Technologies Limited v. Palo Alto Networks, Inc.	*2-23-cv-00214*	EDTX	May 11, 2023 (terminated Jul. 10, 2023)
Lionra Technologies Limited v. Cisco Systems, Inc.	*2-23-cv-00207*	EDTX	May 9, 2023
Lionra Technologies Limited v. Fortinet, Inc.	*2-23-cv-00206*	EDTX	May 9, 2023

C. Lead and Back-up Counsel and Service Information

Lead Counsel
Theodore M. Foster
HAYNES AND BOONE, LLP
2801 N. Harwood St., Suite 2300
Dallas, TX 75219
Phone: (303) 382-6205
Fax: (214) 200-0853
ipr.theo.foster@haynesboone.com
USPTO Reg. No. 57,456

Back-up Counsel
David L. McCombs
HAYNES AND BOONE, LLP
2801 N. Harwood St., Suite 2300
Dallas, TX 75219
Phone: (214) 651-5533
Fax: (214) 200-0853
david.mccombs.ipr@haynesboone.com
USPTO Reg. No. 32,271

Gregory P. Huh
HAYNES AND BOONE, LLP
2801 N. Harwood St., Suite 2300
Dallas, TX 75219
Phone: (972) 739-6939
Fax: (214) 200-0853
gregory.huh.ipr@haynesboone.com
USPTO Reg. No. 70,480

Please address all correspondence in this proceeding to lead and back-up counsel. Petitioner consents to service in this proceeding by email at the addresses above.

CERTIFICATE OF WORD COUNT

Pursuant to 37 C.F.R. § 42.24(d), Petitioner hereby certifies, in accordance with and in reliance on the word count provided by the word-processing system used to prepare this Petition, that the number of words in this paper is 13,941. Pursuant to 37 C.F.R. § 42.24(d), this word count excludes the table of contents, table of authorities, mandatory notices under § 42.8, certificate of service, certificate of word count, appendix of exhibits, and any claim listing.

Dated: April 2, 2024

/Theodore M. Foster/
Theodore M. Foster
Lead Counsel for Petitioner
Registration No. 57,456

CERTIFICATE OF SERVICE

The undersigned certifies that, in accordance with 37 C.F.R. § 42.6(e) and 37 C.F.R. § 42.105, service was made on Patent Owner as detailed below.

Date of service	April 2, 2024
Manner of service	FEDERAL EXPRESS
Documents served	Petition for *Inter Partes* Review Under 35 U.S.C. § 312 and 37 C.F.R. § 42.104 of U.S. 7,623,518; Petitioner's Exhibit List; Exhibits 1001-1020; and Petitioner's Power of Attorney.
Persons served	Hewlett Packard Enterprise 3404 E. Harmony Road Mail Stop 79 Fort Collins, CO 80528

/Theodore M. Foster /
Theodore M. Foster
Lead Counsel for Petitioner
Registration No. 57,456