

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

CISCO SYSTEMS INC.,

Petitioner

IPR2024-01281

U.S. Patent No. 7,738,471

**DECLARATION OF NADER F. MIR, PH.D.,
UNDER 37 C.F.R. § 1.68 IN SUPPORT OF PETITION
FOR *INTER PARTES* REVIEW**

TABLE OF CONTENTS

I.	INTRODUCTION	4
II.	QUALIFICATIONS AND PROFESSIONAL EXPERIENCE	6
III.	LEVEL OF ORDINARY SKILL IN THE ART	10
IV.	RELEVANT LEGAL STANDARDS	11
V.	BACKGROUND	12
A.	TCP/IP and OSI Models	12
B.	Packet Encapsulation	15
VI.	OVERVIEW OF THE '471 PATENT	16
VII.	CLAIM CONSTRUCTION	19
VIII.	IDENTIFICATION OF HOW THE CLAIMS ARE UNPATENTABLE....	20
IX.	THE COMBINATION OF CORNETT, PAATELA, AND NELSON RENDERS OBVIOUS CLAIMS 1-3, 5-10, 12-15, 17-21	20
A.	Summary of Cornett	20
B.	Summary of Nelson	23
C.	Summary of Paatela	25
D.	Reasons to Combine Cornett, Paatela, and Nelson.....	28
1.	Efficient Data Storage (Cornett and Paatela)	30
2.	Efficient Data Movement (Cornett, Paatela, and Nelson)	33
E.	Detailed Analysis of Claims	36
1.	Claim 13	36
2.	Claim 14	50
3.	Claim 15	52
4.	Claim 17	54
5.	Claim 18	55
6.	Claim 19	56
7.	Claim 20	58

8.	Claim 21	59
9.	Claim 1	61
10.	Claim 2	63
11.	Claim 3	63
12.	Claim 5	63
13.	Claim 6	64
14.	Claim 7	64
15.	Claim 8	64
16.	Claim 9	66
17.	Claim 10	66
18.	Claim 12	66
X.	THE COMBINATION OF CORNETT, PAATELA, NELSON, AND RUSSELL RENDERS OBVIOUS CLAIM 8	67
A.	Summary of Russell.....	67
B.	Reasons to Combine Cornett, Paatela, Nelson, and Russell.....	69
C.	Detailed Analysis of Claim.....	74
1.	Claim 8	74
XI.	CONCLUSION.....	77

I, Nader F. Mir, Ph.D., do hereby declare as follows:

I. INTRODUCTION

1. I am making this declaration at the request of Cisco Systems, Inc. in the matter of the *Inter Partes* Review of U.S. Patent No. 7,738,471 (“the ’471 patent”) to Lee.

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

3. I have been asked to provide my opinions regarding whether the subject matter of claims 1-3, 5-10, 12-15, 17-21 (“the Challenged Claims”) of the ’471 patent would have been obvious to a person having ordinary skill in the art (“POSITA”) at the time of the alleged invention, in light of the prior art. It is my opinion that the Challenged Claims would have been obvious to a POSITA.

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

- a. the ’471 patent, Ex.1001;
- b. the prosecution history of the ’471 patent (“’471 File History”), Ex.1002;
- c. U.S. Patent Appl. No. 2006/0072564 to Cornett et al. (“Cornett”), Ex.1005;
- d. U.S. Patent No. 7,333,489 to Nelson et al. (“Nelson”), Ex.1006;

- e. U.S. Patent Appl. No. 2006/0209840 to Paatela et al. (“Paatela”), Ex.1007;
and
- f. U.S. Patent No. 6,678,746 to Russell et al. (“Russell”), Ex.1025.

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

- a. Transmission Control Protocol, RFC 793 (Sept. 1981), Ex.1008;
- b. Andrew Tanenbaum, “Computer Networks,” 3rd ed. (1996) (“Tanenbaum”), Ex.1009;
- c. U.S. Patent Appl. No. 2006/0215695 to Olderdissen (“Olderdissen”), Ex.1010;
- d. U.S. Patent No. 7,821,931 to Swenson et al. (“Swenson”), Ex.1011;
- e. U.S. Patent No. 8,396,064 to Giesberts et al. (“Giesberts”), Ex.1012;
- f. U.S. Patent No. 9,307,054 to Boucher et al. (“Boucher”), Ex.1013;
- g. U.S. Patent No. 6,175,571 to Haddock et al. (“Haddock”), Ex.1014;
- h. U.S. Patent No. 6,438,128 to Kashyap (“Kashyap”), Ex.1015;
- i. U.S. Patent No. 6,501,764 to Fudatate et al. (“Fudatate”), Ex.1016;
- j. U.S. Patent No. 7,110,404 to Temoshenko (“Temoshenko”), Ex.1017;
- k. U.S. Patent Appl. No. 2003/0223376 to Elliott et al. (“Elliott”), Ex.1018;

- l. U.S. Patent Appl. No. 2005/0256975 to Kaniz et al. (“Kaniz”), Ex.1019;
- m. U.S. Patent. No. 6,650,654 to Batty et al. (“Batty”), Ex.1020;
- n. Radia Perlman, “Interconnections: Bridges, Routers, Switches, and Internetworking Protocols,” 2d ed. (1999) (“Perlman”), Ex.1023;
- o. U.S. Patent Appl. No. 2002/0003795 to Oskouy et al. (“Oskouy”), Ex.1024;
and
- p. WO 00/10302 to Lawton et al. (“Lawton”), Ex.1026.

6. Unless otherwise noted, all emphasis in any quoted material has been added. Claim terms are italicized.

II. QUALIFICATIONS AND PROFESSIONAL EXPERIENCE

7. My complete qualifications and professional experience are described in my *Curriculum Vitae*, a copy of which can be found in Exhibit 1004. The following is a brief summary of my relevant qualifications and professional experience.

8. I am a tenured professor in the Electrical Engineering Department of San Jose State University, California, where I also held the position of Associate Chair from 2006 through 2008. I have been a faculty member at San Jose State University since 2001.

9. In 1995, I received a Ph.D. in electrical engineering from Washington University, St. Louis. I received an M.S. degree in electrical engineering in 1990,

also from Washington University. In 1985, I received a B.S. in electrical engineering from Polytechnic University.

10. I have over 30 years of experience with computer networks and communication systems, including both industry and academic experience. Before my current position at San Jose State University, I was an Assistant Professor in the Department of Electrical and Computer Engineering at the University of Kentucky. I joined San Jose State University in 2001 as an Associate Professor. Since 2005, I have been a Full Professor in the Electrical Engineering Department at San Jose State University, where I perform research and teach in the areas of computer networks, networking devices, switches, routers, virtualization, virtual switches, cloud computing, software-defined networking in clouds, and cloud data centers.

11. I hold several technical editorial positions for various journals, including IEEE Communication Magazine. As a Technical Editor of IEEE Communications Standards Magazine for Cloud and Edge Computing, I am responsible for accepting or rejecting scientific articles submitted to the journal in the areas of computer networking focusing on cloud systems. From 2005 through 2020, I held the technical editorial position for IEEE Communications Magazine. I am a senior member of the IEEE and have served as a member of the technical program committees and the steering committees for a number of major IEEE communications and networking conferences.

12. For more than 30 years, I have conducted research in the areas of computer and communication networks; networking devices; protocols including (but not limited to) packet-switched networks, cloud and edge computing, software-defined networking in cloud systems, integrated voice, video, data center networks, TCP/IP, client/server operations, content delivery networks, multimedia and streaming, client/server, networking traffic flow analysis, wireless/mobile networks, networking devices, network security, and network virtualization, among others.

13. I have received a number of university awards as well as national and international awards. In particular, I have received a number of research grants from private, state, and governmental funding agencies for conducting research in the fields of computers and communication networks. I am also the recipient of several university teaching recognition awards from San Jose State University; several research excellence awards; and the school's published book award for the year.

14. Before my positions in academia, I worked in various positions in the telecommunication and computer network industries. From 1985 to 1988, I was a Telecomm R&D Engineer at the Telecommunication Research & Development Center (TRDC) in Surrey, England. In particular, I headed engineering projects in the area of telecommunication engineering. I also participated in the design of a high-speed switch for digital Private Branch Exchange (PBX) and SS7 networks. In 1991, I worked as an Electrical Engineering Consultant for Bussmann Company,

where I instructed engineers on electrical engineering topics and prepared them for the National Principles and Practice of Engineering (PE) exam. From 1989 to 1995, I worked as a Computer/Communications Chip-Design Engineer at the Computer & Communications Research Center in St. Louis, Missouri. In that position, I designed a VLSI gigabit switch control plane chip for a high-speed computer network using CMOS technology. From 1995 to 1996, I was a Research Scientist Engineer at Advanced Telecommunications Institute, Stevens Institute of Technology in Hoboken, New Jersey. There I headed analyzing communications and high-speed computer networking projects.

15. Since obtaining my Ph.D., I have served in various consulting positions in the area of computer networking technology and communication engineering, including for CITS Inc. in San Jose, California (2012-2013), Morrison & Foerster LLP in San Francisco, California (2015-2016), and Butler and Snow in Jackson, Mississippi (2017-2018).

16. I have authored or co-authored over 100 articles in peer-reviewed journals, conference proceedings, texts, and monographs in the aforementioned areas of computer networking, communications, and related fields. I have published two textbooks, one of which is entitled “Computer Communication Networks.” This textbook has been adopted by numerous universities worldwide.

17. I am a named inventor on one U.S. patent, which relates to a packet-switching network and method of routing packets.

18. I have served as an expert witness and technical consultant in litigation and *inter partes* review matters concerning computer networks, wireless networks at both the protocol and hardware levels, and telecommunications.

III. LEVEL OF ORDINARY SKILL IN THE ART

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

20. A POSITA in the field of the '471 patent, as of its priority date of September 14, 2007, would have been someone knowledgeable and familiar with network communications techniques and high speed packet processing technologies available in the mid-2000s. Such a POSITA would have a bachelor's degree in computer science, computer engineering, electrical engineering, or equivalent training, and approximately two years of experience working in the field of network communications. Additional work experience can substitute for specific educational background, and vice versa.

21. For purposes of this Declaration, in general, and unless otherwise noted, my statements and opinions, such as those regarding my own experience and what a POSITA would have understood or known generally (and specifically related to the references I consulted herein), reflect the knowledge that existed in the relevant field as of the priority date of the '471 patent.

IV. RELEVANT LEGAL STANDARDS

22. I am not an attorney. In preparing and expressing my opinions and considering the subject matter of the '471 patent, I am relying on certain basic legal principles that Cisco's counsel has explained to me.

23. I understand that prior art to the '471 patent includes patents and printed publications in the relevant art that predate the priority date of the '471 patent. For purposes of this Declaration, I am applying September 14, 2007, as the priority date of the '471 patent.

24. I have been informed by Cisco's counsel that a claimed invention is unpatentable under 35 U.S.C. §103 if the differences between the claimed invention and the prior art are such that the subject matter as a whole would have been obvious to a POSITA at the time of the invention. I have also been informed by Cisco's counsel that the obviousness analysis considers factual inquiries, including the level of ordinary skill in the art, the scope and content of the prior art, and the differences between the prior art and the claimed subject matter.

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

V. BACKGROUND

A. TCP/IP and OSI Models

26. Computer networking is a complex task, one that is made easier by partitioning it "into smaller subtasks, traditionally by dividing the problem into layers." Perlman, 1. The idea behind layering is that "each layer is responsible for providing a service to the layer above by using the services of the layer below." Perlman, 1.

27. The basic structure of communication networks is represented by the Transmission Control Protocol/Internet Protocol (TCP/IP) model. Tanenbaum, 28.

This model is structured in four layers: link, network, transport, and application. Olderdisen, [0009]-[0013]. The TCP/IP model also includes a physical layer below the link layer.

28. The physical layer defines electrical aspects of activating and maintaining physical links in networks. The layer represents basic network hardware such as cables, connectors, and other hardware used to communicate in a network. Giesberts, 1:65-67; Perlman, 2. The link layer provides reliable synchronization and transfer of information across the physical layer for accessing the transmission medium, e.g., NIC card drivers. Olderdisen, [0013]. It specifies how packets access links and are attached to additional headers to form frames when entering a new networking environment, such as a LAN. The network layer specifies the networking aspects. This layer handles the way that addresses are assigned to packets and the way that packets are supposed to be forwarded from one end point to another. Olderdisen, [0012]. The transport layer handles the details of data transmission and is implemented in the end points (source and destination machines). Olderdisen, [0011]. It acts as an interface protocol between a communicating device and a network. The application layer determines how a specific user application should use a network and interface with the user. Olderdisen, [0010]. Such applications include *Simple Mail Transfer Protocol* (SMTP), *File Transfer Protocol* (FTP), and *World Wide Web* (WWW). Perlman, 4.

29. One of the most common sets of layers is the Open Systems Interconnection (“OSI”) model. It was the original standard description for how packets should be transmitted between any two points. Tanenbaum, 28, 36. The OSI model has seven layers: physical, data link, network, transport, session, presentation, and application. This arrangement is depicted below:

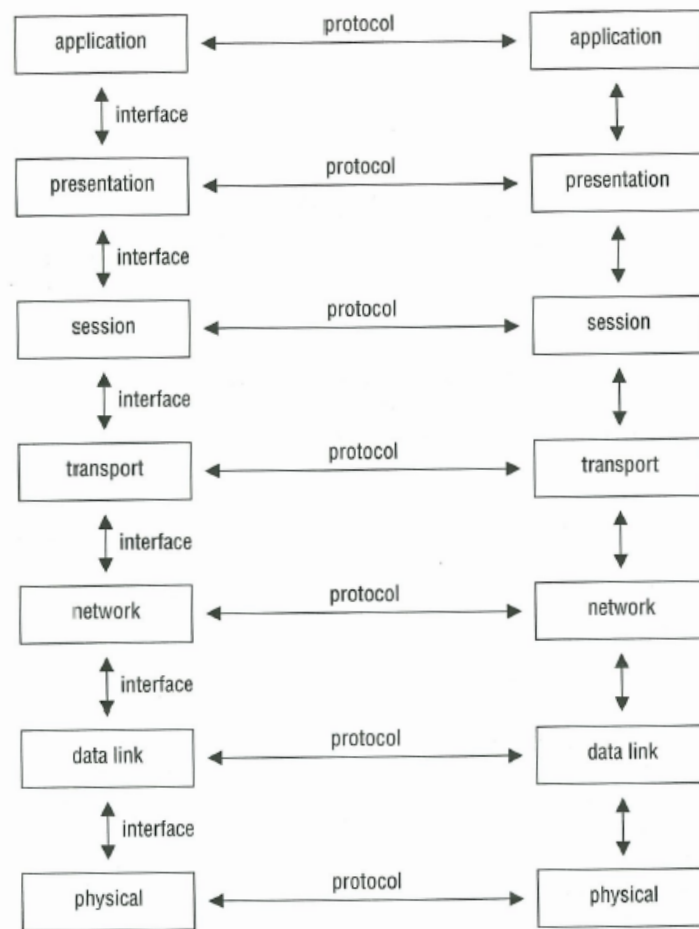


Figure 1.1 The OSI Reference Model

Perlman, 2

30. The OSI model expands TCP/IP model layers 3 and 4 to include two additional layers: the session layer and the presentation layer. Olderdissen, [0011]

(“TCP/IP application layer corresponds to the combination of the application, presentation and session layers of the OSI model.”).

31. The session layer sets up and coordinates sessions between different machines and the applications at each end. Tanenbaum, 50. Another basic task of this layer is “dialog control.” Tanenbaum, 50. The presentation layer converts incoming and outgoing data from one presentation format to another for various reasons such as data compression, data encryption, and data translation. Tanenbaum, 51. This layer of the protocol facilitates the interoperability between the different encoding methods. Tanenbaum, 51.

32. Each layer (irrespective of model) at a source node communicates with its peer layer at a destination node using a protocol. Perlman, 1. It is common for a single node to support several protocols and for a single layer to support several protocols as well. Perlman, 3.

B. Packet Encapsulation

33. In communications networks, a piece of data is normally fragmented into smaller pieces, and then each piece of that data is encapsulated (incorporated) into a header. This encapsulation results in the basic Internet transmission packet. For example, a layer protocol receives a packet and adds a header to the information it received. Perlman, 4. The header is like an envelope for a letter, with information such as a source and destination addresses, that helps deliver the letter to its

destination. Perlman, 4. As the packet is passed down layer by layer, protocols operating within those lower layers may add their own headers to the packet. Tanenbaum, 19 (“A message, *M*, is produced by an application process running in layer 5 and given to layer 4 for transmission. Layer 4 puts a header in front of the message to identify the message and passes the result to layer 3. The header includes control information, such as sequence numbers, to allow layer 4 on the destination machine to deliver messages in the right order if the lower layers do not maintain sequence.”).

VI. OVERVIEW OF THE '471 PATENT

34. The '471 patent relates at a high level to processing network packets at an egress end user node (“EEUN”) “having a TCP/IP based packet interface.” '471 patent, Abstract, 4:62-5:2; *see also* '471 patent, 6:59-63 (discussing the OSI protocol stack, as an example). The EEUN also includes a decoder, direct memory access (“DMA”) device, packet buffer memory, and protocol stack layer memory. '471 patent, 5:3-9.

35. The decoder “performs actions involving the decoding of the encoded packet for placing the same in a form that is readable to a particular computer program installed on the EEUN.” '471 patent, 5:10-14. In one embodiment, the decoder “forward[s] a portion of a decoded packet...to the DMA device.” '471 patent, 5:18-21.

36. The '471 patent states that, in conventional EEUN's, a "packet is transferred from a MAC layer memory to a CPU main memory through a serial or parallel memory transfer performed by a CPU" where the CPU main memory is known as the "packet buffer memory." '471 patent, 5:47-51. Each protocol stack layer writes its respective header into the packet buffer memory. '471 patent, 5:51-54. Each protocol stack layer "processes its respective layer information in a sequential manner." '471 patent, 10:47-48. This prior art configuration is depicted in Figure 5, below. See '471 patent, 4:30-33 (Figure 5 "is a schematic illustration of a conventional egress end user node (EEUN)[.]"')

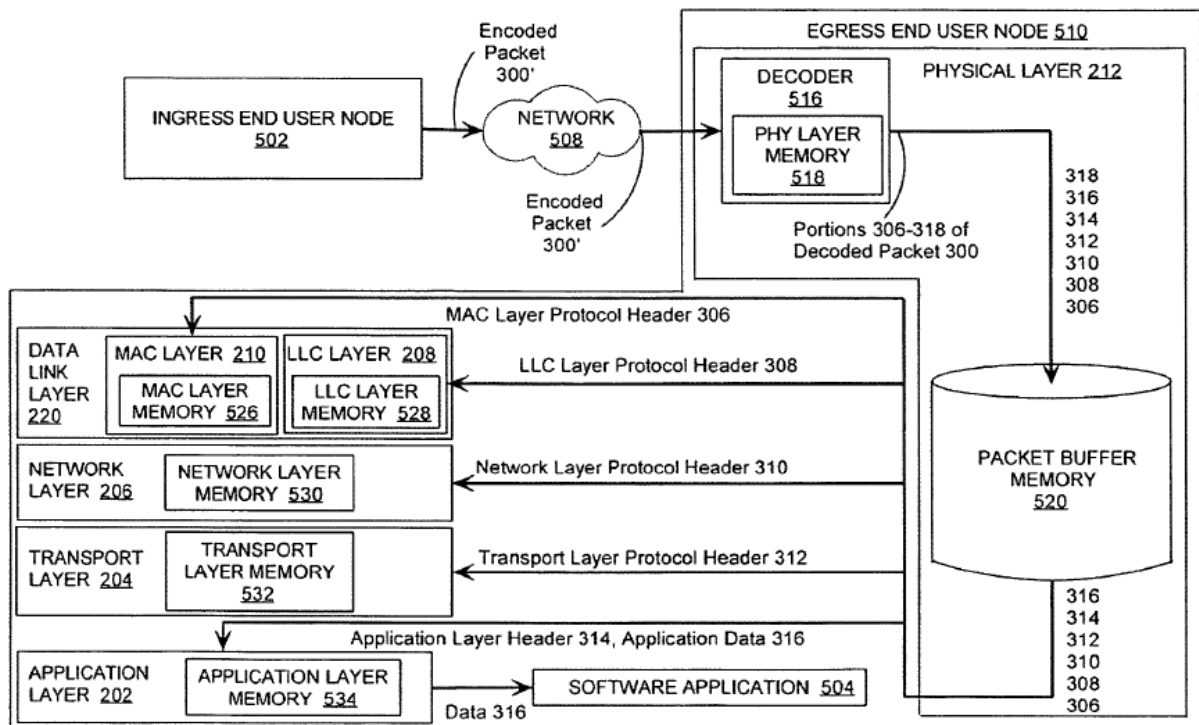
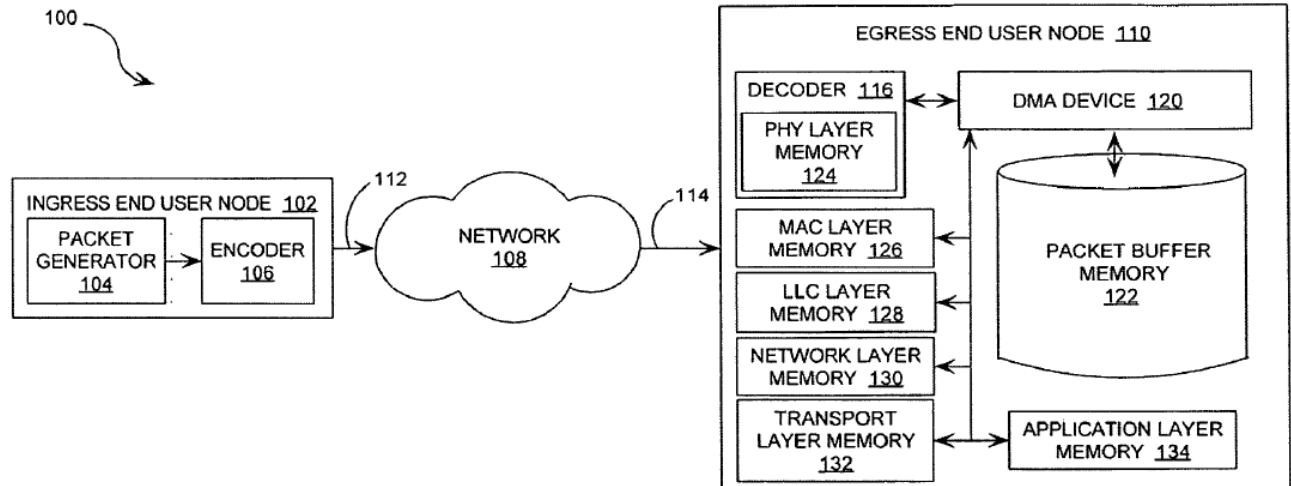


FIG. 5
(PRIOR ART)

'471 patent, Figure 5

37. While the '471 patent admits that “[s]ome conventional EEUN architectures do include one or more DMA devices,” the '471 patent teaches that the prior art DMA device(s) “are provided only for a one time transfer of a packet from the MAC layer memory to a CPU main memory.” '471 patent, 5:57-60. But, according to the '471 patent, a “single transfer does not solve or deal with the varying latencies and the increased processing consequences present in a high speed wireless network.” '471 patent, 5:60-63.

38. This alleged problem is solved by the '471 patent such that a DMA device “concurrently writ[e] (1) each of said plurality of headers to a packet buffer memory and (2) each individual one of said plurality of headers to a respective protocol stack layer memory[.]” '471 patent, claim 13 at 12: 38-43. The additional use of the DMA device purportedly “allow[s] the exchange of information between the protocol stack layers...[and] reduces processing latency and thereby increases bandwidth.” '471 patent, 6:1-6. This arrangement is depicted in Figure 1 which depicts a “block diagram of a packet-based communications system that is useful for understanding the present invention.” '471 patent, 4:20-22.



'471 patent, Figure 1

39. As I will explain below, these concepts were well known as of the priority date of the '471 patent.

VII. CLAIM CONSTRUCTION

40. It is my understanding that to properly evaluate the '471 patent, the terms of the claims must first be interpreted. I have been informed by Cisco's counsel that for the purposes of this *inter partes* review, the claims are to be construed under the so-called *Phillips* standard, under which claim terms are given their ordinary and customary meaning as would have been understood by a POSITA in light of the specification and prosecution history, unless the inventor has set forth a special meaning for a term. I have also been informed that claim terms only need to be construed to the extent necessary to resolve the obviousness inquiry.

41. I have reviewed the entirety of the '471 patent, as well as its prosecution history. It is my opinion that for purposes of applying the asserted prior art to

evaluate the patentability of the Challenged Claims, no terms require express construction.

VIII. IDENTIFICATION OF HOW THE CLAIMS ARE UNPATENTABLE

42. The discussion in this Declaration provides a detailed analysis of how the asserted prior art references teach each limitation of the Challenged Claims.

43. As part of my analysis, I have considered, and discuss in detail, the scope and content of the prior art and any differences between the alleged invention and the prior art.

44. It is my opinion that the alleged invention recited in the Challenged Claims would have been obvious in view of the teachings of the asserted prior art and the knowledge of a POSITA.

IX. THE COMBINATION OF CORNETT, PAATELA, AND NELSON RENDERS OBVIOUS CLAIMS 1-3, 5-10, 12-15, 17-21

A. Summary of Cornett

45. Like the '471 patent, Cornett is focused on “accelerated TCP (Transport Control Protocol) stack processing.” Cornett, [0003]. According to Cornett, overhead “associated with processing protocol stacks” may result from bottlenecks in the computer system from using the core processing module of a host processor to perform data movement. Cornett, [0004]-[0005]. This deficiency is addressed using a “DMA engine”: “TCP/IP processing may be accelerated by using a data movement module, such as a DMA engine, to move data from one buffer to another

buffer. Since the core processing module of a host processor may be bypassed using a DMA engine, slow memory access speeds may be avoided.” Cornett, [0097].

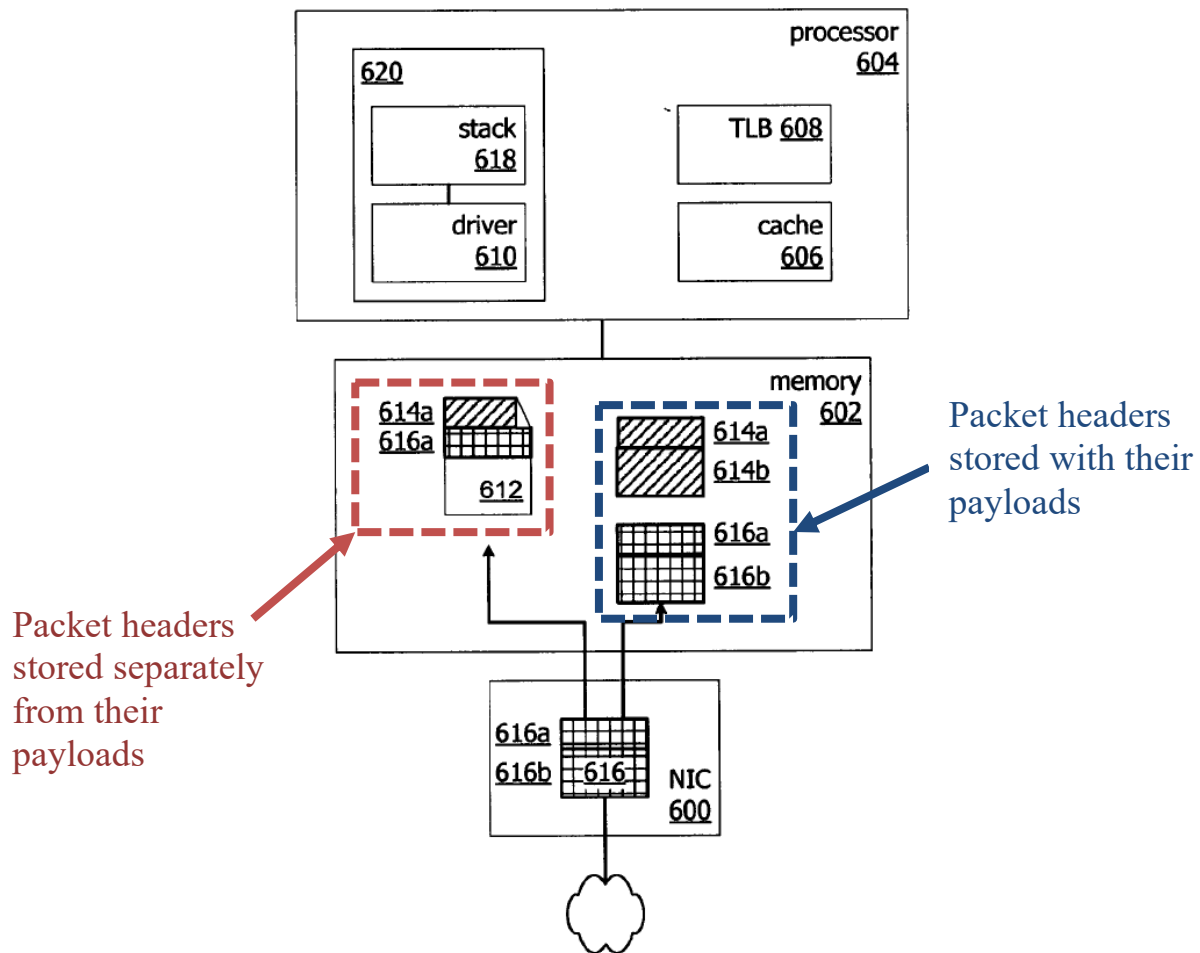
46. Cornett describes a network that is “comprise[d] [of] a plurality of nodes” where each node “may comprise one or more intermediate stations, such as, for example, one or more hubs, switches, and/or routers...[or] one or more end stations.” Cornett, [0021], [0023]; *see also* Cornett, [0038] (discussing the OSI model). Cornett’s nodes include a network interface controller (NIC) which can, in turn, include the following:

a physical layer device (PHY) that translates between the analog signals of a communications medium (e.g., a cable or wireless radio) and digital bits. The PHY may be communicatively coupled to a media access controller (MAC) (e.g., via a FIFO) that performs “layer 2” operations (e.g., Ethernet frame handling).

Cornett, [0079].

47. The NIC in Cornett’s nodes includes the functionality to perform a process called “header splitting.” Cornett, [0073]. For packets selected for header splitting, the NIC “can cause storage 704 (e.g., via Direct Memory Access (DMA)) of the packet’s header in the page(s) used to store headers and separately store 706 the packet’s payload.” Cornett, [0074]. In one of Cornett’s embodiments, the NIC “may store the header 614*a*, 616*a* and payload 614*b*, 616*b* in separate buffers, and additionally store the header 614*a*, 616*a* to a location in which the payload 614*b*,

616b is written. Put differently, the payload 614b, 616b may be stored to the first location as well as a second location different from the first location.” Cornett, [0081]. This is depicted in Figure 10, below:



Cornett, Figure 10 (annotated)

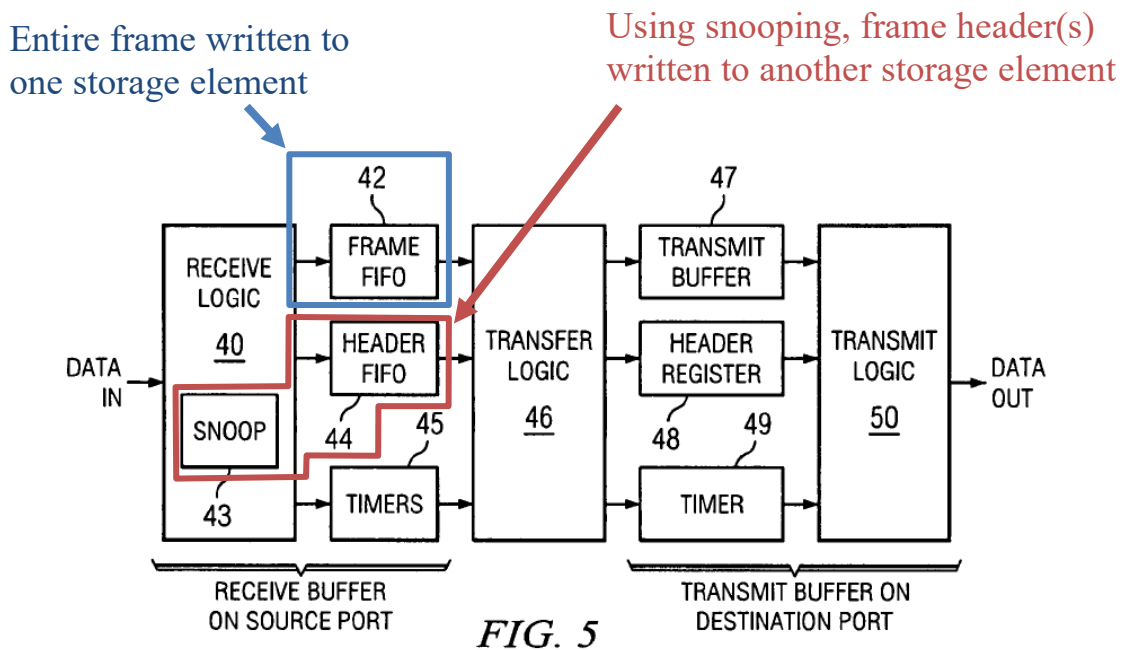
48. Cornett and the '471 patent are both directed to packet-based network communications with the same specific focus of accelerating packet header processing. See '471 patent, 1:8-11 (“invention relates to a method and apparatus for

high speed protocol header processing[.]”); Cornett, [0097] (“TCP/IP processing may be accelerated by using a data movement module, such as a DMA engine, to move data from one buffer to another buffer. Since the core processing module of a host processor may be bypassed using a DMA engine, slow memory access speeds may be avoided.”). In addition, Cornett and the ’471 patent address the same problem of how to accelerate data movement as part of packet processing using a DMA device. ’471 patent, 6:1-6; Cornett, [0097].

B. Summary of Nelson

49. Nelson relates to the use of “packet switching protocols wherein packet (frame) header information is stored in a buffer separate from the corresponding packet buffer to allow routing decisions to be made based on the header information without reading the packet out of the buffer.” Nelson, 1:13-20. In ordinary network operations, a switch “examine[s] the frame’s header to obtain the corresponding transport information.” Nelson, 1:54-57. This information must “be stored while the routing decision is made” and only after the routing decision has been made “can the frame be routed by the switch.” Nelson, 1:60-63. Because the frame must be read and stored in multiple locations while routing decisions are made, “the latency of the frame’s transport from the source device to the destination device is increased.” Nelson, 1:64-67.

50. Like the '471 patent, Nelson addresses this situation using receive logic in the port of a network switch that “stores frames of data in the storage element” and “concurrently snoops on the frame data to obtain header information” that is “stored in a buffer separate from” the memory that “stores the frames.” Nelson, 2:13-17. This arrangement is depicted in Figure 5, below:



Nelson, Figure 5 (annotated)

51. While Nelson describes its invention as using a FIFO to store data, Nelson explains that “dual port RAM can be used to store the header information” as well. Nelson, 6:8-9.

52. Nelson and the '471 patent are both directed to packet-based network communications with the same specific focus of accelerating packet header processing. *See* '471 patent, 1:8-11 (“invention relates to a method and apparatus for

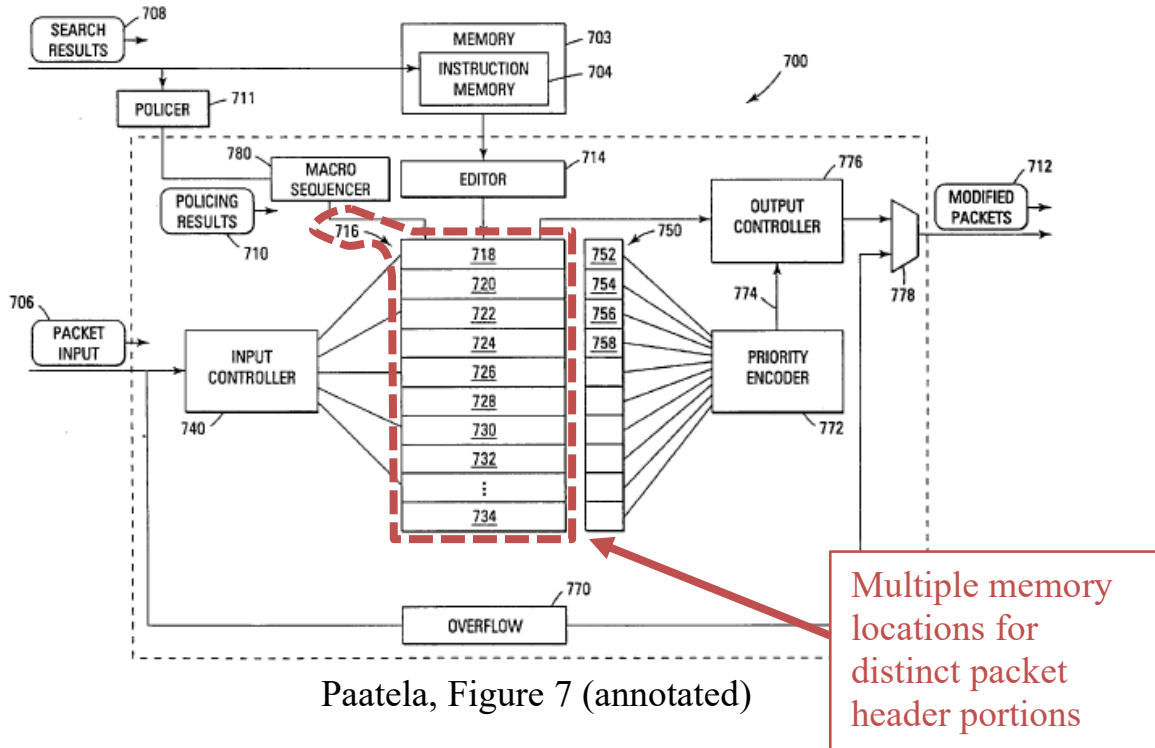
high speed protocol header processing at an intermediate and/or destination node of a packet-based communications network.”); Nelson, 2:13-17 (invention “eliminates the need to read a portion of the frame out of the FIFO and store this information in another buffer while a routing decision is made.”). In addition, Nelson and the ’471 patent address the same problem of how to accelerate packet header processing by storing packets and packet headers separately. *See* ’471 patent, 3:9-15 (“The concurrent writing step involves concurrently writing (1) each of the headers to a packet buffer memory and (2) each individual one of the headers to a respective protocol stack layer memory where it is available for immediate processing within a protocol stack layer.”); Nelson, 2:3-10 (“wherein the frames of data are stored in a first-in-first-out buffer and wherein the header information is accessed to make routing decisions in order to avoid having to read the frames out of the buffer.”).

C. Summary of Paatela

53. Paatela discloses a process for “packet transformation” where packets are “temporarily stored, and acted upon through processing of protocol-dependent instructions.” Paatela, Abstract. Paatela notes that network routers “take on the task of intercepting, analyzing, and moving on millions of packets per second along the best possible route” which requires that “processing occurring at these routers must be extremely efficient to avoid bogging down the system.” Paatela, [0041]. The disclosed process addresses this concern by providing a “packet transformation”

process at a network system. Paatela, [0015]. The network system can be implemented “at an intermediary network node between the source and destination nodes” or “at a network node,” including a switch. Paatela, [0016], [0017], [0043]; *see also* Paatela, [0007] (discussing the OSI model).

54. This system includes a memory 716 that “stores portions of the packet input 706, and stores these packet portions in memory locations 718-734 as dictated by the input controller 740.” Paatela, [0076]. Memory 716 is “partitioned such that at least some of these segments are allocated to store data corresponding to certain portions of the packet(s) 706.” Paatela, [0077]. In one embodiment, “these packet portions correspond to headers of the various protocol layers associated with the incoming packet.” Paatela, [0077]. The input controller directs this header information to memory 716 based on “upstream information developed by a packet classification engine that determines where one networking layer header ends and the next networking layer header begins.” Paatela, [0077]. Memory 716 is “configured and partitioned such that all header information and data is stored within the memory 716.” Paatela, [0085]. This is depicted in Figure 7, below:



Paatela, Figure 7 (annotated)

55. Paatela and the '471 patent are both directed to packet-based network communications with the same specific focus of accelerating packet header processing. *See* '471 patent, 1:8-11 ("invention relates to a method and apparatus for high speed protocol header processing at an intermediate and/or destination node of a packet-based communications network."); Paatela, [0041] ("The present invention may be used in connection with such routing systems, increasing speed and efficiencies of network data throughput."). In addition, Paatela and the '471 patent address the same problem of how to improve processing efficiency by processing headers of various protocol layers individually. *See* '471 patent, 5:64-6:6 (writing "specific layer header fields directly into: (a) each protocol stack layer's internal memory spaces...allowing the exchange of information between the protocol stack

layers...[and] reduc[ing] processing latency and thereby increases bandwidth.”); Paatela, [0077] (“header information corresponding to OSI networking layers two through four may each correspond to a segment of the memory 716”), [0076] (packet 706 “is stored in the memory 716 in time for the editing instructions from the instruction memory 704 to be processed by the editing processor 714, and for the packet data in the memory 716 to be operated on by the editing processor 714.”).

D. Reasons to Combine Cornett, Paatela, and Nelson

56. A POSITA would have combined Cornett, Paatela, and Nelson because all three references are directed to various related approaches to speed up packet processing. For example, Cornett enables faster routing decisions of packets by moving data from one processing buffer to another using a data movement module, such as a DMA engine. Cornett, [0097]. Paatela relatedly speeds up packet processing by moving each one of a packet’s encapsulated headers into individual memory segments and processing each header as needed. Paatela, [0076], Abstract, [0079]. And Nelson accelerates packet routing decisions by eliminating the need to read all of a packet frame (header and payload), focusing instead on the frame header only. Nelson, 1:13-20, 2:3-10.

57. A POSITA would have been motivated to combine the teachings of Cornett, Paatela, and Nelson and doing so would have been within the POSITA’s skillset. The combination would have been obvious because it represents merely the

use of known techniques (**Paatela's** writing packet headers into individual memory segments to accelerate processing and **Nelson's** writing an entire packet frame into one buffer and the packet headers only into another buffer in parallel to accelerate routing decisions) to improve a similar process (**Cornett's** accelerated packet processing techniques where headers are stored separate from an entire packet using a data movement module, such as a DMA engine) in the same way to yield a predictable results.

58. Cornett provides extensive disclosure about how and why accelerated packet processing is beneficial to a network. It also teaches storing packet headers in a buffer separate from an entire packet (header and payload). Cornett, however, does not explicitly disclose how each encapsulated header of a packet is stored in the buffer or how each such header is processed. Paatela discloses this by explaining that each individual encapsulated header, e.g., layer 2, 2.5, 3, or 4, is written to a respective memory segment for processing. Paatela, [0077]; *see* Efficient Data Storage, below. The combination of Cornett and Paatela does not explain the timing of how packet headers are written to the respective buffers of Paatela. Nelson addresses this by explaining that the full packet frame is written into one buffer and the packet headers into another buffer, **in parallel**. This process accelerates packet processing and reduces processing overhead because routing decisions can be made without reading the full frames out of the frame buffer. Nelson, 2:3-10 (“wherein the

frames of data are stored in a first-in-first-out buffer and wherein the header information is accessed to make **routing decisions in order to avoid having to read the frames out of the buffer.**”); *see* Efficient Data Movement, below.

1. Efficient Data Storage (Cornett and Paatela)

59. Cornett explains efficient packet processing using a “packet splitting” technique. Cornett, [0073]. In this process, packet headers are stored in one buffer and entire packets (header and payload, together) are stored in another separate buffer. Cornett, [0081]. One advantage of such header splitting is that a “reduced set of pages can then be managed in a way to permit effective prefetching of packet headers into the processor cache before the protocol stack processes the header.” Cornett, [0064]. Cornett goes on to explain the processing benefits that are gained by using a “data movement module, such as a DMA engine, to move data from one buffer to another buffer.” Cornett, [0097].

60. It would have been appreciated by a POSITA that Cornett’s packets would include a plurality of headers because of header encapsulation. I describe this process in my Background section. Cornett, however, does not explicitly explain how each of the encapsulated headers of the packet are stored or processed.

61. This additional implementation detail is provided by Paatela. Paatela embraces the fact that packets have encapsulated headers, each of which needs to be processed to make effective routing decisions. This is why Paatela teaches that each

individual encapsulated header, e.g., layer 2, 2.5, 3, or 4, is written to a respective segment of memory for processing. Paatela, [0077]. Specifically, Paatela's memory 716 is segmented into "memory locations 718-734" that "correspond to headers of the various protocol layers associated with the incoming packet." Paatela, [0077]. With the protocol headers separated into these memory locations, routing decisions are made for each packet and individual headers are processed and manipulated as needed. For example, Paatela provides an example in the context of switching an IP packet sent in an Ethernet network. If memory location 718 stores an Ethernet header, "the Ethernet header can be modified by writing a new Ethernet header into an otherwise unused, interleaved memory location 720[.]" Paatela, [0079]. When the packet is reassembled for routing, "the new Ethernet header in memory location 720 effectively replaces the original (now-disregarded) Ethernet header at memory location 718. **In this manner, editing of packet layer headers can be effectively and efficiently performed.**" Paatela, [0079]. A POSITA would have thus appreciated that Paatela's individual header processing technique, coupled with Cornett's header splitting technique and the use of a data movement module, would result in an overall faster system.

62. A POSITA would have had a reasonable expectation of success in combining Cornett and Paatela. The combination would account for individual encapsulated headers of Cornett's packets. Cornett already processes headers

separately from an entire packet (header and payload). Cornett, [0081], [0064]. Paatela merely adds details about how each one of a packet's headers is individually stored in separate memory segments and processed. Paatela, [0077]. Cornett's use of a data movement module, such as a DMA engine, would simply offload data movement responsibilities from the core processing module of a host processor. Cornett, [0097]. A POSITA would have understood Paatela to provide the missing implementation details that Cornett expects one to already know. As such, the combination would have been within the skillset of the POSITA.

63. The combination is premised on the teachings of the references. It permits but does not require physical incorporation of elements from Paatela into Cornett. Nevertheless, the physical incorporation of Paatela's teachings for how to process individual encapsulated headers of Cornett's packets would teach the claim limitations. It would amount to using Paatela's input controller and packet classification engine to "determine[] where one networking layer header ends and the next networking layer header begins." Paatela, [0077]. Then, based on this information, using Cornett's data movement module (e.g., DMA engine) to move each individual encapsulated header into a respective memory segment, per Paatela. Cornett, [0074], [0097]. It would have been obvious to incorporate Paatela's teachings into Cornett's technique.

2. Efficient Data Movement (Cornett, Paatela, and Nelson)

64. Cornett moves data efficiently from one buffer to another using a data movement device, such as a DMA engine. Cornett, [0097]. The stated advantage of this methodology is that it “significantly reduce[s] TCP/IP processing overhead that may result from using the core processing module of a host processor.” Cornett, [0097]. Cornett uses the DMA engine to store packet headers in separate buffers from the entire packet (header and payload). Cornett, [0081]. Cornett and Paatela, as described above, would use Cornett’s DMA engine to move each individual encapsulated header of a packet into a respective memory segment, per Paatela. Paatela, [0077]. The combination, however, does not explicitly explain the timing of how packet headers are written to the respective memory buffers.

65. A POSITA would have known that there are only a few options for how to coordinate the storage of packet headers into memory locations. One could move the data sequentially, where each encapsulated packet header would be moved to a memory location one at a time. Another option would be to move each header to a memory location in parallel. That said, a POSITA would have noted that Cornett is focused on reducing processing overhead and reducing processing time. Cornett, [0097]. Similarly, Paatela is optimizing packet processing in an environment where time is of the essence. Paatela, [0076] (packet 706 “is stored in the memory 716 **in time for the editing instructions** from the instruction memory 704 **to be processed**

by the editing processor 714, and for the packet data in the memory 716 to be operated on by the editing processor 714.”). Given this disclosure, a POSITA would have looked to Nelson to provide an implementation that is consistent with Cornett and Paatela. In Nelson, a full packet frame is written into one buffer and the packet headers are written into another buffer in parallel. Nelson, 7:1-4. Nelson writes this data in parallel to accelerate packet processing and reduce processing overhead. Nelson, 2:3-10 (“wherein the frames of data are stored in a first-in-first-out buffer and wherein the header information is accessed to make **routing decisions in order to avoid having to read the frames out of the buffer.**”). Nelson, therefore, presents an obvious and highly logical design choice.

66. A POSITA would have had a reasonable expectation of success in combining Cornett and Paatela further with Nelson. The combination would achieve accelerated packet processing that both Cornett and Paatela strive for. Paatela stores individual encapsulated packet headers in memory segments “in time for the editing instructions from the instruction memory 704 to be processed by the editing processor 714, and for the **packet data in the memory 716 to be operated on by the editing processor 714.**” Paatela, [0076]. A POSITA would have understood that Paatela seeks to make each header available for processing as quickly as possible. Similarly, Cornett seeks to further accelerate packet processing by using a data movement module to free up the host processor and move data more efficiently.

Cornett, [0097]. Nelson furthers the same goal: processing packets faster by writing headers and frames into separate buffers **in parallel**. In this way, header information is made available for routing decisions without the need to read the entire frame from the buffer. Nelson, 2:3-10. A POSITA would have understood that Nelson provides a logical design choice for how to achieve the accelerated packet processing techniques of both Cornett and Paatela. The combination would have been within the skillset of the POSITA.

67. The combination is premised on the teachings of the references. It permits but does not require physical incorporation of elements from Paatela and Nelson into Cornett. Nevertheless, the physical incorporation of Paatela's and Nelson's teachings for how to process individual encapsulated headers of Cornett's packets, and the timing of writing headers and frames into a buffer, would teach the claim limitations. It would amount to using Paatela's input controller and packet classification engine to "determine[] where one networking layer header ends and the next networking layer header begins." Paatela, [0077]. Then, based on this information, using Cornett's data movement module (e.g., DMA engine) to move each individual encapsulated header into a respective memory segment, per Paatela. Cornett, [0074], [0097]. Per Nelson, the write operations would be performed **in parallel** to accelerate packet processing. Nelson, 7:1-4. It would have been obvious to incorporate Paatela's and Nelson's teachings into Cornett's technique.

E. Detailed Analysis of Claims

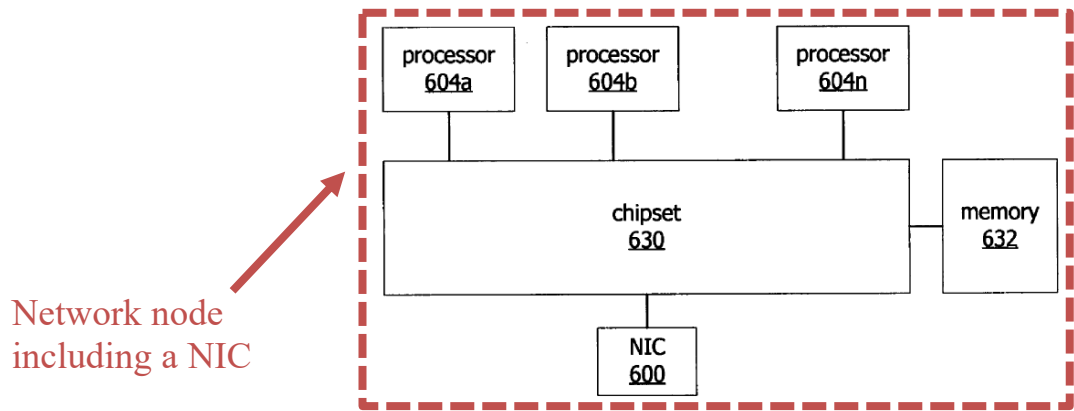
1. Claim 13

a. [13.0] *An egress end user node (EEUN) of a packet based communications system, comprising:*

68. To the extent that the preamble is limiting, Cornett discloses it.

69. Cornett discloses a network comprised of a “**plurality of nodes**” that “may comprise one or more intermediate stations, such as, for example, one or more **hubs, switches, and/or routers**...[and] one or more end stations.” Cornett, [0021], [0023]; *see also* Cornett, [0004] (discussing TCP/IP packet processing using a TCP/IP stack). These nodes may be “communicatively coupled” via a communication medium that enables them to communicate with each other “via wirelined (e.g., copper wires), or wireless (e.g., radio frequency) means.” Cornett, [0021]. Each of the nodes communicate with one another by “transmit[ing] and receiv[ing] sets of one or more signals via medium 104 that may encode one or more packets.” Cornett, [0021]. In this way, Cornett discloses *a packet based communications system*.

70. The nodes include a “network controller” (NIC). Cornett, [0028]; *see also* Cornett, [0077] (“Fig. 9 illustrates a sample computer architecture that can implement the techniques described above [and] includes...[a] NIC 600.”). This is depicted in Figure 9, below:



Cornett, Figure 9 (annotated)

71. Cornett’s “nodes” include a NIC and are therefore the claimed *egress end user node (EEUN)*.

72. This is consistent with the ’471 patent, which describes an EEUN as connected to a communications network as “an intermediate or destination node” such as a “destination computer system, a routing device, a battery powered wireless communications device, or any other device having a TCP/IP based packet interface.” ’471 patent, 4:62-5:2.

b. [13.1] *a decoder configured for decoding a packet having a plurality of headers; and*

73. In one of Cornett’s embodiments, NIC 600 includes “a physical layer device (PHY)” which is the claimed *decoder*. Cornett, [0079]. The PHY is *configured for decoding a packet* because it “**translates between the analog signals of a communications medium (e.g., a cable or wireless radio) and digital bits.**” Cornett, [0079]. A POSITA would have known that, in doing so, the PHY would convert analog-digital representations of a packet as it is transmitted on the

communication medium into digital bits would make that packet readable by digital processing logic within Cornett's "node," which are themselves devices that process digital bits.

74. This is consistent with how the '471 patent explains the function of a decoder: "decoder 116 performs actions involving the decoding of the encoded packet for **placing the same in a form that is readable to a particular computer program installed on the EEUN 110.**" '471 patent, 5:10-14.

75. Cornett is concerned with "accelerated TCP (Transport Control Protocol) stack processing." Cornett, [0003]. A POSITA would have known that packets typically include *a plurality of headers* because a protocol operating at each layer adds its own header to the packet, encapsulating the headers from lower layers. RFC 793, 1 ("TCP is a connection-oriented, end-to-end reliable protocol designed to fit into a layered hierarchy of protocols which support multi-network applications."); Tanenbaum, 19-20 (explaining how a message is generated by an application running in layer 5 "is given to layer 4 for transmission. Layer 4 puts a header in front of the message to identify the message and passes the result to layer 3...layer 3 much break up the incoming messages into smaller units, packets, prepending a layer 3 header to each packet" and so on); Perlman, 4.

c. [13.2] *a direct memory access (DMA) device coupled to said decoder and*

76. In one of Cornett's embodiments, the NIC determines "whether to perform header splitting." Cornett, [0073]. For those packets selected for header splitting, the NIC "can cause storage 704 (e.g., via **Direct Memory Access (DMA)**) of the packet's header[.]" Cornett, [0074]; *see also* Cornett, [0097] ("TCP/IP processing may be accelerated by using a data movement module. Such as a **DMA engine**, to move data from one buffer to another buffer.") Cornett's DMA engine that is used by the NIC to cause storage of packet headers into memory is the claimed *direct memory access (DMA) device*.

77. As described immediately above, Cornett's NIC causes storage of headers into memory using the DMA engine. Cornett, [0074]. And Cornett's NIC includes the PHY, the claimed *decoder*, that converts analog-digital representations of a packet as it is transmitted on the communication medium into digital bits. Cornett, [0079]. On this basis, it would have been obvious to a POSITA that Cornett's DMA engine is *coupled* to the PHY.

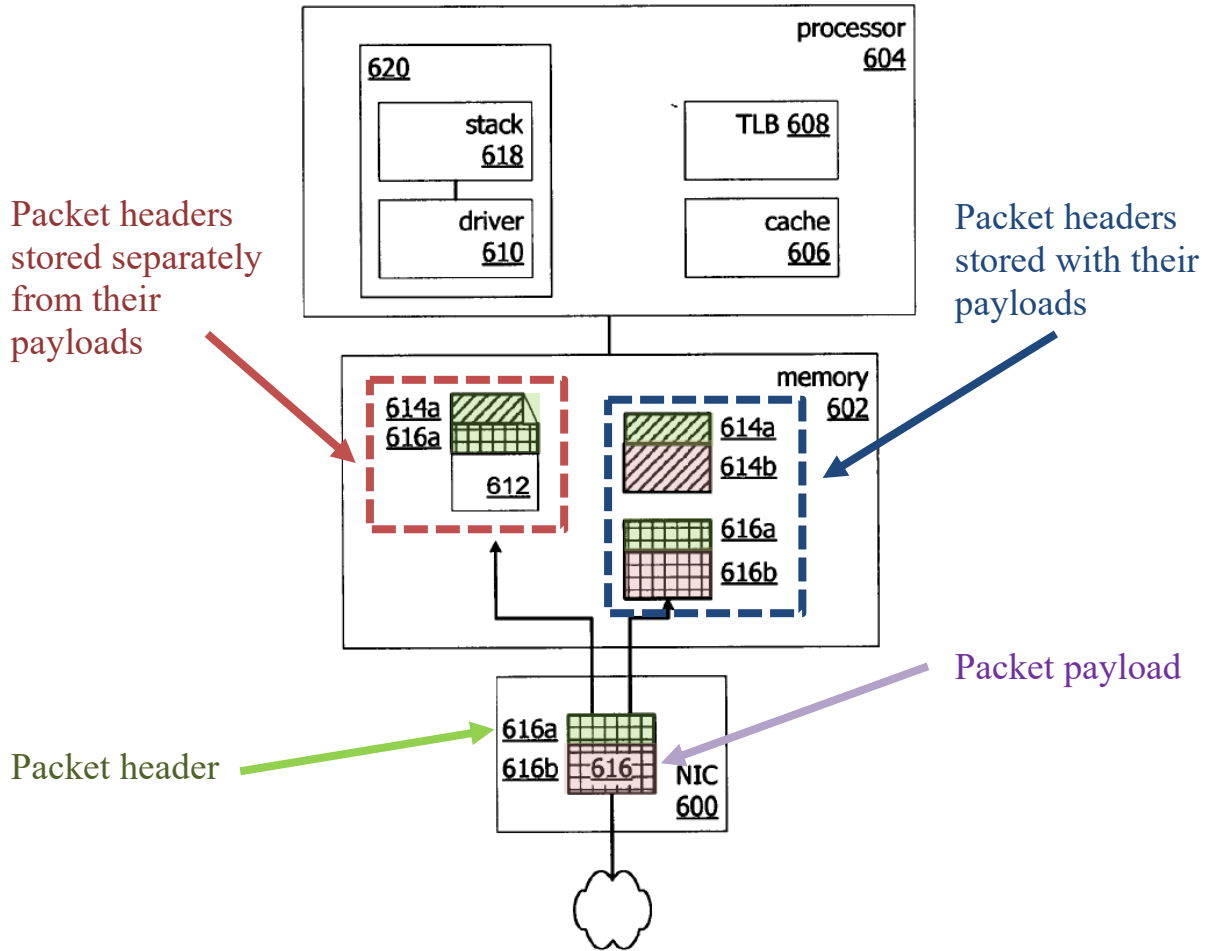
- d. **[13.3] *[a direct memory access (DMA) device coupled to said decoder and] configured for concurrently writing (1) each of said plurality of headers to a packet buffer memory and (2) each individual one of said plurality of headers to a respective protocol stack layer memory where it is available for immediate processing within a protocol stack layer.***

78. Cornett's NIC 600 "receives a packet 614 from a network (shown as a cloud)." Cornett, [0068]. The NIC then determines "whether to perform header splitting." Cornett, [0073]. For those packets selected for header splitting, the NIC "can cause storage 704 (e.g., via **Direct Memory Access (DMA)**) of the packet's header in the page(s) used to store headers and separately store 706 the packet's payload." Cornett, [0074]. Cornett's DMA engine, the claimed *direct memory access (DMA) device*, is therefore *configured for...writing* packet headers.

79. In Figure 10, below, Cornett shows the NIC receiving a packet (labeled 616) that has a header 616a (**green**) and payload 616b (**purple**). Cornett, Figure 10. Cornett further explains that, in Figure 10, the NIC:

may **store the header 614a, 616a and payload 614b, 616b in separate buffers, and additionally store the header 614a, 616a to a location in which the payload 614b, 616b is written.** Put differently, the payload 614b, 616b may be stored to the first location as well as a second location different from the first location.

Cornett, [0081].



Cornett, Figure 10 (annotated)

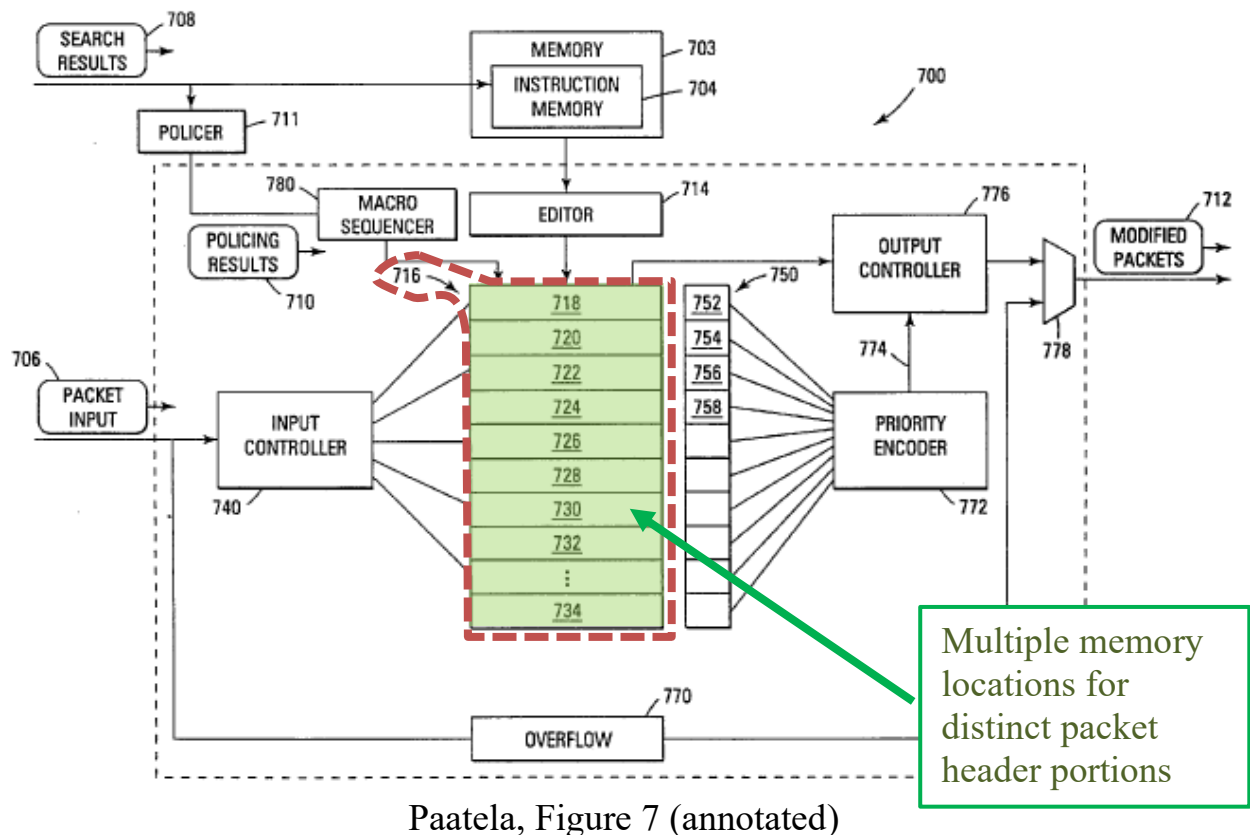
80. A POSITA would have understood Cornett's Figure 10 embodiment as follows: the NIC 600 causes storage of the packet headers 614a and 616a in buffer 612 (red outline) and the entire packet (header and payload together, 614a and b and 616a and b) in another separate buffer (blue outline). See Cornett, [0068] ("controller 600 can write the packet's header 614a into a physical page 612 of memory 602 used for storage of packet headers, while the packet payload 614b is written into a different location[.]") The buffer where the entire packet, with all its

headers and payload, is stored (**blue** outline) is the claimed *packet buffer memory* because a *plurality of headers* is stored there.

81. Cornett's Figure 7 embodiment describes the DMA engine writing headers and payloads to separate locations. A POSITA would have understood that the DMA engine of the Figure 7 embodiment would operate in the same manner in the embodiments depicted and described in connection with Figures 9 and 10. A POSITA would have appreciated that the embodiments include NIC 600. Cornett also states that Figure 9 "illustrates a sample computer architecture that can implement the techniques described above," which would include the Figure 7 embodiment. Cornett, [0077]. Moreover, Cornett explains that the embodiments of its invention may significantly reduce TCP/IP processing overhead by using "a data movement module, such as a DMA engine, to move data from one buffer to another buffer." Cornett, [0097]. This further disclosure would confirm to a POSITA that the DMA engine shown and described in Cornett's Figure 7 embodiment is expected to be a part of Cornett's embodiments in Figures 9 and 10.

82. As described above in my analysis of limitation [13.2], Cornett's NIC causes storage of headers into memory using the DMA engine. Cornett, [0074]. Taken together with the additional disclosure discussed immediately above, a POSITA would have understood that Cornett's DMA engine is *configured for [] writing* packet headers into a *packet buffer memory*.

83. As discussed above, Cornett teaches storing packet headers in a buffer. Cornett, however, does not explicitly disclose how each encapsulated header of the packet is stored or processed. This is disclosed by Paatela. Paatela discloses a process for “packet transformation” where packets are “temporarily stored, and acted upon through processing of protocol-dependent instructions.” Paatela, Abstract. An embodiment of this process is implemented on a system 700 depicted and described in connection with Figure 7.



84. The system receives inputs that “include packet/frame input,” labeled packet input 706. Paatela, [0060]. Memory 716 (red outline) that “stores portions of

the packet input 706, and stores these packet portions in memory locations 718-734 as dictated by the input controller 740.” Paatela, [0076]. Memory locations 718-734 are highlighted in **green**. “[T]hese packet portions correspond to headers of the various protocol layers associated with the incoming packet.” Paatela, [0077]. The input controller 740 directs the header information to memory 716 based on “upstream information developed by a packet classification engine that determines where one networking layer header ends and the next networking layer header begins.” Paatela, [0077]. Memory 716 is “configured and partitioned such that all header information and data is stored within the memory 716.” Paatela, [0085].

Paatela explains:

For example, header information corresponding to OSI networking layers two through four may each correspond to a segment of the memory 716, such that a segment of memory 716 is allocated to store a layer-2 header (e.g., a PPP header), a segment corresponding to a layer-2.5 header (e.g., an MPLS label stack), a segment corresponding to a layer-3 header (e.g., an IP header), and a segment corresponding to a layer-4 header (e.g., a TCP header).

Paatela, [0077].

85. In this way, Paatela discloses that all header information, which includes *each individual one of said plurality of headers*, is written to a segment of

memory that is allocated to store a particular OSI networking layer header. These designated memory segments correspond to a *respective protocol stack layer memory* because each segment stores an individual layer packet header.

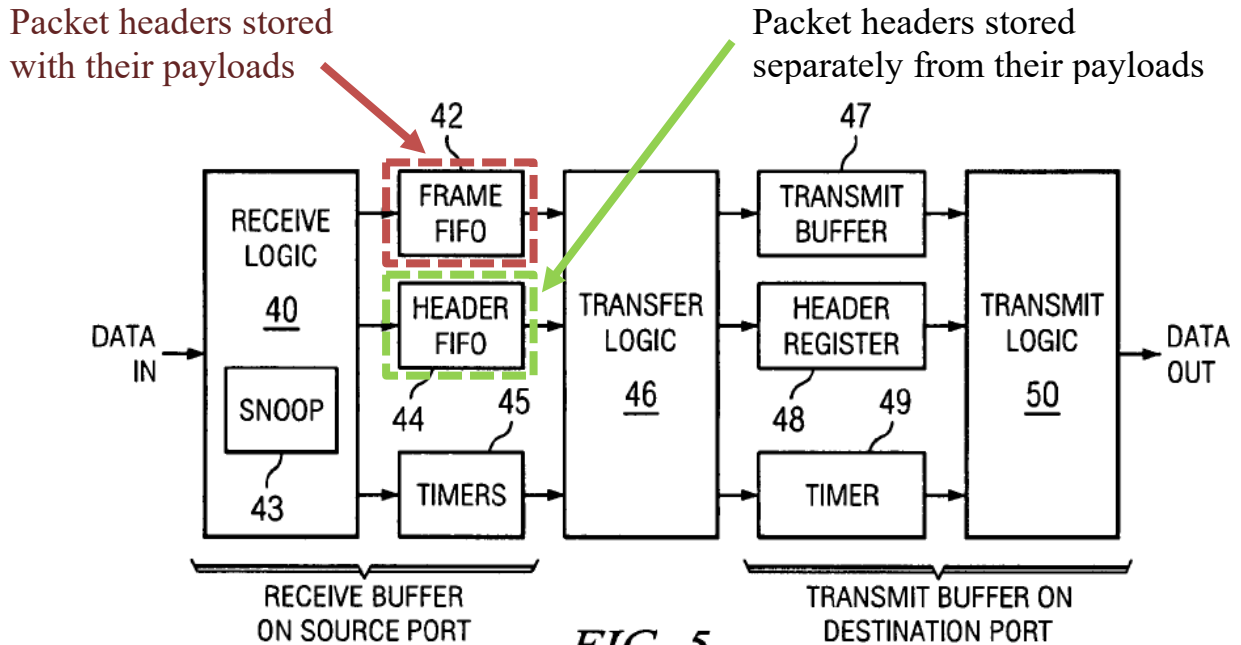
86. As explained above in my Section IX.D analysis, a POSITA would have been motivated to combine Cornett and Paatela because Paatela provides implementation details that are missing from Cornett. As explained in my analysis of limitation [13.1], a POSITA would have understood from Cornett's disclosure that its packets include a plurality of headers. Paatela explains how encapsulated packet headers are processed, i.e., each individual encapsulated header, e.g., layer 2, 2.5, 3, 4, are written to a respective segment of memory for processing. In the combination, Cornett's NIC would cause storage of each individual encapsulated header into a respective segment of memory for processing, per Paatela. Paatela, [0077]. As Cornett's NIC causes storage of headers using the DMA engine, Cornett, [0074]; *see also* limitation [13.2], it would have been obvious to a POSITA that Cornett's DMA engine is *configured for [] writing...each individual one of said plurality of headers to a respective protocol stack layer memory*.

87. Both Cornett and Paatela discuss the OSI model, Cornett, [0038]; Paatela, [0007], Cornett processes TCP/IP packets, Cornett, [0004], and Paatela processes packet headers of "various protocol layers associated with the incoming packet." Paatela, [0077]. Both Cornett and Paatela process Ethernet packets. Cornett,

[0079]; Paatela, [0079]. A POSITA would have therefore understood that both Cornett and Paatela can operate on TCP/IP packets.

88. As explained above, the combination of Cornett and Paatela render obvious a *decoder* that is *configured for writing* (1) *each of said plurality of headers to a packet buffer memory* and (2) *each individual one of said plurality of headers to a respective protocol stack layer memory*. The combination, however, does not explicitly explain the timing of how packet headers are written to the respective buffers. This timing is explained in Nelson.

89. Nelson discloses processing packets and making routing decisions based on packet information. Nelson, Abstract. As depicted in Figure 5, below, packet frames are first received by receive logic 40. Nelson, 6:62-63. Receive logic 40 “includes circuitry to snoop on the received data and to copy the header of each frame to header FIFO 44. **The copying of a header to header FIFO 44 occurs in parallel with the storage of the corresponding frame in frame FIFO 42.**” Nelson, 6:66-7:4. In this way, Nelson discloses *writing the packet header* included in a frame (1) to a frame FIFO 42 (together with the entire frame) (**red** outline) and (2) to the header FIFO 44 (**green** outline). This writing process is performed in parallel, i.e., *concurrently*. It should be noted that while Nelson describes its process as using a FIFO, it also explains that “[d]ual port RAM can be used to store the header information” instead of a FIFO. Nelson, 6:8-9.



Nelson, Figure 5 (annotated)

90. As explained above in my Section IX.D analysis, a POSITA would have been motivated to combine Cornett and Paatela further with Nelson. Cornett is focused on reducing processing overhead and reducing processing time. Cornett, [0097]. Paatela similarly seeks to optimize packet processing in a solution where time is of the essence. Paatela, [0076] (packet 706 “is stored in the memory 716 **in time for** the editing instructions from the instruction memory 704 to be processed by the editing processor 714, and for the packet data in the memory 716 to be operated on by the editing processor 714.”) Nelson, in turn, explains that writing a full packet frame into one buffer and the packet headers into another buffer, **in parallel**, accelerates packet processing and reduces processing overhead because routing decisions can be made without reading the full frames out of the frame

buffer. Nelson, 2:3-10 (“wherein the frames of data are stored in a first-in-first-out buffer and wherein the header information is accessed to make **routing decisions in order to avoid having to read the frames out of the buffer.**”). For example, Nelson’s technique allows the packet headers to be processed (e.g., for making a routing decision) even as the packet payload is being received and stored in the frame buffer. *See, e.g.*, Nelson 7:30-33 (“While the frame is being moved from the receive FIFO 42 to the transmit buffer 47, transfer logic 46 examines the next header in FIFO 44 and makes a routing decision for the corresponding frame in receive FIFO 42.”). Nelson’s use of “dual port RAM” to store packet headers would facilitate immediate processing because data written into a dual port RAM can be read out at the same time. Nelson, 6:8-9. A POSITA would thus have been motivated to combine Cornett and Paatela with Nelson because Nelson’s parallel writing of headers and frames would further accelerate packet/header processing, which is a stated goal of both Cornett and Paatela.

91. In the combination, Cornett’s NIC would cause its DMA engine to store the entire packet, with all its headers and payload, in a first buffer, the claimed *packet buffer memory*. Cornett, [0081]. Cornett’s NIC would further cause its DMA engine to store each individual encapsulated header into a respective segment of memory for processing, per Paatela, which are the claimed *respective protocol stack layer*

memory. Paatela, [0077]. The NIC would cause the DMA engine to perform both storage actions in parallel, i.e., *concurrently*, per Nelson. Nelson, 6:66-7:4.

92. Paatela further explains that packet 706 “is stored in the memory 716 **in time for the editing instructions** from the instruction memory 704 to be processed by the editing processor 714, and for the **packet data in the memory 716 to be operated on by the editing processor 714.**” Paatela, [0076]. In other words, a POSITA would have understood that time is of the essence in Paatela’s processing. Based on the urgency of timely packet header processing in Paatela, it would have been obvious to a POSITA for the headers to be written such that they are *available for immediate processing*. That obviousness is also consistent with Nelson, which describes making header information available for routing decisions without reading the frame out of a frame buffer. Nelson, 2:3-10. It would have been obvious from Nelson, for example, for the headers to be *available for immediate processing* even as the payload portion of a frame (which is generally received after the headers) continues to be written into the frame buffer. Nelson, 6:8-9.

93. Paatela’s editing processor 714 operates on individual packet headers “to perform packet modifications and provide packet steering information...to perform certain networking-specific tasks that depend on the particular networking protocol.” Paatela, [0063]. For example, “if memory location 718 stores an Ethernet header,” it can be modified by editing processor 714 and a new Ethernet header can

be stored in a new memory location 720. Paatela, [0079]. As each header would be processed by the protocol operating within a particular layer, it would have been obvious to a POSITA that a packet's header would be processed *within a protocol stack layer*. See Paatela, [0063] ("The editor instructions are executed to perform packet modifications and provide packet steering information....Such special purpose instructions may be particularly useful to perform certain networking-specific tasks that depend on the particular networking protocol."). In other words, Paatela's description of replacing an incoming Ethernet header with a new outgoing Ethernet header renders obvious making each of the packet headers available for processing *within a protocol stack layer*.

2. Claim 14

- a. **[14.0] *The EEUN according to claim 13, wherein at least one of said plurality of headers is a media access control layer protocol header and said respective protocol stack layer memory is a media access control layer memory.***

94. As I explained in my analysis of limitation [13.3], a POSITA would have understood that both Cornett and Paatela operate on TCP/IP packets. And both Cornett and Paatela process Ethernet packets. Cornett, [0079]; Paatela, [0079], [0113]. A POSITA would have known that an Ethernet header is an example of a *media access control layer protocol header*. See, e.g., Temoshenko, 4:2-4 ("Ethernet MAC headers"); Kaniz, [0008] ("an Ethernet (MAC) header"); Elliott, [0043]

(discussing “Ethernet frames.... Each such frame starts with Media Access Control (MAC) information....”). Thus, it would have been obvious to a POSITA for a packet sent or received by Cornett and Paatela to include a *media access control layer protocol header*.

95. As explained in my analysis of limitation [13.3], it would have been obvious to a POSITA to store a specific protocol layer header in a corresponding layer memory. For example, Paatela specifically discloses that the OSI layer 2 header of a packet frame would be stored in “a segment of memory 716 [that] is allocated to store a layer-2 header (e.g., a PPP header)[.]” Paatela, [0077].¹ The portion of Paatela’s memory segment 716 that stores this information is the claimed *media access control layer memory*. This is because an OSI layer 2 header includes the MAC header. Perlman, 3; Olderdissen, [0048]. Paatela also describes storing an “Ethernet header” in a specific “memory location 718” separate from other protocol headers. Paatela, [0079]; *see also* Paatela [0113], [0120]. The separate memory location 718 alternatively corresponds to the claimed *media access control layer memory*. It would have been obvious to a POSITA to store a specific protocol layer header in the corresponding memory layer.

¹ A POSITA would have known that the PPP header is contained in OSI layer 2.

3. Claim 15

- a. [15.0] *The EEUN according to claim 13, wherein at least one of said plurality of headers is a logic link control layer protocol header and said respective protocol stack layer memory is a logic link control layer memory.***

96. As I explained in my analysis of limitation [13.3], a POSITA would have understood that both Cornett and Paatela operate on TCP/IP packets. And both Cornett and Paatela process Ethernet packets. Cornett, [0079]; Paatela, [0079], [0113]. A POSITA would have been familiar with Ethernet frame headers and would have known that an Ethernet frame header may optionally include a *logic link control layer protocol header*. The term “Ethernet” was “commonly used by those of ordinary skill in the art” to refer to both the canonical “Ethernet” networking standard (sometimes also called DIX Ethernet) and to the IEEE 802.3 standard. Haddock, 8:42-51; Kashyap, 2:4-7. The IEEE “802.3 Ethernet” standard “require[s] use of a logical link control (LLC) header.” Kashyap, 2:14-18; *see also* Haddock, 8:61-64 (“the 802.3 MAC frame has an IEEE 802.2 logical link control (LLC) header following the MAC header and preceding the OSI network layer protocol header, if one exists.”); Fudatate, Figs. 5A & 5B, 4:20-28. Thus, because the term “Ethernet” was understood to include the IEEE 802.3 standard, which mandates the use of a logical link control (LLC) header, it would have been obvious to a POSITA for the

“Ethernet” frames described in Cornett and Paatela to include a *logic link control layer protocol header*.

97. As explained in my analysis of limitations [13.3] and [14.0], it would have been obvious to a POSITA to store a specific protocol layer header in a corresponding layer memory. For example, Paatela specifically discloses that the OSI layer 2 header of a packet frame would be stored in “a segment of memory 716 [that] is allocated to store a layer-2 header (e.g., a PPP header)[.]” Paatela, [0077]. In IEEE 802.3 Ethernet, an LLC header is included in the layer 2 header. Perlman, 3; Swenson, 3:24-26. That portion of memory segment 716 that stores this information is the claimed *logic link control layer memory*. Paatela also describes storing an “Ethernet header” in a specific “memory location 718” separate from other protocol headers. Paatela, [0079]; *see also* Paatela [0113], [0120] (giving further examples of storing an “Ethernet header” in a separate “memory location”). The separate memory location 718 alternatively corresponds to the claimed *logic link control layer memory*. It would have been obvious to a POSITA to store a specific protocol layer header in the corresponding memory layer.

4. Claim 17

- a. [17.0] *The EEUN according to claim 13, wherein at least one of said plurality of headers is a network layer protocol header and said respective protocol stack layer memory is a network layer memory.***

98. As explained in my Background section, the TCP/IP model includes a network layer (layer 2). Olderdisen, [0044]. A POSITA would have known that the “TCP/IP network layer corresponds to the network layer [layer 3] of the OSI model.” Olderdisen, [0012]. In other words, a POSITA would have understood that the TCP/IP network layer includes the same information as the network layer in OSI.

99. A POSITA would also have known that the network layer in the OSI model includes a layer 3 header (e.g., an IP header). *See* Paatela, [0077] (“header information corresponding to OSI networking layers two through four may each correspond to a segment of the memory 716, such that a segment of memory 716 is allocated to store...a segment corresponding to a layer-3 header (e.g., an IP header)[.]”). In the TCP/IP model, the IP header is found in the network layer. A POSITA would have known, or it would have at least been obvious, that the well-known IP header is a *network layer protocol header*.

100. As I explained in my analysis of limitation [13.3], a POSITA would have understood that both Cornett and Paatela operate on TCP/IP packets. Paatela specifically discloses that the OSI layer 3 header of a packet frame would be stored in “a segment of memory 716 [that] is allocated to store a...layer-3 header (e.g., an

IP header)[.]” Paatela, [0077]. Paatela also provides further examples in which an IP packet header is written to an IP header memory. *See* Paatela, [0113], [0120], [0125]. As the IP header is an example of a layer 3 header, that portion of memory segment 716 that stores this information is the claimed *network layer memory*.

5. Claim 18

- a. [18.0] *The EEUN according to claim 13, wherein at least one of said plurality of headers is a transport layer protocol header and said respective protocol stack layer memory is a transport layer memory.***

101. As explained in my Background section, the TCP/IP model includes a transport layer (layer 3). Olderdisen, [0044]. A POSITA would have known that the “TCP/IP transport layer corresponds to the transport layer [layer 4] of the OSI model.” Olderdisen, [0011]. In other words, a POSITA would have understood that the transport layer includes the same information in the OSI model and the TCP/IP model.

102. Paatela teaches that the transport layer in the OSI model includes a layer 4 header (e.g., a TCP header). *See* Paatela, [0077] (“header information corresponding to OSI networking layers two through four may each correspond to a segment of the memory 716, such that a segment of memory 716 is allocated to store...a segment corresponding to a layer-4 header (e.g., a TCP header)[.]”). A POSITA would have known, or it would have at least been obvious, that the TCP header is found in the transport layer. The well-known TCP header is a *transport*

layer protocol header. See '471 patent, 7:60-62 (“transport protocols include a transmission control protocol (TCP)”).

103. As I explained in my analysis of limitation [13.3], a POSITA would have understood that both Cornett and Paatela operate on TCP/IP packets. Paatela specifically discloses that the OSI layer 4 header of a packet frame would be stored in “a segment of memory 716 [that] is allocated to store a...layer-4 header (e.g., a TCP header)[.]” Paatela, [0077]. Paatela also provides further examples in which a TCP packet header is written to a TCP header memory. Paatela, [0125], [0130]. Another well-known transport protocol was User Datagram Protocol (UDP) and Paatela similarly describes storing a UDP header in a separate UDP header memory. Paatela, [0113], [0120]. As the TCP or UDP headers are examples of a layer 4 header, that portion of memory segment 716 that stores this information is the claimed *transport layer memory*.

6. Claim 19

- a. [19.0] *The EEUN according to claim 13, wherein at least one of said plurality of headers is an application layer header and said respective protocol stack layer memory is an application layer memory.***

104. As explained in my Background section, the TCP/IP model includes an application layer (layer 4). Olderdissen, [0010]. A POSITA would have known that the “TCP/IP application layer corresponds to the combination of the application [layer 7], presentation [layer 6] and session [layer 5] layers of the OSI model.”

Olderdissen, [0011]. In other words, a POSITA would have understood that the TCP/IP application layer includes the same information as the combination of the application, presentation, and session layers in the OSI model.

105. It would also have been obvious to a POSITA for the application layer to include an application header. *See* Boucher, 2:43-48 (“Thus an application layer attaches an application header to the payload data[.]”). The well-known application header is an *application layer header*.

106. As I explained in my analysis of limitation [13.3], a POSITA would have understood that both Cornett and Paatela operate on TCP/IP packets. Paatela explains that packets include portions that correspond to “various protocol layers associated with the incoming packet.” Paatela, [0077]. Paatela then explains that header information corresponding to OSI networking layers “may each correspond to a segment of the memory 716.” Paatela, [0077]. While Paatela explicitly addresses storing layer 2 through 4 headers in memory 716, Paatela does not place any limits on which headers it can process. It therefore would have been obvious to a POSITA that Paatela would store and process the header of the application layer, when present. That portion of memory segment 716 that stores the application header corresponds to the claimed *application layer memory*.

7. Claim 20

- a. [20.0] *The EEUN according to claim 13, wherein said decoder is further configured for communicating a portion of said packet to said DMA device subsequent to decoding said packet.***

107. As described above in my analysis of limitation [13.2], Cornett's NIC 600 receives packets from a network and determines whether to perform header splitting. Cornett, [0068], [0073]. For those packets selected for header splitting, the NIC "can cause storage 704 (e.g., via **Direct Memory Access (DMA)**) of the packet's header in the page(s) used to store headers and separately store 706 the packet's payload." Cornett, [0074].

108. As described above in my analysis of limitation [13.1], Cornett's NIC 600 includes "a physical layer device (PHY)," which is the claimed *decoder*. Cornett, [0079]. The PHY may also "be communicatively coupled to a media access controller (MAC) (e.g., via a FIFO) that performs 'layer 2' operations (e.g., Ethernet frame handling)." Cornett, [0079].

109. A POSITA would have first understood, based on this disclosure, that Cornett's NIC receives signals representing packets from the communications medium. The PHY then "translates" the signals representing packets "between the analog signals of a communications medium (e.g., a cable or wireless radio) and digital bits." Cornett, [0079]. A POSITA would have found it obvious for the PHY to perform this translation step before the packets and their headers are transmitted

for storage by the NIC via the DMA engine. Cornett, [0074]. This understanding is supported by Cornett's description of memory 602 as "a collection of physical pages of contiguous memory addresses," Cornett, [0065], and a POSITA's knowledge that physical memory is a form of digital memory. In the digital computing context of Cornett, it is generally not feasible to directly store an analog signal received from a communication medium; instead, it would have been obvious to decode the information contained in a received signal into digital data, then store that digital data. In this way, Cornett renders obvious that the PHY, the claimed *decoder*, is *further configured to communicat[e] a portion of said packet to said DMA device subsequent to decoding said packet.*

8. Claim 21

- a. [21.0] *The EEUN according to claim 20, wherein said portion is absent of a preamble and a physical layer header.***

110. A POSITA would have known that IEEE 802.3 defines the structure of Ethernet frames, i.e., packets. A POSITA would have known the following:

A node starts transmission by sending an 8 byte (64 bit) preamble sequence. This consists of 62 alternating 1's and 0's followed by the pattern 11....The purpose of the preamble is to allow time for the receiver in each node to achieve lock of the receiver Digital Phase Lock Loop which is used to synchronize the receive data clock to the transmit data clock. At the point when the first bit of the

preamble is received, each receiver may be in an arbitrary state (i.e. have an arbitrary phase for its local clock). During the course of the preamble it learns the correct phase, but in so doing it may miss (or gain) a number of bits. **A special pattern (11), known as the start of frame delimiter, is therefore used to mark the last two bits of the preamble.**

Batty 1:30-44.

111. It would have been obvious to a POSITA that the first 62 bits of the preamble sequence are the claimed *preamble* and the last two bits of the preamble sequence, also known as the start of frame delimiter, is the claimed *physical layer header*. A POSITA would have appreciated that the start of frame delimiter is followed by packet addressing information, Tanenbaum, 280-281, all of which is added to the packet by each TCP/IP and OSI layer, as described above in my encapsulation discussion of my Background section. This obviousness is confirmed by the fact that the physical layer is the lowest layer in both TCP/IP and OSI models. This means that information that follows the physical layer header is data communicated at the next higher layer.

112. A POSITA would also have known that “the Ethernet receive interface starts collecting the bits into bytes for processing by the MAC layer” only after the last two bits of the preamble—start of frame delimiter—is received. Betty, 1:44-46. In Cornett, the NIC 600 includes “a physical layer device (PHY),” which is the

claimed *decoder*. Cornett, [0079]. The PHY may also “be communicatively coupled to a media access controller (MAC) (e.g., via a FIFO) that performs ‘layer 2’ operations (e.g., Ethernet frame handling).” Cornett, [0079]. As described above in my analysis of limitation [20.0], Cornett’s NIC receives signals representing packets from the communications medium. The PHY then “translates” the signals representing packets “between the analog signals of a communications medium (e.g., a cable or wireless radio) and digital bits.” Cornett, [0079]. A POSITA would have understood that, in so doing, Cornett’s PHY translates the entire packet and then passes for processing the information that **follows** the preamble and start of frame delimiter. *See, e.g., Betty*, 1:44-46. Therefore, as described above in my analysis of limitation [20.0], it would have been obvious to a POSITA that Cornett’s PHY, the claimed *decoder*, transmits for storage the packets and their headers by the NIC via the DMA engine. Cornett, [0074]. These packets would be *absent of the preamble and physical layer header*.

9. Claim 1

a. [1.0] *A method for processing a packet at an egress end user node, comprising:*

113. To the extent that the preamble is limiting, Cornett discloses it.

114. As described above in my analysis of limitation [13.0], Cornett’s “nodes” includes a NIC and are the claimed *egress end user node (EEUN)*. Cornett further discloses that the NIC in Cornett’s nodes includes the functionality to

perform a process called “header splitting.” Cornett, [0073]. Cornett’s header splitting process is the claimed *method for processing a packet*.

- b. [1.1] *decoding a packet having a plurality of headers;
and*

115. This limitation is substantively identical to limitation [13.1] and my analysis there applies equally to this limitation.

- c. [1.2] *subsequent to said decoding step, concurrently writing (1) each of said plurality of headers to a packet buffer memory and (2) each individual one of said plurality of headers to a respective protocol stack layer memory where it is available for immediate processing within a protocol stack layer.*

116. This limitation includes features from limitations [13.3] and [20.0] with some minor differences. Unlike limitation [13.3], this limitation does not recite a DMA device. Instead, this limitation recites a more generic “concurrently writing” step. Nevertheless, my analysis for limitation [13.3] applies equally to this limitation because this limitation is broader than limitation [13.3]. Unlike limitation [20.0], this limitation merely recites “concurrently writing” information “subsequent to said decoding step.” Limitation [20.0] is again limited to the use of a DMA device. My analysis for limitation [20.0] applies equally here at least because this limitation is broader in that it is not limited to the use of a DMA device.

10. Claim 2

- a. **[2.0] *The method according to claim 1, wherein at least one of said plurality of headers is a media access control layer protocol header and said respective protocol stack layer memory is a media access control layer memory.***

117. This claim is substantively identical to claim 14 and my analysis there applies equally to this claim.

11. Claim 3

- a. **[3.0] *The method according to claim 1, wherein at least one of said plurality of headers is a logic link control layer protocol header and said respective protocol stack layer memory is a logic link control layer memory.***

118. This claim is substantively identical to claim 15 and my analysis there applies equally to this claim.

12. Claim 5

- a. **[5.0] *The method according to claim 1, wherein at least one of said plurality of headers is a network layer protocol header and said respective protocol stack layer memory is a network layer memory.***

119. This claim is substantively identical to claim 17 and my analysis there applies equally to this claim.

13. Claim 6

- a. ***[6.0] The method according to claim 1, wherein at least one of said plurality of headers is a transport layer protocol header and said respective protocol stack layer memory is a transport layer memory.***

120. This claim is substantively identical to claim 18 and my analysis there applies equally to this claim.

14. Claim 7

- a. ***[7.0] The method according to claim 1, wherein at least one of said plurality of headers is an application layer header and said respective protocol stack layer memory is an application layer memory.***

121. This claim is substantively identical to claim 19 and my analysis there applies equally to this claim.

15. Claim 8

- a. ***[8.0] The method according to claim 1, further comprising concurrently processing said plurality of headers in each of a plurality of protocol stack layers.***

122. As described above in my analysis of limitation [13.3], a POSITA would have understood that time is of the essence in Paatela's processing. *See* Paatela, [0076] (packet 706 "is stored in the memory 716 **in time for the editing instructions** from the instruction memory 704 to be processed by the editing processor 714, and for the **packet data in the memory 716 to be operated on by the editing processor 714.**"). Based on the urgency of timely packet header

processing in Paatela, it would have been obvious to a POSITA for the headers to be written such that they are available for immediate processing.

123. Paatela operates to modify headers stored in memory 716 as needed. Once all the modifications have been made, and modified headers have been stored in alternative memory segments, the packet is reassembled and Paatela's "memory locations are read out in the proper order" such that, for example, "the new Ethernet header is memory location 720 effectively replaces the original (now-disregarded) Ethernet header at memory location 718." Paatela, [0079]. As time is of the essence, it would have been obvious to a POSITA that *processing said plurality of headers* would occur *concurrently*. Concurrent processing would make sense because the timing of packet reassembly, per Paatela, would not be impacted by delayed processing of a header that might be processed last.

124. As described above in my analysis of limitation [13.3], packet data in respective memory locations 716 would be processed *within a protocol stack layer*. In other words, Paatela's description of replacing an incoming Ethernet header with a new outgoing Ethernet header renders obvious making each of the packet headers available for processing *within a protocol stack layer*.

16. Claim 9

- a. **[9.0] *The method according to claim 1, further comprising communicating a portion of said packet to a direct memory access (DMA) device subsequent to said decoding step and prior to said concurrently writing step.***

125. This claim is substantively identical to claim 20 and my analysis there applies equally to this claim.

17. Claim 10

- a. **[10.0] *The method according to claim 9, wherein said portion is absent of a preamble and a physical layer header.***

126. This claim is substantively identical to claim 21 and my analysis there applies equally to this claim.

18. Claim 12

- a. **[12.0] *The method according to claim 1, wherein said concurrently writing step is performed within a physical layer utilizing a direct memory access (DMA) device.***

127. As described above in my analysis of limitation [13.1], Cornett's NIC 600 includes "a physical layer device (PHY)," which is the claimed *decoder*. Cornett, [0079]. The PHY may also "be communicatively coupled to a media access controller (MAC) (e.g., via a FIFO) that performs 'layer 2' operations (e.g., Ethernet frame handling)." Cornett, [0079]. As described above in my analysis of limitation [13.2], Cornett's NIC causes storage of headers into memory *utiliz[es] a direct memory access (DMA) device*. Cornett, [0074]. As described in my analysis of

limitation [13.3], it would have been obvious to a POSITA that Cornett's PHY uses the DMA engine to perform the *concurrently writing step*.

128. A POSITA would have understood that Cornett's PHY is operating *within a physical layer*. This is because the PHY receives the packet from the communications medium (the physical layer), translates it, and then causes storage of its headers into memory. Cornett, [0079]. It would also have been obvious to a POSITA that Cornett's PHY operates as an interface between the physical layer and the other, higher layers, at least because it is "coupled to a media access controller (MAC)...that performs 'layer 2' operations (e.g., Ethernet frame handling)." Cornett, [0079].

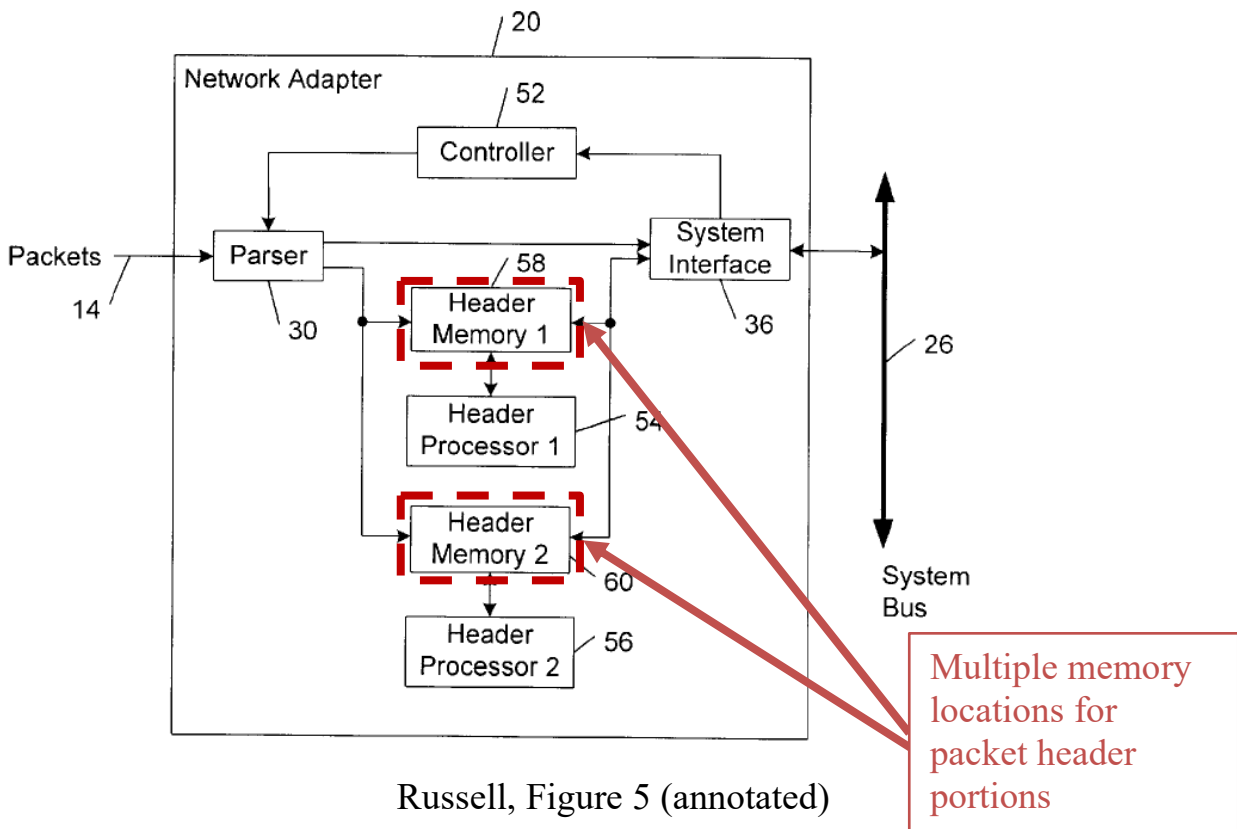
X. THE COMBINATION OF CORNETT, PAATELA, NELSON, AND RUSSELL RENDERS OBVIOUS CLAIM 8

A. Summary of Russell

129. Russell relates to "improved packet processing results" that are achieved by "logically and physically separating packet header processing functions from packet data processing functions." Russell, 1:56-58. This is done to address the known problem of extended "latency between the time a packet is received and the time the packet is processed." Russell, 1:33-35. Russell's proposed process thus enables "a network processing system to perform network handling operations and data processing operations substantially in parallel." Russell, 1:59-61. Russell's

processing system can be implemented on “a personal computer, a workstation, a server computer, a router, a peer device or other network node.” Russell, 3:24-26.

130. Russell’s Figure 5 embodiment describes the operation of a network adapter 20 that is included in its processing system 10. Russell, Figures 5 and 3. This network adapter includes a packet parser 30, a controller 52, a first and second header processors 54 and 56, a first and second header memories 58 and 60, and a system interface 36. Russell, 5:3-7. This configuration is depicted below:



131. In this configuration, “packet parser 30 parses a received information packet 14, and directs a portion of the packet header to first header memory 58 and directs the remaining portion of the packet header to second header memory 60.”

Russell, 5:7-10. Russell's "[f]irst and second header processors 54, 56 are configured to process packet header information stored in header memories 58, 60 in accordance with one or more network protocol handling operations (e.g., firewall filtering, load balancing, and TCP/IP or RTP protocol handling operations)." Russell, 5:22-27.

132. Russell and the '471 patent are both directed to packet-based network communications with the same specific focus of accelerating packet header processing. *See* '471 patent, 1:8-11 ("invention relates to a method and apparatus for high speed protocol header processing at an intermediate and/or destination node of a packet-based communications network."); Russell, Abstract, 1:33-35 (invention "provide[s] improved packet processing results" and "reduce[s] latency between the time a packet is received and the time the packet is processed."). In addition, Russell and the '471 patent address the same problem of how to improve packet processing results by processing packet headers individually. *See* '471 patent, 3:32-34 ("concurrently processing the headers in each of a plurality of protocol stack layers."); Russell, claim 8 at 8:7-9 ("two or more adapter processors each operable to process a respective one or more stored header components substantially in parallel.").

B. Reasons to Combine Cornett, Paatela, Nelson, and Russell

133. As explained above in Section IX.D, Cornett, Paatela, and Nelson are directed to various related approaches to speed up packet processing. Cornett, [0097;

Paatela, [0076], Abstract, [0079]; Nelson, 1:13-20, 2:3-10. Russell also improves packet processing by reducing latency. Russell, Abstract, 1:33-35, 3:13-14.

134. A POSITA would have been motivated to combine the teachings of Cornett, Paatela, and Nelson further with Russell and doing so would have been within the POSITA's skillset. The combination would have been obvious because it represents merely the use of known techniques (**Paatela's** writing packet headers into individual memory segments to accelerate processing, **Nelson's** writing an entire packet frame into one buffer and the packet headers only into another buffer in parallel to accelerate routing decisions, and **Russell's** processing multiple packet headers in parallel) to improve a similar process (**Cornett's** accelerated packet processing techniques where headers are stored separate from an entire packet using a data movement module, such as a DMA engine) in the same way to yield a predictable results.

135. As I explain above in my Section IX.D, the combination of Cornett, Paatela, and Nelson processes packets and uses a DMA engine to save each individual encapsulated header of a packet into a respective memory segment, per Paatela, in parallel, i.e., concurrently, per Nelson. The combination, however, does not explicitly explain the timing of when each of these headers is processed. This missing detail is provided by Russell that explains that one way to reduce latency in processing packets is to process separate packet headers in parallel. Russell, claim 8

at 8:8-9; 1:33-35, 3:13-14. A POSITA would have been motivated to combine Cornett, Paatela, and Nelson further with Russell.

136. As I explain in my analysis of limitation [13.3], a POSITA would have appreciated that time is of the essence in Paatela. Paatela, [0076]. Russell also seeks to reduce latency “between the time a packet is received and the time the packet is processed” by separating packet headers and then processing the headers “**substantially in parallel.**” Russell, 1:33-35, Abstract, 3:13-14, claim 8 at 8:8-9. Such parallel processing would result in more efficient packet processing. A POSITA would have thus appreciated that each of the references—Cornett, Paatela, Nelson, and Russell—disclose mechanisms to accelerate packet processing. Cornett, [0097]; Paatela, [0076], Abstract, [0079]; Nelson, 1:13-20, 2:3-10; Russell, Abstract, 1:33-35, 3:13-14. Russell’s functionality of processing headers “substantially in parallel” would help achieve accelerated packet processing, which is the consistent theme of the references, and would encourage a POSITA to look at Russell’s teachings.

137. A POSITA would have had a reasonable expectation of success in combining Cornett, Paatela, and Nelson further with Russell at least because all the references seek to accelerate packet processing. Cornett, [0097]; Paatela, [0076], Abstract, [0079]; Nelson, 1:13-20, 2:3-10; Russell, Abstract, 1:33-35, 3:13-14. Paatela stores individual encapsulated packet headers in memory segments “in time

for...the **packet data in the memory 716 to be operated on by the editing processor 714.**” Paatela, [0076]. A POSITA would have understood that Paatela seeks to make each header available for processing as quickly as possible. Cornett also seeks to accelerate packet processing by using a data movement module, such as a DMA engine, to free up the host processor and move data more efficiently. Cornett, [0097]. Nelson similarly seeks to process packets faster by writing headers and frames into separate buffers **in parallel** so that routing decisions can be made based on header information, without the need to read the entire frame from the buffer. Nelson, 2:3-10. Russell furthers improved packet processing by processing separate packet headers “substantially in parallel.” Russell, claim 8 at 8:8-9, 1:33-35, Abstract. A POSITA would have thus understood that Russell would further accelerated packet processing that Cornett, Paatela, and Nelson seek to achieve. The combination would have been within the skillset of the POSITA.

138. It should be noted that Russell does not disclose a new or novel concept. Processing header portions in parallel was known in the art. Oskouy is a patent application example that discloses an in-line packet processing mechanism that involves “pre-process[ing]” packet headers to evaluate L2 and L3 header data for errors “in parallel.” Oskouy, [0118]. Another relevant example is the Lawton PCT publication that teaches using a header translator to perform look-ups on any combination of layer 2-4 protocol fields. Lawton, 3. Russell is just another example

and a POSITA would have viewed Russell as disclosing details that are obvious, but missing from Cornett, Paatela, and Nelson for how to further accelerate packet processing.

139. The combination is premised on the teachings of the references. It permits but does not require physical incorporation of elements from Paatela, Nelson, and Russell into Cornett. Nevertheless, the physical incorporation of Paatela's, Nelson's, and Russell's teachings for how to process individual encapsulated headers of Cornett's packets, and the timing of writing and processing the headers, would teach the claim limitations. It would amount to using Paatela's input controller and packet classification engine to "determine[] where one networking layer header ends and the next networking layer header begins." Paatela, [0077]. Then, based on this information, using Cornett's data movement module (e.g., DMA engine) to move each individual encapsulated header into a respective memory segment, per Paatela. Cornett, [0074], [0097]. Per Nelson, the write operations would be performed **in parallel** to accelerate packet processing. Nelson, 7:1-4. The individual packet headers, once stored, would then be processed by header processors "substantially **in parallel**," per Russell. Russell, claim 8 at 8:8-9. It would have been obvious to incorporate Paatela's, Nelson's, and Russell's teachings into Cornett's technique.

C. Detailed Analysis of Claim

140. This additional obviousness analysis is provided as an alternative to my earlier analysis of claim 8. *See* Section IX.E.15.

1. Claim 8

- a. **[8.0] *The method according to claim 1, further comprising concurrently processing said plurality of headers in each of a plurality of protocol stack layers.***

141. The combination of Cornett, Paatela, and Nelson does not explicitly explain the timing of when each individual packet header, that is stored in separate memory segments, per Paatela, is processed. This timing is explained in Russell.

142. Russell discloses an embodiment in connection with its Figure 5. In that embodiment, a network adapter 20—include in processing system 10, Russell, Figures 5 and 3—includes a “packet parser 30, a controller 52, first and second header processors 54,56, first and second header memories 58, 60, and a system interface 36.” Russell, 5:4-7. In operation:

packet parser 30 parses a received information packet 14,
and **directs a portion of the packet header to first
header memory 58** and directs the **remaining portion of
the packet header to second header memory 60.**

Russell, 5:7-10. A POSITA would have appreciated, or it would have been at least obvious, that Russell’s reference to “portions” of packet headers is meant to refer to

a packet's one or more encapsulated headers at least because, as I explain in my Background section, packets are known to have multiple headers.

143. Packet parser 30 knows how and where to store the packet header portions because controller 52 “programs packet parser 30 with the first header memory address [and] the second header memory address[.]” Russell, 5:17-21. Once the header portions have been stored, they are processed as follows:

First and second header processors 54, 56 are configured to process packet header information stored in header memories 58, 60 in accordance with one or more network protocol handling operations (e.g., firewall filtering, load balancing, and TCP/IP or RTP protocol handling operations).

Russell, 5:22-27.

144. Russell explains that the processors “each operable to process a respective one or more stored header components **substantially in parallel.**” Russell, claim 8 at 8:8-9.

145. As explained above in my Section X.B analysis, a POSITA would have been motivated to combine Cornett, Paatela, and Nelson further with Russell. Cornett is focused on reducing processing overhead and reducing processing time. Cornett, [0097]. Paatela similarly seeks to optimize packet processing in a solution where time is of the essence. Paatela, [0076] (packet 706 “is stored in the memory 716 **in time for** the editing instructions from the instruction memory 704 to be

processed by the editing processor 714, and for the packet data in the memory 716 to be operated on by the editing processor 714.”) Nelson, in turn, explains that writing a full packet frame into one buffer and the packet headers into another buffer, **in parallel**, accelerates packet processing and reduces processing overhead because routing decisions can be made without reading the full frames out of the frame buffer. Nelson, 2:3-10 (“wherein the frames of data are stored in a first-in-first-out buffer and wherein the header information is accessed to make **routing decisions in order to avoid having to read the frames out of the buffer.**”). And Russell teaches reducing packet processing latency by writing packet headers to separate memory locations and processing them “substantially in parallel.” Russell, Abstract, 3:13-14, 5:7-10, claim 8 at 8:8-9. A POSITA would thus have been motivated to combine Cornett, Paatela, and Nelson with Russell because Russell’s parallel processing of packet headers would further accelerate packet/header processing, which is a stated goal of Cornett, Paatela, and Nelson.

146. In the combination, and as initially explained in my analysis of limitation [13.3], Cornett’s NIC would cause its DMA engine to store the entire packet, with all its headers and payload, in a first buffer, the claimed *packet buffer memory*. Cornett’s NIC would further cause its DMA engine to store each individual encapsulated header into a respective segment of memory for processing, per Paatela, which are the claimed *respective protocol stack layer memory*. The NIC

would cause the DMA engine to perform both storage actions in parallel, i.e., *concurrently*, per Nelson. Then, per Russell, the individual packet headers would each be processed “substantially in parallel,” that is, *concurrently*. Russell, claim 8 at 8:8-9.

147. Russell teaches that each packet header is processed “**accordance with one or more network protocol handling operations** (e.g., firewall filtering, load balancing, and **TCP/IP or RTP protocol handling operations**).” Russell, 5:22-27. A POSITA would have known, based on my Background section, that each packet header would be processed by the appropriate layer protocol. A POSITA would have therefore understood that each of a packet’s *plurality of headers* are *process[ed]...in each of a plurality of protocol stack layers*.

XI. CONCLUSION

148. I declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true, and that these statements were made with knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under section 1001 of Title 18 of the United States Code.

Dated: Aug 23/2024

Nader F. Mir
Nader F. Mir, Ph.D.