UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

GOOGLE LLC,
Petitioner

v.

JENAM TECH, LLC,
Patent Owner

Patent No. 10,986,217

DECLARATION OF DR. BILL LIN

I. INTRODUCTION 1
II. BACKGROUND AND QUALIFICATIONS 2
III. MATERIALS CONSIDERED AND SUMMARY OF OPINIONS 5
IV. PERSON OF ORDINARY SKILL IN THE ART 11
V. TECHNICAL BACKGROUND 13
A. Computer Networks and Communication Protocols 13
1. Layered Protocols 16
2. Client-Server Architectures 26
3. Keepalive Mechanisms 31
4. Communication Protocols 36
a) TCP/IP 38
b) IPX/SPX 47
c) UDP/IP 48
d) RUDP 49
B. Hypertext Markup Language (HTML) 53
C. Virtual Network Computing (VNC) 54
VI. THE ’217 PATENT 57
VII. CLAIM CONSTRUCTION 64
VIII. OVERVIEW OF THE PRIOR ART 65
A. Overview of Morris (Ex. 1005) 65
B. Overview of Session Initiation Protocol (SIP) (Ex. 1006) 69
IX. THE PRIOR ART DISCLOSES AND/OR SUGGESTS ALL RECITED FEATURES OF CLAimS 1, 6, 12, 14, 17, AND 20 OF THE ’217 PATENT 75
A. The Combination of Morris and SIP Discloses and/or Suggests Claims 1, 6, 12, 14, 17, and 20 75
1. Claim 1 75
a) An apparatus, comprising: 78

b) a non-transitory memory storing instructions; and.....83

c) one or more processors in communication with the non-transitory memory, wherein the one or more processors execute the instructions to: 84

d) receive first information on which at least a first duration for detecting a first type of time period is based; 85

e) generate a first packet including a first parameter field identifying first metadata for use in determining a second duration for detecting the first type of time period;.....103

f) set up a first connection, by sending, from a first node to a second node, the first packet to provide the first metadata to the second node, for use by the second node in determining the second duration for detecting the first type of time period; 107

g) in response to detecting, based on the first duration and by the first node during at least a portion of the first connection including at least a portion of the first connection set up, a first time period of the first type of time period, at least partially close the first connection; and.....109

h) in response to detecting, based on the second duration and by the first node after the first duration is changed to the second duration, a second time period of the first type of time period, at least partially close the first connection;.....112

i) wherein the apparatus is configured such that: during the second time period of the first type of time period, an empty packet is sent without causing a timer utilized to detect the second time period to be reset;.....116

j) wherein the apparatus is configured such that: 117

k) wherein the apparatus is configured such that the first connection set up further includes, in addition to (a-1) sending the first packet: 125

2. Claim 6 126

a) The apparatus of claim 1, wherein the apparatus is configured such that the first duration is used and the second duration is not used, for detecting the first type of time period until being changed to the second duration. 126

3. Claims 12, 14 127

a) 12: The apparatus of claim 1, wherein the apparatus is configured such that the second duration is capable of being different from the first duration. 128

b) 14: The apparatus of claim 12, wherein the apparatus is configured such that the second duration is the same as the first duration, despite being capable of being different from the first duration. 128

4. Claim 17 130

a) The apparatus of claim 1, wherein the apparatus is configured such that the first type of time period is started in response to the received packet. 130

5. Claim 20 133

a) The apparatus of claim 1, wherein the apparatus is configured such that the first connection is not a Transmission Control Protocol (TCP) extension connection, and the first connection is not a TCP extension connection that involves a TCP extension. 133

X. DISCLOSURE OF CERTAIN CLAIM LIMITATONS IN THE ’217 PATENT AND ITS PARENT APPLICATIONS/PATENTS 136

A. Claim 1: No Support for Non-TCP Features Encompassed by “First Connection” 137

1. The ’824 Application and Its Priority Applications Are Limited to TCP-Based Embodiments 139

2. Materials Incorporated by Reference in the Priority Applications Do Not Support the Non-TCP Features 150

3. Mere Introduction of Non-TCP Type Terms In Summaries and Claims of Some Priority Applications Does Not Support the Non-TCP Features 151

4. Dependent Claims 6, 12, 14, 17, and 20 Are Not Supported 159

B. Claim 1: No Support for Detecting a Time Period During At Least A Portion of the First Connection Setup 159

1. The ’824 Application Does Not Support the Time Period Detection During Connection Setup Features of Claim 1 161

2. The Priority Applications Do Not Support the Time Period Detection During Connection Setup Features of Claim 1 167

3. Dependent Claims 6, 12, 14, 17, and 20 Are Not Supported 168

XI. CONCLUSION.....169

I, Bill Lin, declare as follows:

I. INTRODUCTION

1. I have been retained by Google LLC ("Petitioner") as an independent expert consultant in this proceeding before the United States Patent and Trademark Office ("PTO") regarding U.S. Patent No. 10,986,217 ("the '217 patent") (Ex. 1001). I have been asked to consider whether prior art references disclose or suggest the features recited in claims 1, 6, 12, 14, 17, and 20 ("the challenged claims") of the '217 patent. My opinions are set forth below.

2. I am being compensated at my standard consulting rate for the time I spend on this matter. My compensation is in no way contingent on the nature of my findings, the presentation of my findings in testimony, or the outcome of this or any other proceeding. I have no other interest in this proceeding.

II. BACKGROUND AND QUALIFICATIONS

3. In this section, I have summarized my educational background, career history, publications, and other relevant qualifications. My Curriculum Vitae, which states my qualifications more fully, is filed as a separate Exhibit (Ex. 1003).

4. I am currently a Professor of Electrical and Computer Engineering at the University of California, San Diego (UCSD).

5. I received a Bachelor of Science in Electrical Engineering and Computer Sciences from the University of California, Berkeley in May 1985; a Master of Science in Electrical Engineering and Computer Sciences from the University of California, Berkeley in May 1988; and a Ph.D. in Electrical Engineering and Computer Sciences from the University of California, Berkeley in May 1991.

6. I joined UCSD in 1997, and I have been a tenured professor since 1999. My teaching and research focus on computer architecture and computer network problems, including the design of high-performance switches and routers, high-speed packet processors, wireless networks, data networks, and various hardware accelerators for networking applications. I regularly teach a senior-level design course on the design of advanced processors, and I have taught graduate courses and advanced special topics in computer architecture and computer networks.

7. At UCSD, I am a Principal Investigator in the UCSD Center for Networked Systems (CNS). CNS brings together researchers to work on a range of challenges in the design of future networked systems. My contribution to CNS has been expertise in the design of computer architecture solutions for packet processing, computer networking, and network measurements. I am also a Principal Investigator in the UCSD Center for Wireless Communications (CWC). CWC brings together researchers to work on a range of challenges in the design of future wireless communications systems. My contribution to CWC has been expertise in the design of computer architecture solutions for wireless networking and mobile computing.

8. Prior to joining UCSD, I was the Head of the Systems and Communications Group at IMEC in Leuven, Belgium, where I led a team of researchers who worked on a range of computer design problems, including specialized processors for wireless communications and computer networking.

9. During my career, I have received or worked on research efforts that received millions of dollars in research funding from both government agencies and industry, including funding for research in packet processing, computer networks, network measurements, and wireless communications.

10. I have served as an Associate or Guest Editor for several journals published by the Association for Computing Machinery ("ACM") and the Institute of Electrical and Electronics Engineers ("IEEE"). I have also served as General

Chair of several ACM/IEEE conferences, and on the Organizing or Steering Committees of many ACM/IEEE conferences, and I have served on the Technical Program Committees of numerous ACM/IEEE conferences. I am the author of over 190 top-tier peer-reviewed publications in the field of computer engineering dating to the 1980s, including journal articles, conference papers, book chapters, technical reports, and invited papers. A number of these publications have received best paper awards or distinguished paper citations. I have also given numerous invited and keynote talks around the world.

11. I am a named inventor on five patents in the fields of computer design and computer networking: U.S. Patent Nos. 8,443,444, 7,860,004, 7,672,005, 5,870,588, and 5,748,487.

III. MATERIALS CONSIDERED AND SUMMARY OF OPINIONS

12. The opinions contained in this Declaration are based on the documents I reviewed and my professional judgment, as well as my education, experience, and/or knowledge regarding technologies relating to, among other things, networking technologies.

13. In forming my opinions expressed in this Declaration, I reviewed the following materials (and any other materials that I reference in this Declaration):

EX1001	U.S. Patent No. 10,986,217
EX1003	Curriculum Vitae of Bill Lin, Ph.D.
EX1004	File History of U.S. Patent Application No. 17/148,824 (U.S. Patent No. 10,986,217)
EX1005	U.S. Publication No. 2011/0213820 to Morris
EX1006	IETF RFC 3261 "SIP: Session Initial Protocol"
EX1007	U.S. Patent No. 6,981,048 to Abdolbaghian
EX1008	DARPA RFC 793 "Transmission Control Protocol"
EX1009	IETF RFC 1122 "Requirements for Internet Hosts -- Communication Layers"
EX1011	IETF RFC 1001 "Protocol Standard for a NetBIOS Service On A TCP/UDP Transport: Concepts and Methods"
EX1012	U.S. Patent No. 6,212,175 to Harsch
EX1013	U.S. Patent No. 6,665,727 to Hayden
EX1014	U.S. Patent No. 7,636,805 to Rosenberg

EX1015	IETF RFC 5440 "Path Computation Element (PCE) Communication Protocol (PCEP)"
EX1016	The Xpress Transfer Protocol (XTP): A tutorial (short version), Sanders et al. (January, 1990)
EX1018	L. Eggert, TCP Abort Timeout Option, draft-eggert-tcm-tcp-abort-timeout-option-00, Network Working Group, Internet-Draft (April 14, 2004) ("*Eggert*")
EX1019	U.S. Publication No. 2007/0110046 to Farrell *et al.*
EX1020	U.S. Pre-Grant Publication No. 2004/0093376 to De Boor et al.
EX1021	U.S. Patent 7,535,913 to Minami et al.
EX1022	U.S. Pre-Grant Publication No. 2005/0204013 to Raghunath et al.
EX1023	U.S. Pre-Grant Publication No. 2007/0005804 to Rideout
EX1024	U.S. Pre-Grant Publication No. 2004/0098748 to Bo et al.
EX1025	IETF RFC 2616 "Hypertext Transfer Protocol -- HTTP/1.1"
EX1026	U.S. Patent No. 8,259,716 to Diab
EX1027	Bova et al., RELIABLE UDP PROTOCOL <draft-ietf-sigtran-reliable-udp-00.txt> 25 February 1999
EX1028	U.S. Patent No. 6,674,713 to Berg et al.
EX1034	U.S. Publication No. 2010/0202366 to Tseng
EX1035	IETF RFC 2119 "Key words for use in RFCs to Indicate Requirement Levels"
EX1036	U.S. Patent No. 10,306,026
EX1037	U.S. Patent No. 10,742,774
EX1038	File History of U.S. Patent Application No. 12/714,454 (U.S. Patent No. 8,219,606)
EX1039	File History of U.S. Patent Application No. 13/477,402

EX1040	Computer-generated Redline Between U.S. Patent Application No. 12/714,454 and U.S. Patent Application No. 13/477,402
EX1041	File History of U.S. Patent Application No. 12/714,063
EX1042	Computer-generated Redline Between U.S. Patent Application No. 12/714,454 and U.S. Patent Application No. 12/714,063
EX1043	File History of U.S. Patent Application No. 14/667,642
EX1044	File History of U.S. Patent Application No. 15/694,802 (U.S. Patent No. 9,923,995)
EX1045	File History of U.S. Patent Application No. 15/915,047 (U.S. Patent No. 10,075,564)
EX1046	File History of U.S. Patent Application No. 16/040,522 (U.S. Patent No. 10,375,215)
EX1047	File History of U.S. Patent Application No. 16/368,811 (U.S. Patent No. 10,742,774)
EX1048	File History of U.S. Patent Application No. 16/914,267
EX1049	Computer-generated Redline Between U.S. Patent Application No. 13/477,402 and U.S. Patent Application No. 14/667,642
EX1050	Computer-generated Redline Between U.S. Patent Application No. 12/714,454 and U.S. Patent Application No. 15/694,802 (U.S. Patent No. 9,923,995)
EX1051	Computer-generated Redline Between U.S. Patent Application No. 15/694,802 (U.S. Patent No. 9,923,995) and U.S. Patent Application No. 15/915,047 (U.S. Patent No. 10,075,564)
EX1052	Computer-generated Redline Between U.S. Patent Application No. 15/694,802 (U.S. Patent No. 9,923,995) and U.S. Patent Application No. 16/040,522 (U.S. Patent No. 10,375,215)
EX1053	Computer-generated Redline Between U.S. Patent Application No. 15/694,802 (U.S. Patent No. 9,923,995) and U.S. Patent Application No. 16/368,811 (U.S. Patent No. 10,742,774)

EX1054	Computer-generated Redline Between U.S. Patent Application No. 15/694,802 (U.S. Patent No. 9,923,995) and U.S. Patent Application No. 16/914,267
EX1055	Computer-generated Redline Between U.S. Patent Application No. 15/694,802 (U.S. Patent No. 9,923,995) and U.S. Patent Application No. 17/079,397 (U.S. Patent No. 10,951,742)
EX1056	IETF RFC 3168 "The Addition of Explicit Congestion Notification (ECN) to IP"
EX1057	IETF RFC 6093 "On the Implementation of the TCP Urgent Mechanism"
EX1058	IETF RFC 6528 "Defending against Sequence Number Attacks"
EX1059	Excerpt of IETF webpage hosting RFC 793, retrieved May 2, 2021
EX1060	RFC 768 USER DATAGRAM PROTOCOL
EX1061	DARPA RFC 792 INTERNET CONTROL MESSAGE PROTOCOL
EX1062	File History of U.S. Provisional Patent Application No. 62/137,173
EX1063	File History of U.S. Patent Application No. 14/294,059
EX1064	File History of U.S. Patent Application No. 12/857,851
EX1065	File History of U.S. Patent Application No. 13/941,502
EX1066	File History of U.S. Patent Application No. 12/789,550
EX1067	File History of U.S. Patent Application No. 12/857,847
EX1068	File History of U.S. Patent Application No. 12/857,857
EX1069	File History of U.S. Patent Application No. 12/857,836
EX1070	File History of U.S. Patent Application No. 12/788,364 (U.S. Patent No. 8,346,853)
EX1071	File History of U.S. Patent Application No. 12/788,373
EX1072	File History of U.S. Patent Application No. 12/789,538

EX1073	File History of U.S. Patent Application No. 12/788,381
EX1074	File History of U.S. Patent Application No. 12/789,568
EX1076	File History of U.S. Patent Application No. 17/079,397 (U.S. Patent No. 10,751,742)
EX1077	Computer-generated Redline Between U.S. Patent Application No. 15/694,802 (U.S. Patent No. 9,923,995) and U.S. Patent Application No. 17/148,824 (U.S. Patent No. 10,986,217)
EX1083	U.S. Pre-Grant Publication No. 2005/0255870 to Chang et al.
EX1084	International Publication No. WO 2019/125260 to Stille et al.
EX1085	U.S. Patent No. 6,810,263 to Cheng et al.
EX1086	U.S. Pre-Grant Publication No. 2006/0191003 to Bahk et al.
EX1087	U.S. Patent No. 10,951,742
EX1090	IETF RFC 1323 “TCP Extensions for High Performance”

14. In support of my opinions, I have taken into account how a person of ordinary skill in the art (as defined below in Section IV) would have understood the claims and the specification of the ’217 patent at the time of the alleged invention. My opinions take into account how a person of ordinary skill in the art would have understood the ’217 patent and the state of the art at the time of the alleged invention.

15. Based on my experience and expertise, it is my opinion (1) that a person of ordinary skill in the art would not have understood that the named inventor of the ’217 patent was in possession of the subject matter recited in the challenged claims

of the ’217 patent (and that a person of ordinary skill in the art would have found that the specification and drawings of the ’217 patent and its priority applications do not support what is recited in the challenged claims) and (ii) that certain references disclose and/or suggest all the features recited in the challenged claims of the ’217 patent, as I discuss in detail below.

IV. PERSON OF ORDINARY SKILL IN THE ART

16. I have considered the time of the alleged invention for the ’217 patent to be around the early 2021 timeframe, including up to January 14, 2021, the filing date of U.S. Application No. 17/148,824 (“the ’824 application”), which issued at the ’217 patent. (Ex. 1001 at Cover.) I may refer to this time frame in this declaration as the time of the alleged invention, the relevant time, or the time of the ’217 patent.

17. Based on my knowledge and experience, I understand what a person of ordinary skill in the art (“POSITA”) would have known at the time of the alleged invention of the ’217 patent. In my opinion, based on my review of the ’217 patent, the technology, the educational level and experience of active workers in the field, the types of problems faced by workers in the field, the solutions found to those problems, the sophistication of the technology in the field, and drawing on my own experience, a person of ordinary skill in the art would have had an undergraduate degree in Electrical Engineering, Computer Engineering, Computer Science or a related field along with at least 2 years of work experience in the field of networking, or in the alternative, equivalent experiences, such as 6 years of work or research experience in the field of networking. (*See*, *e.g.*, Ex. 1001, 1:66–2:50 (discussing the background of the ’217 patent), 2:54–3:13 (discussing a “summary of the disclosure”).)

18. My analysis of the ’217 patent and my opinions in this declaration are from the perspective of one of ordinary skill in the art, as I have defined it above, at the time of the alleged invention for that patent.

19. I note that my opinions in this declaration do not change even if the time of the alleged invention for the ’217 patent is considered to be around the early 2010 timeframe, including up to the February 27, 2010 filing date of U.S. Application No. 12/714,454 (“the ’454 application”), which I understand is the earliest application to which the ’824 application claims priority.

V. TECHNICAL BACKGROUND

20. In this section, I discuss the state of the art relevant to the subject matter of the ’217 patent as would have been known, understood, and appreciated by a person of ordinary skill in the art prior to and at the time of the alleged invention for the ’217 patent (e.g., around early 2021 timeframe, including up to January 14, 2021, as I mentioned above in Section IV). I rely on, and incorporate as applicable (even if not expressly mentioned below in Section IX), the following disclosures and opinions to support my opinions in this Declaration, including those opinions relating to how the prior art discloses and/or suggests the challenged claims of the ’217 patent and how and why a person of ordinary skill in the art would have been motivated to consider and combine the disclosures and suggestions from that prior art as I explain below in Section IX. The concepts below would have been within the knowledge and mindset of a person of ordinary skill in the art at the time.

A. Computer Networks and Communication Protocols

21. It was known at the time of the alleged invention to configure and operate a group of computers according to the well-known client-server architecture. (Ex. 1021 (Minami), 1:20–33 (below).) Data communications between such computers at the time often involved known switched packet network features. In such known configurations, data, including images, voice, video, and text, is first broken up into chunks, known as “packets.” The resulting data packets are

transported between computers (also known as "nodes") over the network using a communication protocol standard. (*Id*.) At the time, the transported packets were formatted in ways that enabled the network to know which node should receive the packet and so that the node will know how to process the packet when it receives it.

22. Many different communication protocol standards were known and used at that time to facilitate communications in such computer networks. (*Id*.) A communication protocol was known (and still is known) as a system of rules, syntax, and semantics that allow two or more computing devices of a communications system to transmit information. (*Id*.) For example, several references provided exemplary descriptions of common network protocol features consistent with that known by those of ordinary skill in the art at the time:

> **Computer networks necessitate the provision of various communication protocols to transmit and receive data.** Typically, a computer network comprises a system of devices such as computers, printers and other computer peripherals, communicatively connected together. **Data are transferred between each of these devices through data packets which are communicated through the network using a communication protocol standard. Many different protocol standards are in current use today. Examples of popular protocols are Internet Protocol (IP), Internetwork Packet Exchange (IPX), Sequenced Packet Exchange (SPX), Transmission Control**

> **Protocol (TCP), and Point to Point Protocol (PPP)**. Each network device contains a combination of hardware and software that translates protocols and processes data.

(Ex. 1021 (Minami), 1:20–33 (excerpted with emphasis added).)

> **Communications networks use protocols to transmit and receive data.** Typically, **a communications network comprises a collection of network devices**, also called nodes, such as computers, printers, storage devices, and other computer peripherals, communicatively connected together. Data is transferred between each of these network devices using data packets that are transmitted through the communications network using a protocol**. Many different protocols are in current use today. Examples of popular protocols include the Internet Protocol (IP), Internetwork Packet Exchange (IPX) protocol, Sequenced Packet Exchange (SPX) protocol, Transmission Control Protocol (TCP), Point-to-Point Protocol (PPP) and other similar new protocols that are under development.** A network device contains a combination of hardware and software that processes protocols and data packets.

(*Id*., 2:23–38 (excerpted with emphasis added).)

> **Modern computer networks enable communication between computers in part by assigning each connected computer an address** (for example, an Internet Protocol (IP) address), **and one or more ports through which communication may**

> **proceed between the assigned IP addresses. Once a logical connection between computers is established, various techniques assure the authenticity of the ongoing information transfers, for example assigning sequence numbers to packets forming part of an ongoing connectio**n. …

(Ex. 1014 (Rosenberg), 2:47–62 (excerpted with emphasis added).)

> Several computer systems may be coupled together through a network, such as the Internet. The term "Internet" as used herein refers to a network of networks which uses certain protocols, such as the TCP/IP protocol, and possibly other protocols such as the hypertext transfer protocol (HTTP) for hypertext markup language (HTML) documents that make up the World Wide Web (web). The physical connections of the Internet and the protocols and communication procedures of the Internet are well known to those of skill in the art.

(Ex. 1023 (Rideout) ¶[0385] (excerpted).)

1. Layered Protocols

23. One primary goal of computer networking was to provide fast and reliable data communications between computer systems. Interoperability has been accomplished through adherence to standards, and performance has steadily increased through new technology and optimizations of hardware and software. The use of layered protocol architectures to achieve those goals has been well-known since at least the 1980s.

24. The two most dominant models of protocol layers that were known at the time (and still today) are the OSI model and the TCP/IP model. (*E.g.*, Ex. 1026 (Diab), 1:66–2:14; *see also* Ex. 1021 (Minami), 2:39–60.) The OSI model defines a seven-layer protocol stack, whereas the TCP/IP model defines a simpler four-layer protocol stack. The OSI model includes physical, data link, network, transport, session, presentation, and application layers. The TCP/IP model includes a link, internet, transport, and application layers. The figure below shows the relationship between the OSI layering and the TCP/IP layering that was known to one of ordinary skill in the art at the time.

OSI Model and TCP/IP Model	
OSI Model	**TCP/IP Model**
Application Layer	**Application Layer**
Presentation Layer	
Session Layer	
Transport Layer	**Transport Layer**
Network Layer	**Internet Layer**
Data Link Layer	**Link layer**
Physical Layer	

25. At a conceptual level, each layer is responsible only for its respective functions. This enables, for example, hiding the complexity of the physical data connection (used to actually transmit the data onto the physical wires) from layers above the physical, data link, and network layers above. Likewise, the lower layers facilitate the transmission of the data on the physical wires, but need not be concerned about what application the data belongs to or how the user data has been partitioned into individual packets.

26. The bottom "link" layer of the TCP/IP model has roughly the same functions as the "data link" and "physical" layers of the OSI reference model. The next two layers up, the "transport" and "Internet" layers, have roughly the same

functions as their similarly named counterparts in the OSI model (namely, the "transport" and "network" layers).

27. In the TCP/IP model, the "Internet" layer was also known as the "Internet Protocol" (IP) layer, which provides for Internet addressing and routing functions. The "transport" layer is concerned with conveying a message from one application to another over a network. One of ordinary skill in the art would have recognized and understood at the time that the two most widely used transport protocols are TCP and UDP, either of which was known to be used to transport application-layer messages. TCP was known to provide a connection-oriented service to its applications that include guaranteed delivery and congestion control. UDP was known to provide a simpler connectionless service that does not provide for reliability or flow control.

28. Compared to the TCP/IP model, which includes an application layer above the transport layer, the OSI model has two additional layers above the "transport" layer; the session layer and the presentation layer, before the application layer. The session layer controls the connections between computers, and the presentation layer establishes context between application-layer entities. These two layers were not (and are not) present in the TCP/IP model.

29. Below I provide a brief summary of the OSI model's seven-layer functions:

Layer		Exemplary Function
7	Application	refers to the high-level Application Programming Interfaces (APIs), including resource sharing, remote file access, file sharing, message handling, printing, and so on
6	Presentation	controls the manner of representing the data and ensures that the data is in the correct form (e.g., provides the translation of data between a networking service and an application)
5	Session	allows for two way communications (e.g., back-and-forth transmissions) between two nodes
4	Transport	ensures reliable delivery of data packets between nodes on a network
3	Network	builds data packets following a specific protocol
2	Data Link	controls the transmission of data between two nodes connected by a physical layer in discrete forms known as frames that contain data packets
1	Physical (PHY)	defines the physical components connecting the network device to the network;

30. It was known prior to and at the time of the alleged invention for the ’217 patent that each intermediate layer in the OSI model serves a class of functionality to the layer above it and is served by the layer below it. (Ex. 1026 (Diab), 1:66–2:14, 2:15–3:2.) These classes of functionality are realized in software by standardized communication protocols. (*See, e.g.*, Ex. 1021 (Minami), 2:39–60, 2:61–3:24.) For example, prior references described the OSI model in a manner consistent with how a person of ordinary skill in the art would have understood at the time of the alleged invention.

> **Most networks adhere to the layered approach provided by the open systems interconnect (OSI) reference model.** The OSI reference provides a seven (7) layer approach, which

> includes an application layer, (Layer 7), a presentation layer (Layer 6), a session layer (Layer 5), a transport layer (Layer 4), a network layer (Layer 3), a data link layer (Layer 2) and a physical layer (Layer 1). Layer 7 through layer 5 inclusive may comprise upper layer protocols, while layer 4 through layer 1 may comprise lower layer protocols. Some networks may utilize only a subset of 7 layers. For example, the TCP/IP model, or Internet Reference model generally utilizes a fiver layer model, which comprises an application layer, (Layer 7), a transport layer (Layer 4), a network layer (Layer 3), a data link layer (Layer 2) and a physical layer (Layer 1). These five layers can be broken down into a fairly specific set of responsibilities or services, which they provide.

(Ex. 1026 (Diab), 1:66–2:14 (emphasis added).)

> **In 1978, the International Standards Organization (ISO), a standards setting body, created a network reference model known as the Open System Interconnection (OSI) mode**l. The OSI model includes seven conceptual layers: 1) The Physical (PHY) layer that defines the physical components connecting the network device to the network; 2) The Data Link layer that controls the movement of data in discrete forms known as frames that contain data packets; 3) The Network layer that builds data packets following a specific protocol; 4) The Transport layer that ensures reliable delivery of data packets; 5) The Session layer that allows for two way communications between network devices; 6) The Presentation layer that controls the manner of

> representing the data and ensures that the data is in correct form; and 7) The Application layer that provides file sharing, message handling, printing and so on. Sometimes the Session and Presentation layers are omitted from this model. For an explanation of how modern communications networks and the Internet relate to the ISO seven-layer model see, for example, chapter 11 of the text "Internetworking with TCP/IP" by Douglas E. Comer (volume 1, fourth edition, ISBN 0201633469) and Chapter 1 of the text "TCP/IP Illustrated" by W. Richard Stevens (volume 1, ISBN 0130183806).

(Ex. 1021 (Minami), 2:39–60 (excerpted with emphasis added); *see also id*., 2:61–3:24 ("the Network, Transport, Session, Presentation and Application layers. The Network, Transport, Session, and Presentation layers, are implemented using protocol-processing software, also called protocol stacks.").)

31. The OSI model's Layer 7 (the application layer) was known at the time to be typically responsible for supporting network applications such as web browsers over Hypertext Transfer Protocol (HTTP) (Ex. 1025 (RFC 2616), 1) and email clients over Simple Mail Transfer Protocol (SMTP). (Ex. 1026 (Diab), 2:15–20.) The applications were known to be typically implemented in software in the network's end systems, such as personal computers and servers. (*Id*.) The OSI model's Layer 6 (the presentation layer) was known to be typically responsible for masking any differences in data formats that may occur between dissimilar or

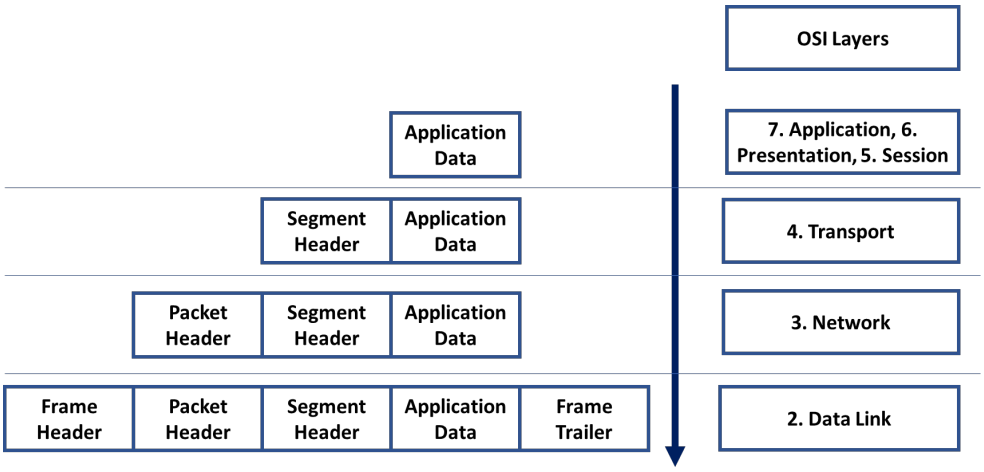
disparate systems. (*Id*., 2:21–26.) The presentation layer specifies architecture-independent data transfer formats. (*Id*.) The OSI model's Layer 5, the session layer, was known to be typically responsible for managing user session dialogues. (*Id*.) In this regard, the session layer may be enabled to control the establishment and/or termination of logical links between users. (*Id*., 2:26–32.) The session layer may also be enabled to provide handling and reporting of upper layer errors. (*Id*.)

32. The OSI model's Layer 4 (the transport layer) was known to be typically responsible for passing application layer messages between the client and server sides of an application. (*Id*., 2:33–42.) In this regard, the transport layer may be enabled to manage the end-to-end delivery of messages in the network. (*Id*.) The transport layer was known to comprise various error recovery and/or flow control mechanisms, providing reliable delivery of messages. (*Id*.) Two standard Layer 4 protocols used on the Internet at the time is the transmission control protocol (TCP) and the user datagram protocol (UDP). (*Id*.)

33. The OSI model's Layer 3 (the network layer) was known to be typically responsible for determining how data may be transferred between network devices. (*Id*., 2:43–52.) One standard Layer 3 protocol known at the time is the Internet Protocol (IP). (*Id*.) The IP defines the form and content of the datagrams. (*Id*.) The OSI model's Layer 2 (the data link layer) was known to be typically responsible for moving a data packet from one node to another. (*Id*., 2:53–60.) The Ethernet (IEEE

802.3) protocol is one common link-layer protocol that was known and used at the time in computer networks. (*Id*.) The OSI model's Layer 1 (the physical layer) was known to be typically responsible for defining the physical means, which may comprise optical, electrical and/or mechanical means for communicating data via network devices over a communication medium. (*Id*., 2:61–3:2.)

34. A person of ordinary skill in the art would have known that whenever data was sent from one node to another in a network configured according to the OSI model, the data was encapsulated at the sender's side and de-encapsulated at the receiver's end. (*E.g.*, *id.*, 13:25–31 ("payloads may be formatted and/or encapsulated according to one or more protocols. Other protocols may be utilized for the packetization and/or conveyance of data.").) The encapsulation of data at various layers of the implementing model (OSI) was known to add multiple functionalities and features to the data transmission. (E.g., *see* my discussion below regarding RUDP). For example, below, I have illustrated an exemplary process of encapsulating data at the various layers of the OSI model as was known to those of ordinary skill in the art at the time:



35. As I exemplify above, starting at the top, each lower layer takes the entire contents of the previously received message from its upper layer as data and then encapsulates it into its message format, for example, by adding a header that contains layer-specific control information. At each layer in this process, layer-specific information is added to the received data from an upper layer. This information may include physical addresses, network addresses, port numbers, application-specific addresses, etc.

36. For example, the full packet to be transmitted over the network is located at the data link layer, as I exemplify in the illustration above. The "Frame Header" in this packet contains the sending and receiving nodes' local network addresses. This packet also includes its upper layer's packet (e.g., the network layer packet.) For example, the "Packet Header" contains the IP source and destination addresses of the sending and receiving nodes. The "Segment Header" includes the information used by the transport protocol (e.g., TCP or UDP). The "Application

Data" consists of the application's information to provide its services and the data the application is transferring.

2. Client-Server Architectures

37. As I mentioned above, a typical computer network architecture used prior to and at the time of the alleged invention was the client-server architecture. The client-server architecture was known as a computing model in which the server computer hosts, delivers, and manages client computer resources and services. (Ex. 1012 (Harsch), 2:7–18.) In a client-server architecture, there may be many clients (e.g., remote computing devices) that can request resources from one or more servers, each configured to operate with the protocols to receive and process the respective client requests. (*See, e.g., id.*) For example, one common scenario is where the client computer can request and view HTML pages fetched from a server (e.g., a webserver) with the appropriate web browsing software. (Ex. 1023 (Rideout) ¶¶[0387]–[0388].)

38. It was known at the time that in many scenarios using this architecture, following a connection being established between the client and the server node(s), a client was typically prompted to provide credentials to ensure communications were allowed and proper. For example, it was known at the time to configure the system in such an architecture to allow the client to log in and begin a session with a particular server. (Ex. 1012 (Harsch), 1:59–62.) In such arrangements, a session

was known to be initiated once at start up by the client and remain active until the client or server ends the session. (*Id*.) For example, Rosenberg (Ex. 1014) explains such features consistent with that known by a person of ordinary skill in the art at the time.

> **When communication between a first host computer and a second host computer is desired, and the computers are connected to each other through a communications network** without an intervening firewall or an intervening device performing NAT, **either host may initiate the connection by simply sending a suitable message addressed to a suitable port for the message type at the address of the other host computer**. **Communication using a client/server architecture,** or a peer-to-peer architecture, or any other suitable architecture can be initiated in this way, absent an intervening firewall or NAT device.

(Ex. 1014 (Rosenberg), 2:63–3:5 (excerpted with emphasis added).)

> To make a conventional feature such as worldwide web browsing work, several operations occur. **Connections from a local host computer to a remote server are initiated by the local computer.** If the local computer is behind a firewall, the local computer initiates communication with a desired server, through the firewall. By initiating the communication through the firewall, the local computer instructs the firewall to allow certain types of communications back from the server for a certain

> period of time. When the server replies using a correct address, port and sequence number, the firewall recognizes the response as expected and passes the response on to the local computer. While the server may also be located behind a firewall, that firewall conventionally has a known port open to inbound traffic, so that certain types of contact with the server are permitted by the firewall.

(*Id.*, 3:6–22 (excerpted with emphasis added).)

39. It was also known at the time that to ensure proper routing of messages between a server node and client node, "the messages are initially broken up into data packets, each of which receive[s] a destination address according to a consistent protocol, and which are reassembled upon receipt by the target computer." (*See* Ex. 1012 (Harsch), 2:18–24; *see also* Ex. 1021 (Minami), 1:20–3, 2:23–3.)

40. As I have explained, data exchanged between nodes in a network was known to be achieved via a "protocol." (*E.g.*, Ex. 1012 (Harsch), 2:18–24; Ex. 1021 (Minami) at 2:23–3.) A commonly accepted network protocol for this purpose at the time was the Internet Protocol (IP). (Ex. 1012 (Harsch), 2:18–24.) The IP was known to be combined with a transport layer protocol, such as the Transmission Control Protocol (TCP), which provides a reliable stream delivery of messages, or a User Datagram Protocol (UDP) which allows for distinguishing messages among multiple destinations with a given host computer. (*Id.*) I will explain both of these

protocols further below. There are many different protocol standards, including alternatives to the TCP and UDP, such as Internetwork Packet Exchange (IPX), Sequenced Packet Exchange (SPX), and Reliable User Datagram Protocol (RUDP), which I also discuss below. (*E.g.*, Ex. 1021 (Minami), 1:20–33, 2:23–33.) Harsch (Ex. 1012) also explains:

> In order to ensure proper routing of messages between the server and an intended mobile communication unit, **the messages are initially broken up into data packets, each of which receive a destination address according to a consistent protocol, and which are reassembled upon receipt by the target computer.** The exchange of information between endpoints in a packet network is achieved via a "protocol." A commonly accepted protocol for this purpose is the Internet Protocol (IP), which provides for networking. **Used in conjunction with the IP may be a Transmission Control Protocol (TCP) which provides for a reliable stream delivery of messages or a User Datagram Protocol (UDP)** which allows for distinguishing messages among multiple destinations with a given host computer.

(Ex. 1012 (Harsch), 2:18–32 (emphasis added).) Minami (Ex. 1021) also explains that:

> **Many different protocol standards are in current use today. Examples of popular protocols are Internet Protocol (IP), Internetwork Packet Exchange (IPX), Sequenced Packet**

> **Exchange (SPX), Transmission Control Protocol (TCP), and Point to Point Protocol (PPP)**. Each network device contains a combination of hardware and software that translates protocols and processes data.

(Ex. 1021 (Minami), 1:20–33 (excerpted with emphasis added).) Minami further explains:

> Communications networks use protocols to transmit and receive data. Typically, a communications network comprises a collection of network devices, also called nodes, such as computers, printers, storage devices, and other computer peripherals, communicatively connected together. Data is transferred between each of these network devices using data packets that are transmitted through the communications network using a protocol. Many different protocols are in current use today. **Examples of popular protocols include the Internet Protocol (IP), Internetwork Packet Exchange (IPX) protocol, Sequenced Packet Exchange (SPX) protocol, Transmission Control Protocol (TCP),** Point-to-Point Protocol (PPP) and other similar new protocols that are under development. A network device contains a combination of hardware and software that processes protocols and data packets.

(Ex. 1021 (Minami), 2:23–38 (excerpted with emphasis added).)

3. Keepalive Mechanisms

41. Prior to and at the time of the alleged invention, it was known that to maintain an established connection between two nodes in a network, a node may be configured to employ a keepalive mechanism to allow an idle TCP connection to stay connected and not time out periodically. (*E.g.*, Ex. 1021 (Minami), 26:29–34 (excerpted below).) A keepalive mechanism was a feature known to those of ordinary skill in the art that allows a node to periodically send a packet (e.g., a keepalive packet) to the other node based on a timer (e.g., a keep alive timer). (*E.g.*, *id.*, 26:29–34 (excerpted below), 42:10–14 (excerpted below); Ex. 1012 (Harsch), 2:45–3:4 (excerpted below), 3:17–25 (excerpted below); Ex. 1011 (RFC 1001), 54.) Those of ordinary skill in the art would have also understood that such keepalive messages are used to determine if the other node is still connected and operational. (Ex. 1012 (Harsch), 3:5–16 (excerpted below).) For example, when a node, which has employed a keepalive mechanism, does not receive a data packet within a specified period of time, it will send a keepalive message to the other node to assess its activity. Upon receiving a keepalive message, the other node, if active, returns an acknowledgment packet. Minami and Harsch describe such features in a manner consistent with that known by those of ordinary skill in the art at the time:

> Keepalive
>
> Keepalive is an enhanced function of TCP described in RFC1122 section 4.2.3.6. See http://www.faqs.org/rfcs/rfc1122.html. **Keep alive allows an idle TCP connection to stay connected and not time out by periodically sending a keep alive packet across the link.**

(Ex. 1021 (Minami), 26:29–34 (excerpted with emphasis added).)

> Keep Alive Time 0x05 [31:24], 8 bits)
>
> **This field represents the time in the future when the keep alive trigger is asserted.** This time is reset every time a packet is sent on the socket or a pure ACK packet is received. The time is represented here in units of minutes.

(*Id.*, 42:10–14 (emphasis added).)

> … To avoid this problem, servers (i.e., host computers) are often configured to employ a mechanism deemed a "keepalive" probe." **The keepalive probe consists of several IP packets, sent in a burst, by the server. The probe determines if the client (mobile communication unit) is still connected to the LAN and operational.** The IP packets of keepalive probes differ from ordinary IP packets in that they do not increment a sequence number that synchronize the transfer of bytes in a data stream between the two end points as a convention packet using the TCP protocol would. In this manner, such IP packet is 'invisible' or 'transparent'. Generally, the IP packet sequence number is one

> less than the number that the receiving node expects to receive. This has two effects: (1) the receiving node will immediately return an acknowledgment packet to the sender; and (2) the IP packet does not advance the sequence number of the receiving node and therefore it does not change the synchronization state between the two end points.

(Ex. 1012 (Harsch), 2:45–3:4 (excerpted with emphasis added).)

42. The keepalive mechanism was known to often use a keep alive timer that triggers when the keepalive packet is to be sent to the other node. (Ex. 1021 (Minami), 67:1–30 (excerpted below).) The keep alive timer was known to be set to a specified keep alive time that must elapse on an idle connection before a keep alive packet is sent. (Ex. 1014 (Rosenberg), 4:38–43 (excerpted below).)

> Keep Alive Timer for TCP
>
> Overview
>
> **This section describes the keepalive timer used for TCP connections.** This section contains the theory of

operation on how this feature is implemented in the IT 10G network stack.

Keep Alive Parameters

The following parameters are used for the keepalive feature in the network stack:

TABLE 24

Keep Alive Parameters

Parameter	Description	Default Value
Keep Alive Enable	Enable bit for the keep alive function. If the feature is not enabled, then an idle socket will stay up until it is closed in the regular manner.	off
Keep Alive Time[7:0]	This fields represents the number of minutes that must elapse on an idle connection before a keep alive probe is sent on the socket.	0x78 (120 minutes or 2 hours)
Keep Alive Interval [7:0]	This field represents the time interval in number of minutes between keep alive probes on a single socket.	0x02 (2 minutes)
Keep Alive Retries[3:0]	This field represents the number of keep alive probes that are sent before assuming that the connection has been terminated.	0x0A (10 retries)

(Ex. 1021 (Minami), 67:1–30 (excerpted with emphasis added).)

When a server, that has activated the keepalive feature, does not receive a data packet for a period of time called the "keepidle time," it will send a keepalive probe to the mobile communication unit to assess continued activity. Upon receiving a keepalive message, the mobile communication unit, if active, returns an acknowledgment packet. The server is configured to end its connection with the mobile communication

> unit when none of the packets in the keepalive probe are acknowledged. Of course the mobile communication unit will not realize the server ended this connection until the mobile communication unit next attempts to communicate with the server.

(Ex. 1012 (Harsch), 3:5–16 (excerpted with emphasis added).)

> **If the mobile communication unit acknowledges one of the packets in the probe, the server will determine that the mobile communication unit is still active and reset its keepidle timer, thus maintaining the current connection**. However, if none of the packets in the keepalive burst are acknowledged, the server will terminate the connection. This typically causes an application program running on the server to end its session with the mobile communication unit.

(*Id.*, 3:17–25 (excerpted with emphasis added).)

43. A person of ordinary skill in the art would have known at the time that the keep alive time used in such configurations would generally be set to a time period that is less than the timeout period for the connection in order to maintain the connection. (*E.g.*, Ex. 1014 (Rosenberg), 4:38–43 (excerpted below); Ex. 1013 (Hayden), 5:50–63 (excerpted below).) For example, consistent with the example provided by Hayden, it was known that a node may send a keep-alive message at a rate of T/2, where T is the timeout period for connection. (Ex. 1013 (Hayden), 5:50–63 (excerpted below).) If no response is received before the timeout period T

expires, then the connection is broken in a standard connection. (Ex. 1012 (Harsch), 3:17–25 (excerpted above).)

> … **Periodically, the client 30 sends a keep-alive message to the server 20. For example, the client 30 sends a keep-alive message at rate T/2, where T is the server timeout period (described below).** The keep-alive message informs the server **20** that the client **30** is still reading the video data stream. As long as a client **30** is reading the data, the server **20** will continue to transmit the video data stream on the transmission address A_n. …

(Ex. 1013 (Hayden), 5:50–63 (excerpted with emphasis added).) Rosenberg (Ex. 1014) describes similar features consistent with what a person of ordinary skill in the art would have understood about keep-alive mechanisms at the time:

> According to one aspect of the invention, it is realized that the lowest timeout value on commercially-available firewalls is 30 seconds. Therefore, **periodic keep-alive messages are transmitted by the host to the directory server with a period shorter than the timeout to maintain the connection established through the firewall**.

(Ex. 1014 (Rosenberg), 4:38–43 (excerpted with emphasis added).)

4. Communication Protocols

44. As I explained above, a communication protocol was known as a system of rules that allow two or more nodes of a communications system to transmit

information. The protocol defines the rules, syntax, semantics, and synchronization of communication. Some examples of communication protocols known prior to and at the time of the alleged invention include the Internet Protocol (IP), Transmission Control Program (TCP), User Datagram Protocol (UDP), Internetwork Packet Exchange (IPX), Sequenced Packet Exchange (SPX), and Transmission Control Protocol (TCP). Minami (Ex. 1021) and Bo (Ex. 1024) describe such features consistent with the knowledge of a person of ordinary skill in the art at the time:

> **Communications networks use protocols to transmit and receive data**. Typically, a communications network comprises a collection of network devices, also called nodes, such as computers, printers, storage devices, and other computer peripherals, communicatively connected together. Data is transferred between each of these network devices using data packets that are transmitted through the communications network using a protocol. **Many different protocols are in current use today. Examples of popular protocols include the Internet Protocol (IP), Internetwork Packet Exchange (IPX) protocol, Sequenced Packet Exchange (SPX) protocol, Transmission Control Protocol (TCP),** Point-to-Point Protocol (PPP) and other similar new protocols that are under development. **A network device contains a combination of hardware and software that processes protocols and data packets.**

(Ex. 1021 (Minami), 2:23–38 (excerpted with emphasis added).)

> **Now quite a few protocols have been designed and standardized for communications between clients and streaming servers.** Obviously, IP serves as the network-layer protocol for the Internet video streaming. **Since TCP's retransmission feature always introduces delays that are not acceptable for video streaming applications with stringent delay requirements, UDP is now typically employed as the transport protocol for video streaming.** In addition, RTP/RTCP (real-time transport protocol/real-time control protocol) is designed to provide end-to-end transport functions on top of UDP for supporting real-time applications. Moreover, RTSP (real-time streaming protocol) defines the messages and procedures to control the delivery of the multimedia data during an established session.

(Ex. 1024 (Bo) ¶[0010] (excerpted with emphasis added).)

45. It was known at the time that computer systems may use more than a single protocol to handle a network transmission between nodes. For instance, it was known to handle network transmission using a set of cooperating protocols, sometimes called a protocol suite. A person of ordinary skill in the art would have been aware of known protocol suites, such as the TCP/IP, IPX/SPX, and the UDP/IP protocol suites. (Ex. 1012 (Harsch), 1:53–59; Ex. 1021 (Minami), 1:20–33, 2:23–3; Ex. 1015 (RFC 1131); Ex. 1016 (XTP).)

a) TCP/IP

46. TCP/IP stands for Transmission Control Protocol/Internet Protocol. TCP/IP was known as the main protocol used for Internet communications (e.g., Web pages were (and still are) typically served using TCP/IP). (E.g., Ex. 1012 (Harsch), 2:33–35.) The TCP/IP was standardized in a series of publicly available Request for Comments (RFCs) published by the Internet Engineering Task Force, including RFC 793, entitled "Transmission Control Protocol," published in 1981. (E.g., Ex. 1008 (RFC 793).) TCP/IP consists of two parts: (1) Transmission Control Protocol (TCP), which provides a virtual bi-directional end-to-end connection that provides guaranteed in-order, error-free delivery of arbitrary amounts of data between programs running on different computers over the internet (Ex. 1012 (Harsch), 2:35–39; Ex. 1023 (Rideout) ¶[0109]; Ex. 1009 (RFC 1122), 82–94 (discussing TCP).); and (2) Internet Protocol (IP), which provides delivery of datagrams (IP packets) to any routable Internet address, without any reliability or ordering guarantees (Ex. 1026 (Diab), 2:43–52.) Rideout (Ex. 1023) provides a description of TCP consistent with that known to those of ordinary skill in the art at the time:

> TCP stands for Transmission Control Protocol. It is a basic communication language or protocol of the internet. It is a set of rules (protocol) used along with the IP (Internet Protocol) to send data in the form of message units between computers over the internet. In a combination of TCP/IP, TCP takes care of keeping

> track of the individual units of data (packets) and IP handles the actual delivery of the data. It also works for re-transmission mechanisms, tolerates against the delay of data, multiplicative decrease in case of congestion, and sharp variation in visual effect.

(Ex. 1023 (Rideout) ¶[0109] (excerpted).)

a. Three-way Handshake

47. It was known at the time that to establish a TCP connection the nodes must first initiate a three-way handshake (to negotiate parameters for the connection) before any data can be sent between the nodes. The three-way handshake was known as the necessary procedure used to establish a TCP connection. The RFC 793 provides descriptions of such features known to those of ordinary skill in the art at the time. (*See, e.g.*, Ex. 1008 (RFC 793), 31–32, 34–37.) In particular, the three-way handshake was known to involve transmitting three messages to negotiate and start a TCP-based session between the two nodes, as RFC 793 illustrates below:

> 1) A --> B SYN my sequence number is X
>
> 2) A <-- B ACK your sequence number is X
>
> 3) A <-- B SYN my sequence number is Y
>
> 4) A --> B ACK your sequence number is Y

(Ex. 1008 (RFC 793), 31–32.) Because steps 2 and 3 can be combined in a single message, this is called the three-way (or three messages) handshake. At the first step

in the three-way handshake, the node (A) that wants to set up a session sends a synchronization (SYN) packet to the node (B) with which it intends to communicate. (*Id.*) This first packet includes an indication that a new connection is wanted and a sequence number used to identify the first packet. (*Id.*) The receiving node (B) responds to the second node (A) with an acknowledgment (ACK) packet that includes another SYN packet. (*Id.*) This ACK/SYN packet lets the first node (A) know that the second node (B) has recorded (synchronized) the sequence number that was in the first packet. (*Id.*) Finally, the first node responds to the second node with an ACK packet, indicating that the first node (A) has recorded the second node's sequence number. (*Id.*) This is summarized in RFC 793:

> **For a connection to be established or initialized, the two TCPs must synchronize on each other's initial sequence numbers**. This is done in an exchange of connection establishing segments carrying a control bit called "SYN" (for synchronize) and the initial sequence numbers. As a shorthand, segments carrying the SYN bit are also called "SYNs". Hence, the solution requires a suitable mechanism for picking an initial sequence number and a slightly involved handshake to exchange the ISN's.

(*Id.*, 31.)

> **A three way handshake is necessary** because sequence numbers are not tied to a global clock in the network, and TCPs may have different mechanisms for picking the ISN's. The

> receiver of the first SYN has no way of knowing whether the segment was an old delayed one or not, unless it remembers the last sequence number used on the connection (which is not always possible), and so it must ask the sender to verify this SYN. The three way handshake and the advantages of a clock-driven scheme are discussed in [3].

(*Id.*, 32 (emphasis added).)

> **The "three-way handshake" is the procedure used to establish a connection.** This procedure normally is initiated by one TCP and responded to by another TCP. The procedure also works if two TCP simultaneously initiate the procedure. When simultaneous attempt occurs, each TCP receives a “SYN” segment which carries no acknowledgment after it has sent a “SYN”. Of course, the arrival of an old duplicate "SYN" segment can potentially make it appear, to the recipient, that a simultaneous connection initiation is in progress. Proper use of "reset" segments can disambiguate these cases.

(*Id.*, 34 (emphasis added).)

> The synchronization requires each side to send its own initial sequence number and to receive a confirmation of it in acknowledgment from the other side. Each side must also receive

> the other side's initial sequence number and send a confirming acknowledgment.
>
> 1) A --> B SYN my sequence number is X
>
> 2) A <-- B ACK your sequence number is X
>
> 3) A <-- B SYN my sequence number is Y
>
> 4) A --> B ACK your sequence number is Y
>
> *Because steps 2 and 3 can be combined in a single message this is called the three way (or three message) handshake*.

(*Id.*, 31–32 (emphasis added).)

48. Those of ordinary skill in the art would have known that once a connection is established using the three-way handshake, data packets can then be exchanged between the two nodes. Each data packet in an exchange is also acknowledged by a response packet containing an ACK flag. (*Id.*, 31–32 (emphasis added).)

b. Retransmission Timeout

49. TCP was also known to include a retransmission mechanism that handles lost transmissions (e.g., unacknowledged packets). (*See, e.g.*, Ex. 1023 (Rideout) ¶[0109] (excerpted above).) For example, this mechanism was known to be implemented by a retransmission timer at the node. Upon the expiration of the retransmission timer, a node will resend the unacknowledged transmitted packet. One of ordinary skill in the art would have known that an exemplary scenario when

this occurs is where a first node transmits a packet of data to a second node, and the packet is lost before it reaches the second node. The first node will, upon the expiration of retransmission timer ("retransmission timeout"), resend the packet to the second node. RFC 793 describes this feature:

> RETRANSMISSION TIMEOUT
>
> **For any state if the retransmission timeout expires on a segment in the retransmission queu**e, **send the segment at the front of the retransmission queue again**, reinitialize the retransmission timer, and return.

(*E.g*., Ex. 1008 (RFC 793), 77 (emphasis added).)

c. User Timeout

50. TCP was also known to include a "user timeout" function for terminating an idle connection. For example, one of ordinary skill in the art would have known that a TCP connection is aborted when a sending node does not receive an acknowledgment (ACK) packet within a length of time defined by the user timeout for the connection following the node's transmission of a data packet. RFC 793 also describes these features:

> Abort
>
> Format: ABORT (local connection name)
>
> **This command causes all pending SENDs and RECEIVES to be aborted, the TCB to be removed, and a special RESET**

> **message to be sent to the TCP on the other side of the connection.** Depending on the implementation, users may receive abort indications for each outstanding SEND or RECEIVE, or may simply receive an ABORT-acknowledgment.

(*Id.*, 50 (excerpted with emphasis added).)

> USER TIMEOUT
>
> **For any state if the user timeout expires, flush all queues, signal the user "error: connection aborted due to user timeout"** in general and for any outstanding calls, delete the TCB, enter the CLOSED state and return.

(*Id.*, 77 (excerpted with emphasis added).)

51. It was also known by those of ordinary skill in the art that the user timeout period for this "user timeout" function was either defined (or calculated) by the operator or application running on the node. (*E.g.*, *id.*, 37–39, 77 (reproduced above)).

d. TCP Options

52. TCP was also known to be extendable. For example, one of ordinary skill in the art would have been familiar with the TCP header "options" field. RFC 793 shows the options field in its Figure 3:

TCP Header Format

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Source Port          |       Destination Port        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Sequence Number                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Acknowledgment Number                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Data |           |U|A|P|R|S|F|                               |
| Offset| Reserved  |R|C|S|S|Y|I|            Window             |
|       |           |G|K|H|T|N|N|                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Checksum            |         Urgent Pointer        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             data                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

TCP Header Format

Note that one tick mark represents one bit position.

Figure 3.

(Ex. 1008 (RFC), 15 (Figure 3).)

53. The options field was known to include zero or more variable length options. (Ex. 1008 (RFC), 17–19.) In general, it was known that the TCP options field consisted of three sub-fields, as RFC 793 shows:

Maximum Segment Size

```
+--------+--------+---------+--------+
|00000010|00000100|   max seg size   |
+--------+--------+---------+--------+
 Kind=2   Length=4
```

(*Id.*, 18.)

54. One of ordinary skill in the art would have been aware and appreciated that the *Kind* sub-field is used to identify the kind of option. (*Id.*, 18–19.) This sub-field was known to have been defined in accordance with an identification number assigned by the Internet Assigned Numbers Authority (IANA). The *Length* sub-field was known to be used to specify the total length of the options field, including the Kind and Length sub-fields. The actual option-data was known to contain the actual data associated with the option. (*Id.*) It was also known that the meaning of the data depends on the Kind value. (*Id.*, 18.) For example, when the Kind value is 2, the data value represents the maximum segment (packet) size for the TCP connection. (*Id.*)

b) IPX/SPX

55. IPX/SPX stands for Internetwork Packet Exchange/Sequenced Packet Exchange. IPX was known as a network layer protocol (layer 3 of the OSI Model), while SPX was known as a transport layer protocol (layer 4 of the OSI Model). (E.g., Ex. 1021 (Minami), 1:20–32 (reproduced above), 2:23–37 (reproduced above); Ex. 1007 (Abdolbaghian), 2:36–48 (excerpted below).) The SPX layer sits on top of the IPX layer and provides connection-oriented services between two nodes on the network. SPX is used primarily by client-server applications. Those of ordinary skill in the art would have known at the time that IPX and SPX both provided

connection services similar to TCP/IP, with the IPX protocol having similarities to IP, and SPX having similarities to TCP. Abdolbaghian (Ex. 1007) describes such features consistent with that known by one of ordinary skill in the art at the time:

> … **Information communicated over the communication network 4 may conform to any data communications protocol, including TCP/IP, IPX/SPX**, NetBios and AppleTalk. The communication network 4 may include wire line (Such as twisted-pair telephone wire, coaxial cable, electric power line, optical fiber wire, leased line or the like or wireless (Such as Satellite, cellular, radio frequency or the like) connections.

(Ex. 1007 (Abdolbaghian), 2:36–48 (excerpted with emphasis added).)

c) UDP/IP

56. UDP/IP stands for User Datagram Protocol /Internet Protocol. (Ex. 1023 (Rideout) ¶[0110].) UDP was known as an alternative to the Transmission Control Protocol (TCP). (*Id.*; Ex. 1009 (RFC 1122), 77–80 (discussing UDP)) One of ordinary skill in the art would have understood that UDP was a lightweight transport protocol as compared to TCP. UDP was known to be built on top of the IP and provided extra performance from IP by not implicating some of the features that make IP a heavy-weight protocol. (Ex. 1023 (Rideout) ¶[0110].) For example, UDP was known as a connectionless protocol that offered a limited amount of services, such as no reliability or ordering guarantees, when messages are exchanged

between computers in a network. Rideout (Ex. 1023) describes such features consistent with that known by one of ordinary skill in the art at the time:

> UDP stands for User Datagram Protocol. It is a communication protocol that offers a limited amount of service when messages are exchanged between computers in a network that uses the Internet Protocol (IP). UDP is an alternative to the Transmission Control Protocol (TCP) and, together with IP, is sometimes called UDP/IP. It is a lightweight transport protocol as compared to TCP, it is built on top of the IP, and squeezes extra performance from IP by not implicating some of the features that make IP a heavy-weight protocol. It is used where delivery is not guaranteed. It provides connectionless service that is not possible in TCP.

(*Id.* ¶[0110] (excerpted).)

d) RUDP

57. RUDP stands for Reliable User Datagram Protocol. RUDP was known as a simple packet-based transport protocol. RUDP is layered on the UDP/IP Protocols and provides reliable in-order delivery for virtual connections. One of ordinary skill in the art would have understood and appreciated that the flexibility of

the RUDP made it suitable for a variety of transport uses. RUDP was described in an IETF internet-draft ("RELIABLE UDP PROTOCOL") in 1999 (Ex. 1027).[1]

> **RUDP is a simple packet based transport protocol**. RUDP is based on RFCs 1151 and 908 - Reliable Data Protocol. RUDP is layered on the UDP/IP Protocols and provides reliable in-order delivery (up to a maximum number of retransmissions) for virtual connections. RUDP has a very flexible design that would make it suitable for a variety of transport uses. One such use would be to transport telecommunication signaling protocols.

(Ex. 1027, 1 (emphasis added).)

> This Internet Draft discusses **a simple packet based transport protocol** designed to support applications that require a reliable, sequenced packet based transport protocol. RUDP is based on RFCs 1151 and 908 - Reliable Data Protocol. RUDP is layered on the UDP/IP Protocols and provides reliable in-order delivery (up to a maximum number of retransmissions) for virtual connections.

(*Id.*, 2 (emphasis added.))

58. RUDP was designed to allow the characteristics of each connection to be individually configured so that many protocols with different transport

[1] *Berg* (Ex. 1028) also includes the RUDP specification (Ex. 1027) in its Appendix 1. (*See, e.g.*, Ex. 1028 (Berg) at 8:11–18, 16:58–25:54.)

requirements can be implemented simultaneously on the same platform. (*Id.*, 3.) For example, to ensure quality, RUDP extends UDP by employing:

- Acknowledgment of received packets (*Id.*, 4, 9);
- Keep-alive feature (*Id.*, 10–11);
- Retransmission of lost packets (*Id.*, 12);

59. RUDP accomplishes this by adding a header to the UDP packet. (*Id.*, 3.) For example, the RUDP header is reproduced from the 1999 IETF internet-draft ("RELIABLE UDP PROTOCOL") (Ex. 1027) below:

```
 0 1 2 3 4 5 6 7 8             15
+-+-+-+-+-+-+-+-+---------------+
|S|A|E|R|N|C|T| |    Header     |
|Y|C|A|S|U|H|C|0|    Length     |
|N|K|K|T|L|K|S| |               |
+-+-+-+-+-+-+-+-+---------------+
|  Sequence #   +   Ack Number  |
+---------------+---------------+
|             Checksum          |
+---------------+---------------+
```

Figure 1, RUDP Header

(*Id.*, 3 (Figure 1).)

60. One of ordinary skill in the art would have known at the time that to establish an RUDP connection, the nodes negotiate parameters for the connection

before any data can be sent between them. (*Id.*, 16 (reproduced below).) The feature negotiation was known to involve transmitting three messages to negotiate and start an RUDP-based connection between the two nodes, as explained in the 1999 IETF internet-draft ("RELIABLE UDP PROTOCOL") (Ex. 1027):

> **When client initiates a connection its sends a SYN segment which contains the negotiable parameters** defined by the ULP via the API. **The server can accept these parameters by echoing them back in its SYN with ACK response** or propose different parameters in its SYN with ACK response. **The client can then choose to accept the parameters sent by the server by sending an ACK to establish the connection** or it can refuse the connection by sending send a RST. Features cannot be re-negotiated during an auto reset.

(*Id.*, 16 (emphasis added).)

61. The SYN Segment was known to be used to establish a connection and synchronize sequence numbers between two hosts. The SYN segment was also known to be used to perform an auto-reset on a connection. The SYN segment contains the negotiable parameters of the connection. (*Id.*, 4–5.) One of ordinary skill in the art would have known that all configurable parameters that the peer must know about are contained in this segment. The SYN segment includes option flags

that indicate the features of the connection being established. The ACK segment is used to acknowledge in-sequence segments. (*Id.*, 9.)

B. Hypertext Markup Language (HTML)

62. A markup language was known prior to and at the time of the alleged invention as a computer programming language. For example, one of ordinary skill in the art would have understood that the markup language includes instructions that define the content's layout on the page using a predefined set of symbols. (*See, e.g.*, Ex. 1020 (De Boor) ¶[0022].) One common markup language known by those of ordinary skill in the art is Hypertext Markup Language (HTML). HTML was known as the standard markup language for documents designed to be displayed in a web browser.

> **A markup language is a computer programming language** that allows the content of a page or a screen display to be defined by the inclusion of predefined symbols in the content itself **indicating** the logical components of the content, **instructions for the layout of the content on the page or scree**n, or other data which can be interpreted by some automatic system responsible for displaying, manipulating or modifying the content.

(*Id.* ¶[0022] (excerpted with emphasis added).)

63. One of ordinary skill in the art would have understood that in order to display a webpage, the browser must first fetch the page from a web server. (*E.g.*,

id. ¶[0024] (excerpted below).) The browser does this by requesting the webpage from a server. For example, a browser's request to a server usually included a page address, such as http://www.domain.com/webpage.htm.

> **The browser can also modelessly fetch markup language pages or other content that is stored remotely, by accessing such pages via a telecommunications network such as the World Wide Web, and likewise decode and display these remotely accessed pages.** When a user interface page is displayed, user selection of a control function passes a URL or command data to the browser. The browser effects a telecommunication function in response the received URL or command.

(*Id.* ¶[0024] (excerpted with emphasis added).)

C. Virtual Network Computing (VNC)

64. Those of ordinary skill in the art would have also known that another useful application in a network environment was remote access. A remote access service was known to connect a client to a host computer, known as a remote access server. The most common approach to this service that was known to those of ordinary skill in the art at the time was the remote control of a computer using another device over a network connection. For example, Microsoft's Remote Desktop was known to allow a client to send keystrokes and events to a host (server) computer. (Ex. 1022 (Raghunath) ¶[0016].) The host computer would then send,

for example, graphics over to the remote (client) computer for display. (*Id.*) As another example, in a virtual network computing (VNC) system, a server machine was known to supply an entire desktop environment that can be accessed from a client machine over a network. The underlying technology was known as a simple remote display protocol, which was independent of the operating system used. (*Id.* ¶[0019].)

> **The Microsoft Remote Desktop provides a solution that just sends the key strokes and events to the host (server) computer. The host computer then sends the graphics over to the remote (client) computer.** This is similar to what X11 did for workstations in the 80s and 90s. The session is not resumed where it was left off. This is just a client server approach.

(*Id.* ¶[0016] (excerpted with emphasis added).)

> In the virtual network computing (VNC) system, server machines supply not only applications and data but also an entire desktop environment that can be accessed from any Internet-connected machine using a simple software NC. **Whenever and wherever a VNC desktop is accessed, its state and configuration (right down to the position of the cursor) are exactly the same as when it was last accessed. The technology underlying VNC is a simple remote display protocol.** Unlike other remote display protocols such as the X Window System and Citrix's ICA, the VNC protocol is totally independent of operating system, windowing system, and applications The VNC

> system is freely available for download from the ORL Web site at http://www.orl.co.uk/vnc/. It does not require the user to carry any hardware. However, it assumes network connectivity. See T. Richardson, Q. Stafford-Fraser, K. R. Wood, and A. Hopper, "Virtual Network Computing", IEEE Internet Computing, Vol. 2 No.1, January/February 1998 pp 33–38.

(*Id*. ¶[0019] (excerpted with emphasis added).)

VI. THE '217 PATENT

65. The '217 patent is directed to networking, and in particular, to the sharing of information for detecting an idle TCP connection. (*See, e.g.*, Ex. 1001, 2:48-50, 5:50-59.) Figure 7 below depicts a method for sharing this information.

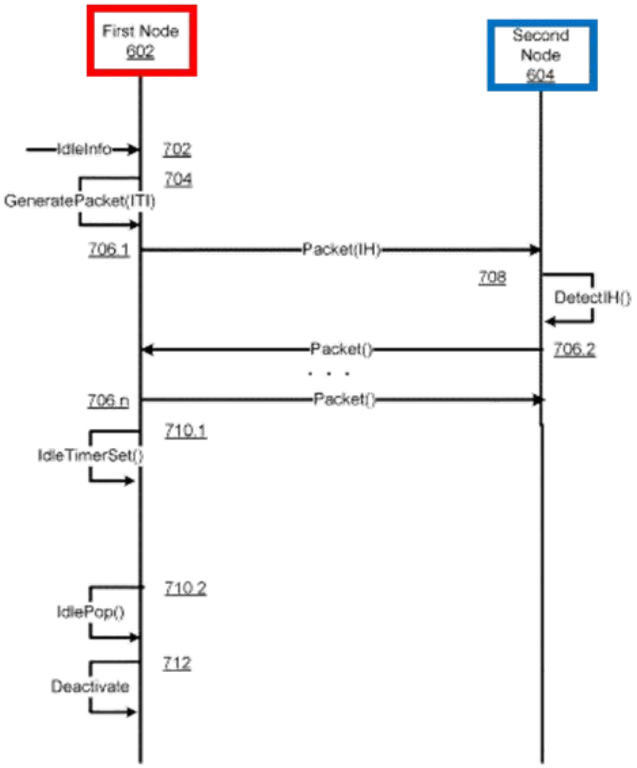


Fig. 7

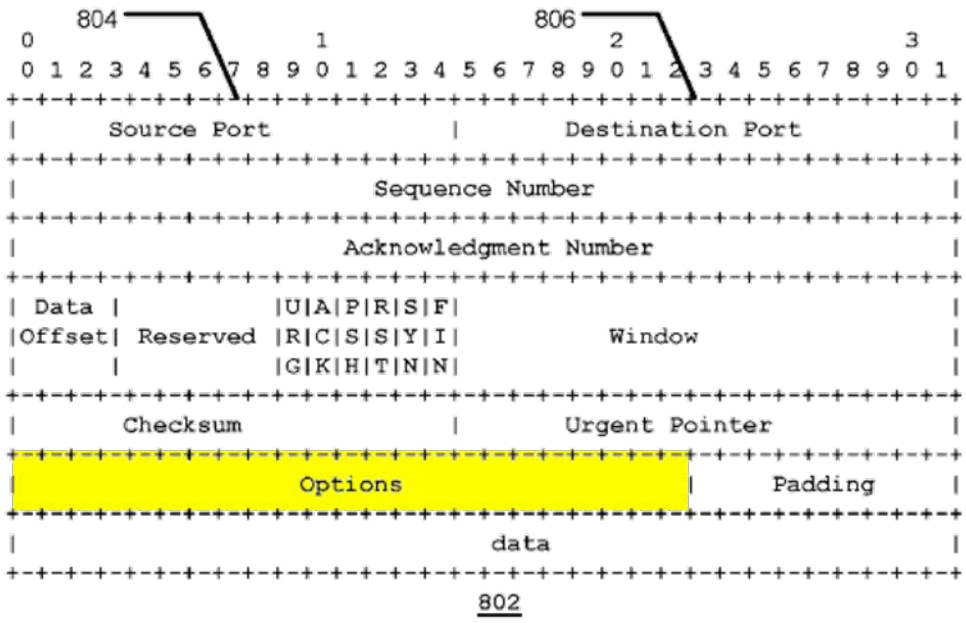
(*Id.*, FIG. 7 (annotated).)

66. As shown in Figure 7, the **first node 602** receives a message 702 that identifies idle information representing a duration for an idle time period ("ITP") from an ITP policy component. (*Id.*, 9:9-17.) Message 702 may take various forms,

and thus can be "a message received via a network." (*Id.*) The idle information "may include and/or identify a duration of time for detecting an idle time period." (*Id.*, 9:57-61.) The duration "may be specified according to various measures of time[,] including seconds, minutes, hours, and/or days." (*Id.*)

67. Next, the **first node 602** "generat[es] a TCP packet including an ITP header based on received idle information" and sends the TCP packet to the **second node 604**. (*Id.*, 13:24-27.) The TCP options field of a TCP packet may store this ITP header. (*Id.*, 12:66-13:7, FIG. 8) Figure 8 below, which is "adapted from RFC 793," illustrates the TCP options field in a TCP packet that is used to convey the idle information.[2] (*Id.*, FIG. 8, 3:49-51, 12:66-13:7.)

[2] RFC 793 is titled "Transmission Control Protocol, DARPA Internet Program Internet Protocol Specification." (Ex. 1008, 1; Ex. 1001, 2:12-19.) RFC 793 dates back to 1981 (Ex. 1008, 1). A person of ordinary skill in the art would have understood that RFC 793 defined TCP at that time. (*See* my discussion below in Section X.A.2 regarding RFC 793.)



(*Id.*, FIG. 8 (cropped and annotated); *see also* Ex. 1008, FIG. 3.)

68. As also shown in Figure 8 below, the options field contains a KIND sub-field that identifies the type of option being presented, a length sub-field that specifies the length of the option field, and an ITP Data sub-field 826 that contains data. (Ex. 1001, FIG. 8.) Thus, the TCP options field is used to carry ITP information. (*Id.*, 12:66-13:7.)

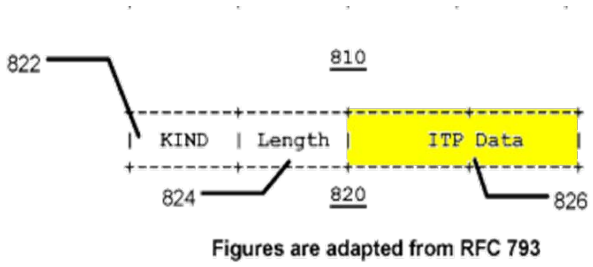


Fig. 8

(*Id.*, FIG. 8 (annotated).)

69. The ITP header ("IH") is exchanged between nodes "during setup of [a] TCP connection" by way of a "three-way handshake," which is a procedure "for establishing a TCP connection" (*Id.*, 12:4-18.) For example, as shown in Figure 7 above, the **first node 602** transmits a message 706.1, which is a TCP packet including an (IH) that contains ITP information, to the **second node 604**. (*Id.*, 13:49-51.) Message 708 exemplifies the detection of the ITP header in the TCP packet received from the **first node 602** by the **second node 604**, i.e., the **second node 604** detects the ITP information identifying a duration for the ITP that is stored in the IH. (*Id.*, 19:7-20, 17:36-18:3.) In other words, the **second node 604** detects the ITP information identifying a duration for the ITP that is stored in the IH—this detected ITP information is used later to detect an idle period in the connection. (*Id.*.)

70. The '217 patent explains that "a TCP keep-alive option, a TCP user timeout, a retransmission timeout, an acknowledgment timeout, and/or another timeout associated with a TCP connection may be modified based on the first idle information." (*Id.*, 11:1-4.)

71. The challenged claims, on the other hand, include limitations directed to a first connection between two nodes, where the first connection encompasses non-TCP connections. (EX1001, claim 1, from which the challenged claims

depend).[3] The challenged claims also recite detecting a first time period during at least a portion of the first connection "**including at least a portion of the first connection set up**" (where the detection of the time period causes the connection to at least partially close). (*Id.*, claim 1.)

72. However, as I have discussed above and further discussed below, the '217 patent only describes its features in terms of a TCP connection and detecting time periods **after** a TCP connection is established. (*See* my discussions and explanations below in Section X; *see also generally* Ex. 1001, 3:55-21:27 (Detailed Description section discussing aspects in terms of a "TCP" packet and "TCP" connection)[4].)

73. Based on my review, neither the '217 patent nor its priority applications describe a non-TCP connection in the context of the claimed invention, let alone

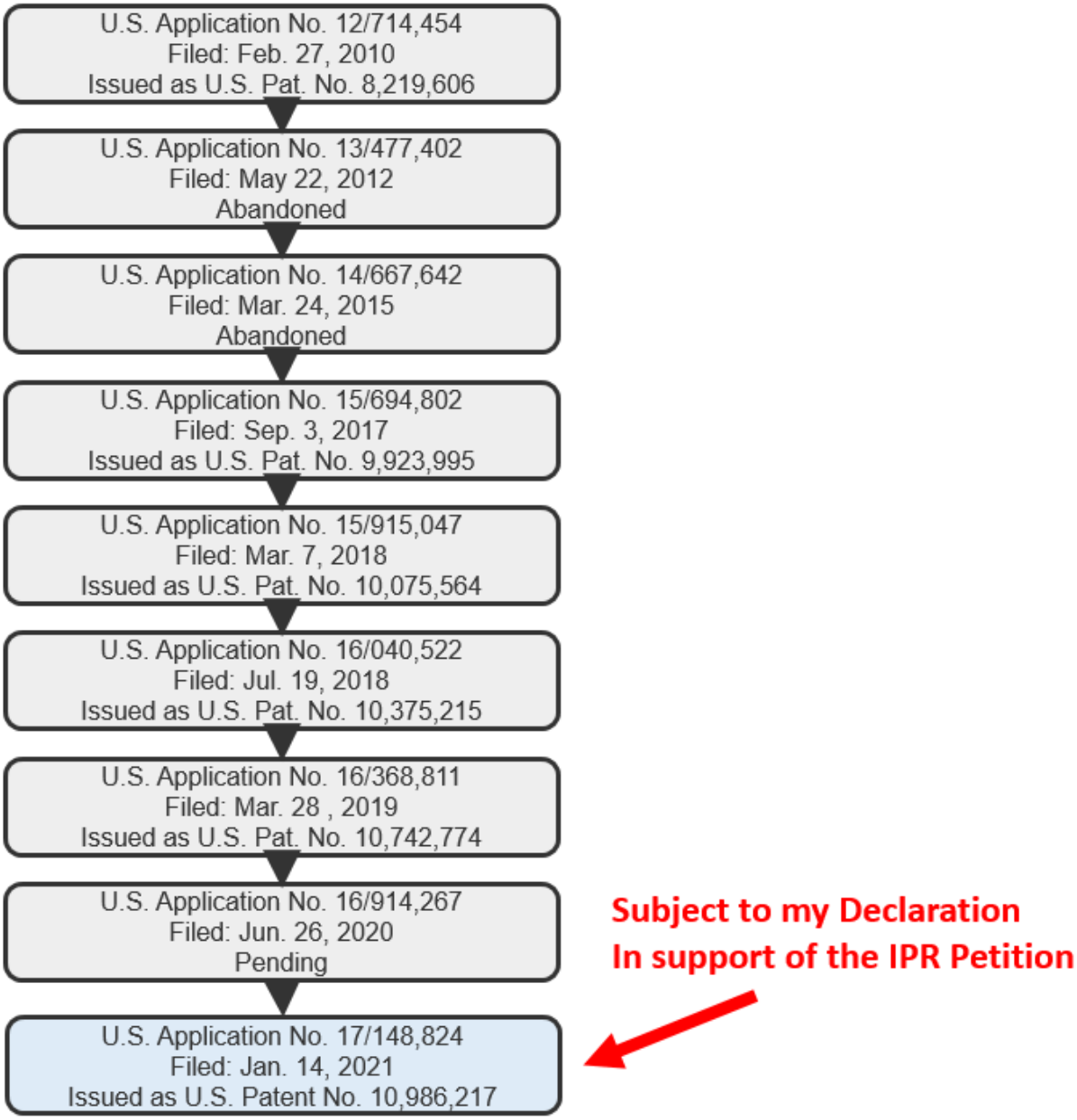
[3] I sometimes add emphases to text of a document that I quote (as I do here) in this declaration. All such emphasis is added unless I state or explain otherwise.

[4] As I discuss further below, the specification does mention a "variant of the current TCP," but does so in terms of the location and structure of an ITP header that "may be included in a TCP packet." (Ex. 1001, 13:11-13; *see* my discussions and explanations below in Section X.A.)

describe the detection of a time period **during the time such a connection is being setup**, as recited in claim 1. (*See* my discussion below in Section X.)

74. I understand that the ’217 patent issued from U.S. Patent Application No. 17/148,824 (“the ’824 application”) (Ex. 1004), which was filed on January 14, 2021. (Ex. 1001 at Cover.) I also understand that the ’824 application is a continuation of U.S. patent application no. 16/914,267 (“the ’267 application”). I also understand that the ’267 application is a continuation of U.S. patent application no. 16/368,811 (“the ’811 application”), which issued as U.S. Patent No. 10,742,774 (“the ’774 patent”). I understand that the ’811 application is a continuation of U.S. patent application no. 16/040,522 (“the ’522 application”), which issued as U.S. Patent No. 10,375,215 (“the ’215 patent”). I also understand that the ’522 application is a continuation of U.S. patent application no. 15/915,047 (“the ’047 application”), which issued as U.S. Pat. No. 10,075,564 (“the ’564 patent”). I understand that the ’047 application is a continuation of U.S. patent application no. 15/694,802 (“the ’802 application”), which issued as U.S. Pat. No. 9,923,995 (“the ’995 patent”). I also understand that the ’802 application is a continuation-in-part of U.S. patent application no. 14/667,642 (“the ’642 application”). I understand that the ’642 application is a continuation-in-part of U.S. patent application no. 13/477,402 (“the ’402 application”). I also understand that the ’402 application is a continuation of U.S. patent application no. 12/714,454 (“the ’454 application”),

which issued as U.S. Patent No. 8,219,606 ("the '606 patent"). I have created the demonstrative below to help visualize the purported priority chain for the '217 patent.



VII. CLAIM CONSTRUCTION

75. I have given the terms in the challenged claim their plain and ordinary meanings, as would have been understood by a person of ordinary skill in the art at the time of the alleged invention, taking into consideration the language of the claims, the specification, and the prosecution history of the ’217 patent. I have applied these understandings of the claim terms in my analysis and in forming my opinions where appropriate in this Declaration.

VIII. OVERVIEW OF THE PRIOR ART[5]

A. Overview of Morris (Ex. 1005)

76. Morris is directed to "[m]ethods and systems … for sharing information for detecting an idle TCP [transmission control protocol] connection." (EX1005, ¶[0008]; *see also id.*, ¶¶[0005] ("The broader issue of enabling cooperation and negotiation between nodes in a TCP connection in detecting and managing idle, underactive, and/or dead TCP connections remains unaddressed."), [0006], [0009]-[0011].)

77. Morris discloses a hardware device 100 ("apparatus") in Figure 1, exemplified by first node 602 or second node 604 in Figures 6-7, for carrying out its disclosed method(s). (EX1005, ¶¶[0021]-[0023], [0040], [0126]-[0127], claims 11, 22; FIGS. 1, 6-7.)

> An exemplary device included in an execution environment that may be configured according to the subject matter is illustrated in FIG. 1.

[5] These summaries of the prior art in Section VIII are applicable to my analysis below in Section IX. Accordingly, I incorporate these summaries, which reflect how a person of ordinary skill in the art would have understood the references, into my analysis below where appropriate (*see, e.g.*, Section IX below).

(EX1005, ¶[0021].)

> FIG. 1 illustrates hardware device 100 included in execution environment 102 which includes instruction-processing unit (IPU) 104, such as one or more microprocessors; physical IPU memory 106 including storage locations identified by addresses in a physical memory address space of IPU 104 …

(EX1005, ¶[0023].)

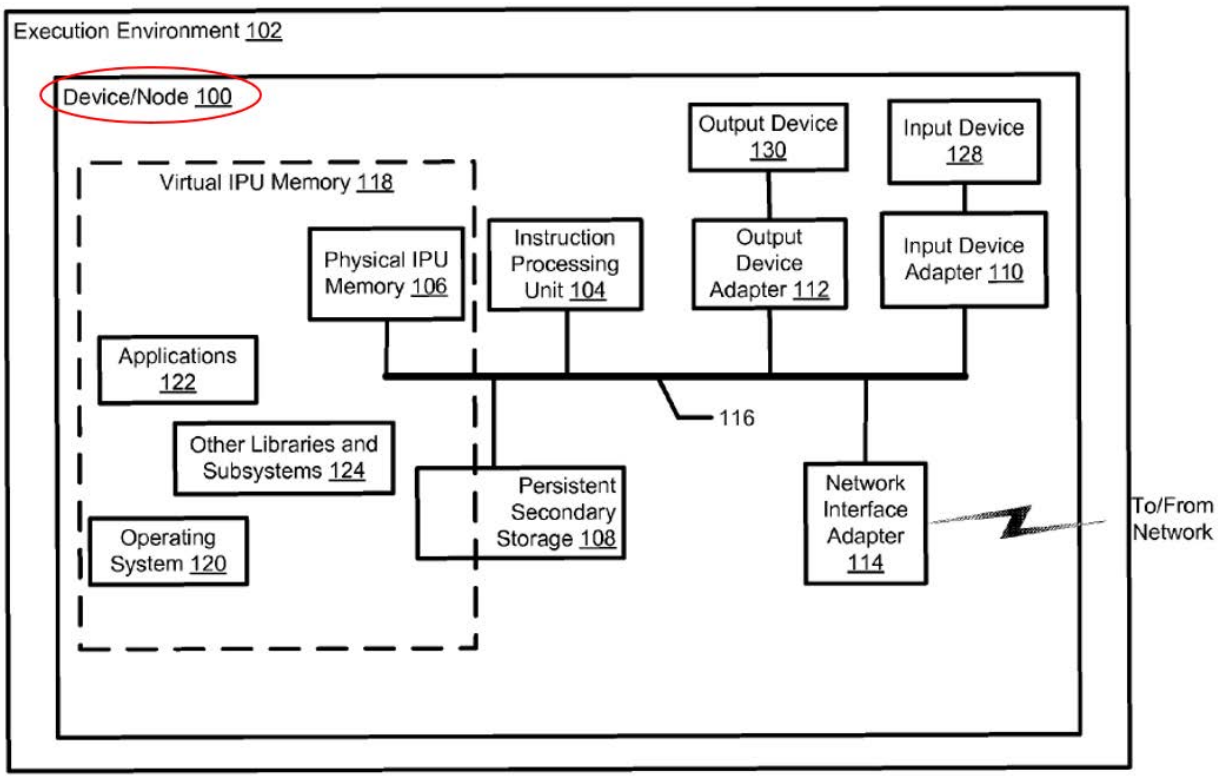


Fig. 1

(EX1005, FIG. 1 (annotated).)[6]

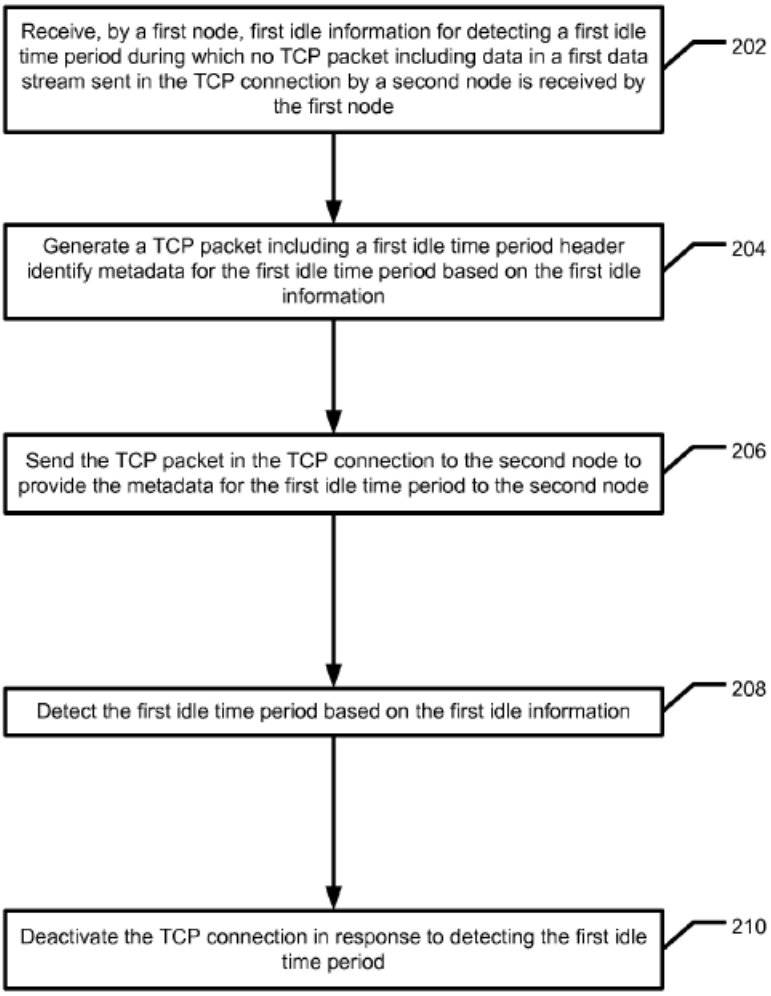


Fig. 2

(EX1005, FIG. 7.)

[6] Morris also refers to the device 100 as node 100, as shown in Figure 1.

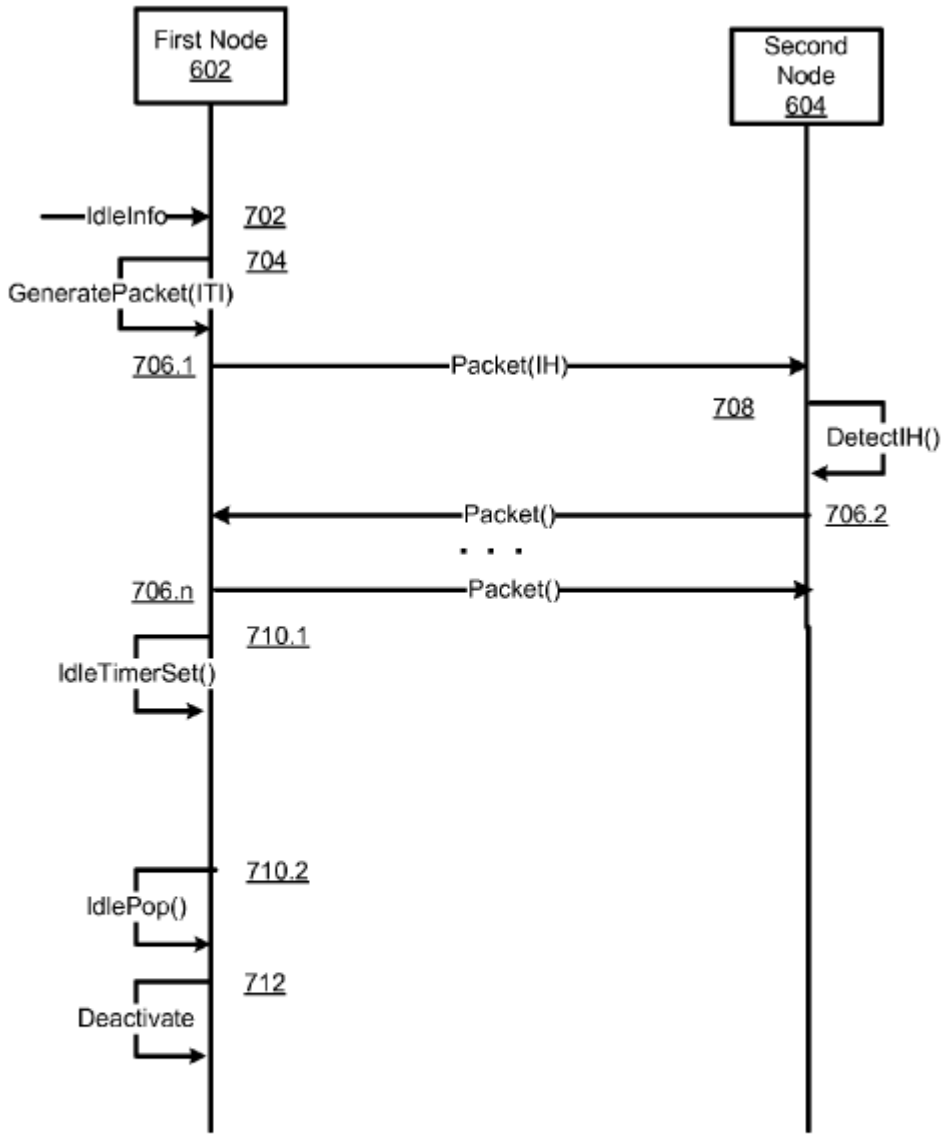


Fig. 7

(EX1005, FIG. 7.)

78. Morris discloses negotiating the value of an idle time period (ITP) during connection setup between two nodes (e.g., first node 602 and second node

604 in Figures 2 and 7 above) and modifying timeout attributes for the established connection based on the negotiated ITP to detect timeouts. (*See, e.g.*, EX1005, FIGS. 1-7, ¶¶[0035]-[0040], [0052]-[0059], [0069], [0082]-[0083], [0088], [0096], [0104], [0106]-[0109].)

B. Overview of Session Initiation Protocol (SIP) (Ex. 1006)

79. SIP is directed to a protocol in the networking field for enabling communication between two nodes. For example, SIP discloses the Session Initiation Protocol, which is an application-layer protocol for creating, modifying, and terminating sessions (between nodes) such as telephone calls, multimedia distribution, or multimedia conferences. (EX1006, Abstract; *see also id.* at 8 (describing that "a session is considered an exchange of data between an association of participants").) SIP discloses that it enables Internet endpoints to agree on a characterization of a session (EX1006, 8-9) including "negotiation of session parameters to establish the session" (EX1006, 10-11.) SIP also discloses a three-way handshake procedure to establish communication between nodes. (EX1006, 16 (describing "the INVITE/200/ACK three-way handshake used to establish SIP sessions"), 125 ("The INVITE transaction consists of a three-way handshake.").)

80. SIP discloses that it is a layered protocol including:

(i) a syntax and encoding layer (first layer),

(ii) a transport layer (second layer above first layer) used to transmit requests/responses between a client and server,[7]

(iii) a transaction layer (above the transport layer) which handles transactions like requests/responses (sent using the transport layer) using a client transaction instance, and

(iv) a transaction user (TU) layer above the transaction layer, where SIP entities (e.g., nodes or endpoints like client and server) are generally referred to as a TU. TUs create a client transaction instance when wishing to communicate.

(EX1006, 18-19; *see also id.*, 219-232 (describing first layer), 141-147 describing transport layer), 122-141 (describing transactions).)

[7] SIP can use either a reliable transport protocol (such as TCP) or an unreliable transport protocol (such as UDP). (EX1006, 142 ("All SIP elements MUST implement UDP and TCP."), 125 (sections 17.1.1.1-2); *see also id.*, 34 (Section 7.5), 52 (describing a SIP response that may "specify additional transport parameters such

81. As part of its protocol, SIP discloses an INVITE Client Transaction (ICT). (EX1006, 125-129 (section 17.1.1).) SIP discloses that this ICT involves setting up the session using a three-way handshake where the client transaction sends an INVITE, the server transaction sends responses, and the client transaction sends an acknowledgment (ACK). (EX1006, 125; *see also id.*, 10-16 (section 4 describing session setup by way of Figure 1).) Figure 5 shows the state machine for the ICT. (EX1006, 125.)

---

as … a change of SIP transport from UDP to TCP or vice versa."), 146 (section 18.2.2).).

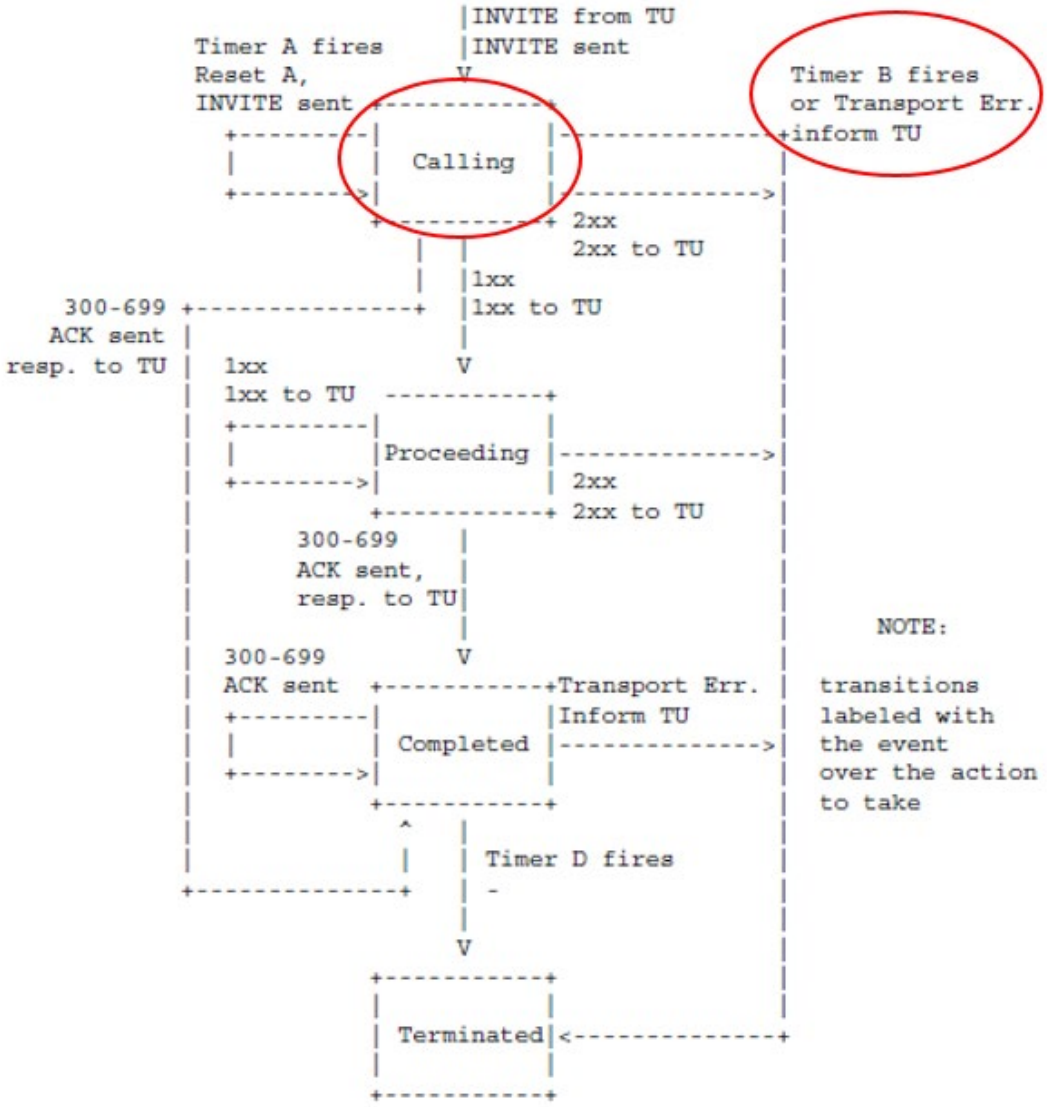


Figure 5: INVITE client transaction

(EX1006, 128 (FIG. 5 (annotated)).)

82. As shown, the ICT includes an initial "calling" state that a transaction user (TU) enters when the TU initiates a new transaction by sending an INVITE request (i.e., begin the handshake procedure). (EX1006, 125.) Upon the sending of the INVITE request, the client transaction instance for the TU starts timer B which

controls transaction timeouts and optional timer A (which controls retransmissions). (*Id.*)[8] Timer A is set to a value of T1, and timer B is set to a value of "64*T1 seconds," where "T1 is an estimate of the round-trip time (RTT)" between the client and server. (*Id.*, 125-126) The value of timer B allows for a certain number of retransmissions in case an unreliable transport protocol like UDP is used. (*Id.*, 126 ("The value of 64*T1 is equal to the amount of time required to send seven requests in the case of an unreliable transport.").)

83. When no response message to the INVITE request is received by the requesting TU for the duration of timer B such that the response message would have caused the calling state to advance to a subsequent state like a "proceeding" state or a "completion" state, timer B expires. (EX1006, 126, FIG. 5.) Upon expiration of timer B, the client transaction informs the TU that a timeout has occurred and

[8] Depending upon the implementation of the transport layer, timer A (which controls retransmissions of the INVITE message) is also started. (EX1006, 125-126.) For example, when the transport layer is implemented using an unreliable transport protocol like UDP, timer A must be started. (*Id.*) Timer A is not required for a transport layer implemented using a reliable transport protocol like TCP. (*Id.*; *see also id.*, 10; EX1035, 1.)

proceeds to the "terminated" state. (*Id.*) The client transaction is destroyed "the instant it enters the 'Terminated' state." (*Id.*, 128.)

IX. THE PRIOR ART DISCLOSES AND/OR SUGGESTS ALL RECITED FEATURES OF CLAIMS 1, 6, 12, 14, 17, AND 20 OF THE '217 PATENT

84. In my opinion, the prior art discussed below discloses and/or suggests the limitations of claims 1, 6, 12, 14, 17, and 20.[9]

A. The Combination of Morris and SIP Discloses and/or Suggests Claims 1, 6, 12, 14, 17, and 20

1. Claim 1

85. To assist the Board in navigating the mapping of the prior art to the arguably confusing claim language of claim 1, I initially note that Morris discloses and/or suggests the features recited elements 1.a-c, 1.e-f, 1.h-i, 1.j.(3), and 1.k.(1)-(2), which include, for example, features relating to a "second duration" for detecting a second time period (of a first type) in a first connection between a first node and a second node. The "second duration" is specified by metadata included in a first packet that is sent from the first node to the second node to set up the first connection, as recited in element 1.e. As I have explained below with respect to claim elements

[9] In this section (Section IX), I refer to exhibits other than those identified prior art reference(s) that disclose and/or suggest the claimed limitations. Such exhibits reflect the state of the art known to a person of ordinary skill in the art at the time of the alleged invention.

1.a-c, 1.e-f, 1.h-i, 1.j.(3), and 1.k.(1)-(2), these elements relate to features that largely track the processes (and components to perform them) discussed by Morris (which is a pre-grant publication of the ’454 application, from which the ’217 patent claims priority). (*See e.g.*, Ex. 1005, FIGS. 2, 7 (below), ¶¶[0035], [0052], [0073], [0083], [0088], [0106]; *see also* Sections IV, X.)

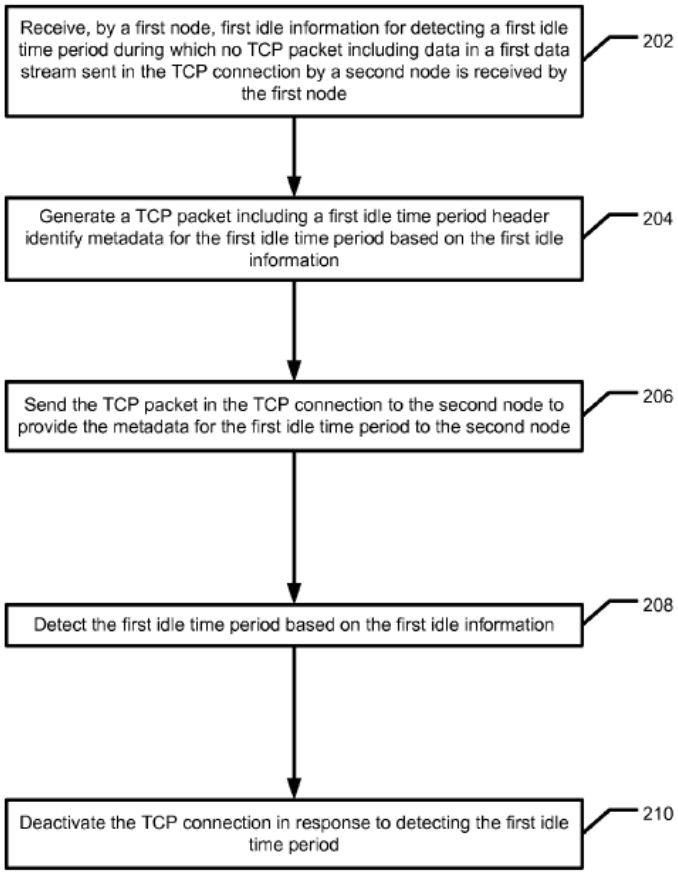


Fig. 2

(EX1005, FIG. 2.)

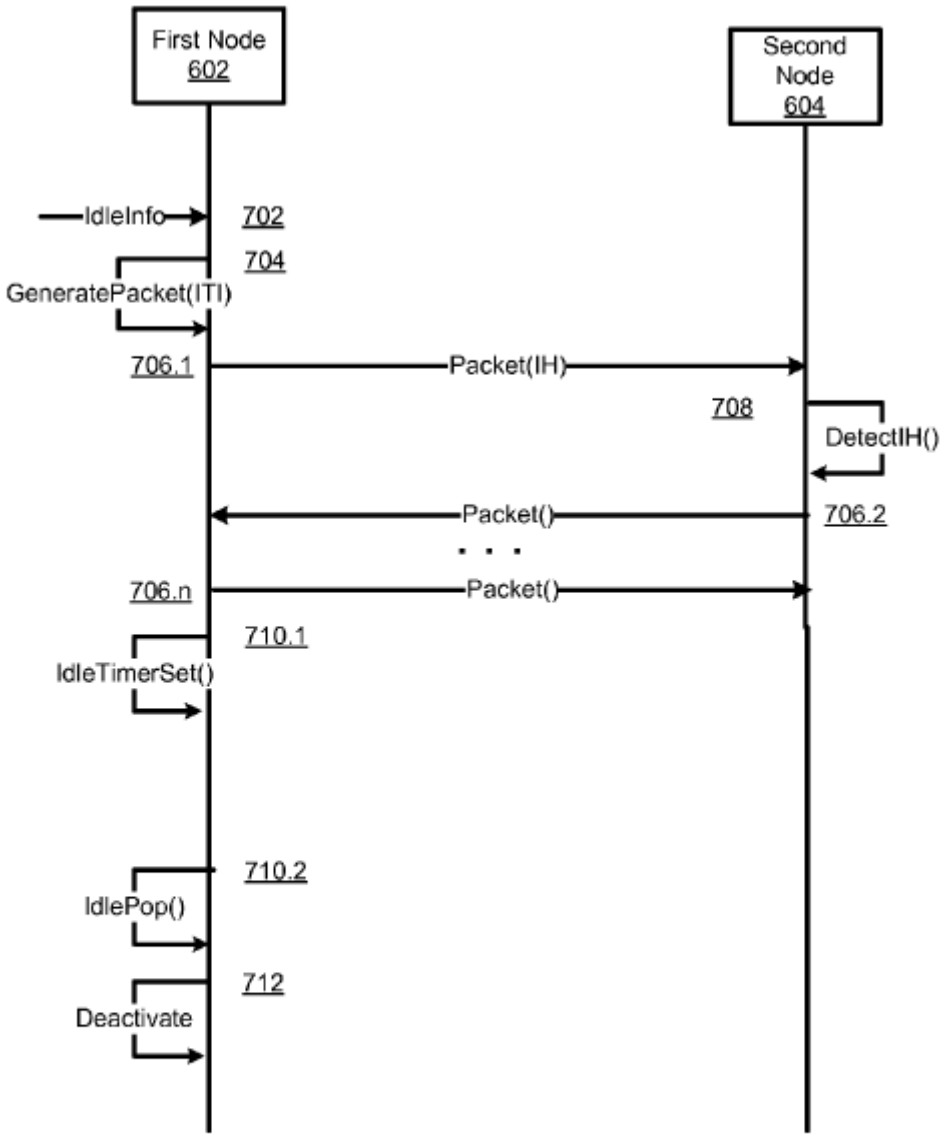


Fig. 7

(EX1005, FIG. 7.)

86. Morris, however, does not expressly disclose every feature recited in claim elements 1.d, 1.g and 1.j.(1)-(2), which relate, *inter alia*, to a "first duration"

for detecting a first time period (also of the first type) **during** at least a portion of the first connection including **a portion of the first connection setup**. (*See infra* Sections IX.A.1.d, IX.A.1.g, IX.A.1.j.(1)-(2).) But, as demonstrated below, a POSITA would have been motivated to modify the processes/system of Morris to provide features consistent with those recited in elements 1.d, 1.g and 1.j.(1)-(2) in light of the teachings/suggestions of SIP, in context of a POSITA's knowledge at the time. (*Id.*)

a) An apparatus, comprising:

87. I understand that this limitation is a preamble of the claim. Morris discloses it. For example, Morris is directed to "[m]ethods and systems … for sharing information for detecting an idle TCP [transmission control protocol] connection." (EX1005, ¶[0008]; *see also id.*, ¶¶[0005] ("The broader issue of enabling cooperation and negotiation between nodes in a TCP connection in detecting and managing idle, underactive, and/or dead TCP connections remains unaddressed."), [0006], [0009]-[0011].)

88. Morris discloses a hardware device 100 ("apparatus") in Figure 1, exemplified by first node 602 or second node 604 in Figures 6-7, for carrying out its

disclosed method(s). (EX1005, ¶¶[0021]-[0023], [0040], [0126]-[0127], claims 11, 22; FIGS. 1, 6-7.)[10]

> An exemplary device included in an execution environment that may be configured according to the subject matter is illustrated in FIG. 1.

(EX1005, ¶[0021].)

> FIG. 1 illustrates hardware device 100 included in

[10] For purposes of my analysis, I map the first node 602 in Morris to the "apparatus" in the preamble and the second node 604 to the "second node" recited in claim element 1.f. (*See also* EX1005, FIGS. 2-3 (referencing "first node" and "second node" to similarly refer to first node 602 and second node 604, respectively).) The claim appears to interchangeably, and arguably confusingly, use "apparatus" and "first node," as elements 1.f-h and 1.k.(1)-(2) refer to a "first node" that is involved with setting up the first connection with a "second node" and detecting first/second time periods while elements 1.i-k also refer to the "apparatus" being "configured" to perform some of the features. My analysis thus maps the first node 602 to the "apparatus" and "first node." (*See generally* Section IX.) Nothing in the '217 patent or its priority applications suggests an "apparatus" separate from a "first node," as claimed.

> execution environment 102 which includes instruction-processing unit (IPU) 104, such as one or more microprocessors; physical IPU memory 106 including storage locations identified by addresses in a physical memory address space of IPU 104 …

(EX1005, ¶[0023].)

> Moreover, the methods described herein may be embodied in executable instructions stored in a computer readable medium for use by or in connection with **an instruction execution machine, system, apparatus, or device, such as a computer-based or processor-containing machine, system, apparatus, or device**. As used herein, a "computer readable medium" may include one or more of any suitable media for storing the executable instructions of a computer program in one or more of an electronic, magnetic, optical, electromagnetic, and infrared form, such that the instruction execution machine, system, apparatus, or device may read (or fetch) the instructions from the computer readable medium and execute the instructions for carrying out the described methods. A non-exhaustive list of conventional exemplary computer readable media includes a portable computer diskette; a random access memory (RAM); a read only memory (ROM); an erasable programmable read only memory (EPROM or Flash memory); optical storage devices,

> including a portable compact disc (CD), a portable digital video disc (DVD), a high definition DVD (HD-DVD™), a Blu-ray™ disc; and the like.

(EX1005, ¶[0127].)

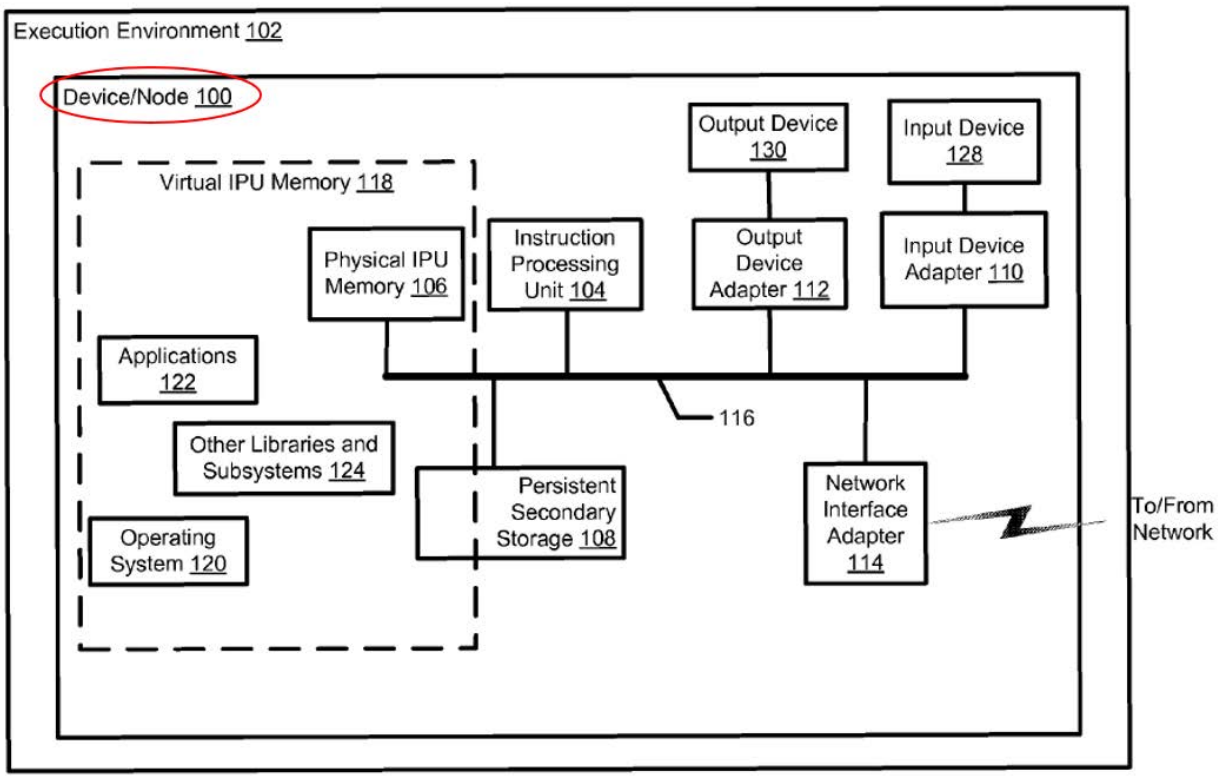


Fig. 1

(EX1005, FIG. 1 (annotated).)[11]

[11] Morris also refers to the device 100 as node 100, as shown in Figure 1.

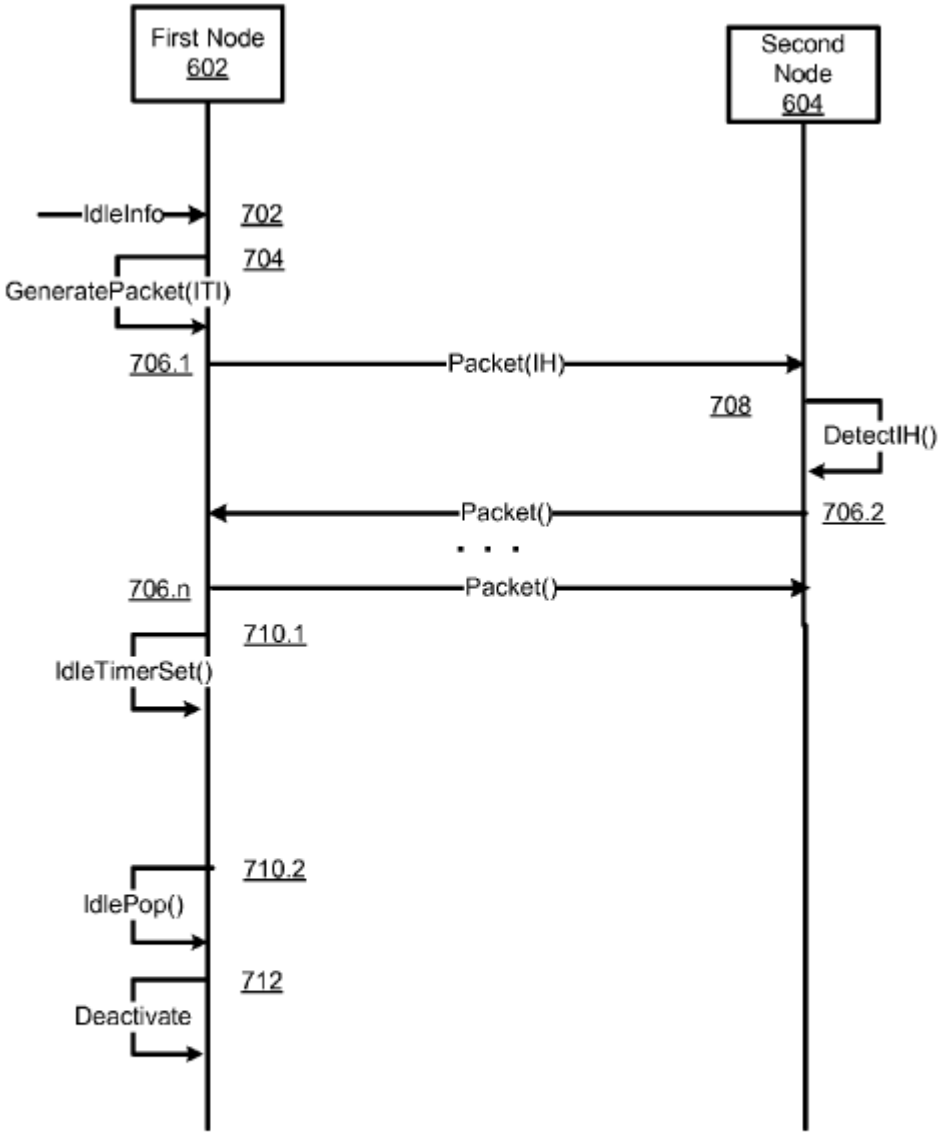


Fig. 7

(EX1005, FIG. 7.)

89. Thus, Morris discloses "[a]n apparatus," as claimed. (*See infra* Sections IX.A.1.b–j.)

b) a non-transitory memory storing instructions; and

90. Morris discloses this limitation. For example, Morris discloses that its hardware device 100 (e.g., first node 602) includes instruction-processing unit (IPU) memory 106 and persistent secondary storage 108 (collectively, "non-transitory memory") which stores instructions. (EX1005, ¶¶[0021]-[0028] (disclosing that the IPU 104 is an instruction execution machine which fetches instructions from memories 106 and/or 108), [0127] (disclosing that "the methods described herein may be embodied in executable instructions stored in a computer readable medium for use by or in connection with an instruction execution machine"), claims 11, 21-22 ("A computer readable medium embodying a computer program, executable by a machine, for sharing information for detecting an idle TCP connection, the computer program comprising executable instructions for:").)

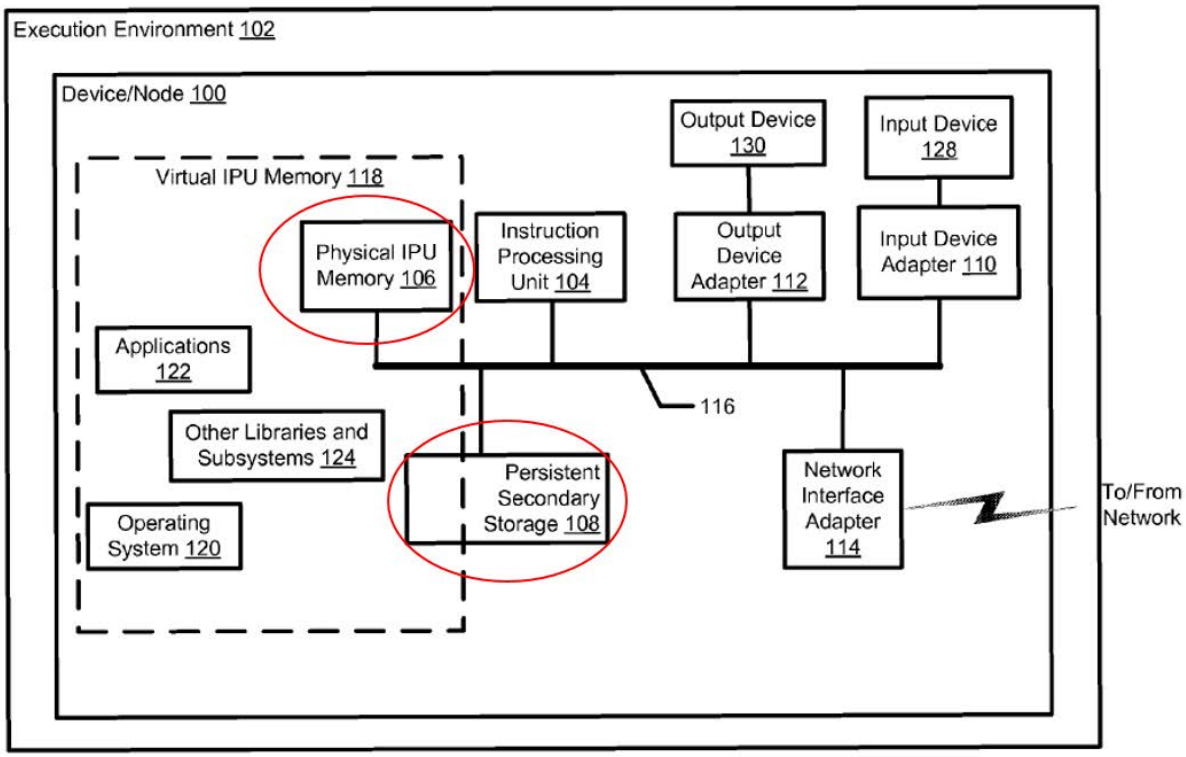


Fig. 1

(EX1005, FIG. 1 (annotated).)

91. Thus, Morris discloses "a non-transitory memory storing instructions," as claimed.

c) one or more processors in communication with the non-transitory memory, wherein the one or more processors execute the instructions to:

92. Morris discloses this limitation. For example, Morris discloses that its hardware device 100 (e.g., first node 602) includes IPU 104 which includes one or more microprocessors ("one or more processors") in communication with the memories 106 and 108. (EX1005, ¶¶[0021]-[0028], [0127], claims 11, 21-22.)

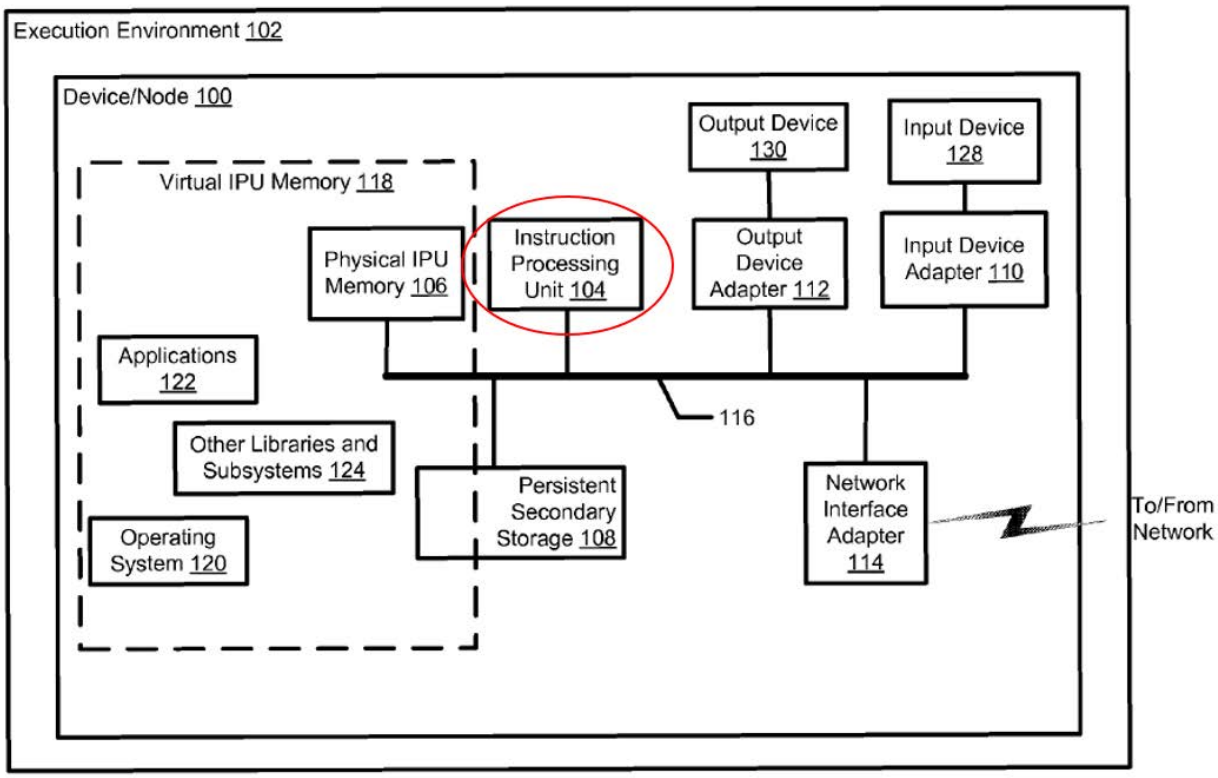


Fig. 1

(EX1005, FIG. 1 (annotated).)

93. Thus, Morris discloses "one or more processors in communication with the non-transitory memory," as claimed. Moreover, as discussed below, Morris discloses that the one or more processors execute the instructions set forth in claim elements 1.d-1.k, as claimed. (*See infra* Sections IX.A.1.d-k.)

d) receive first information on which at least a first duration for detecting a first type of time period is based;

94. Morris in combination with SIP discloses or suggests this limitation.

95. As I noted at the outset of Section IX, during connection setup when the metadata for the "second duration" is being sent by the first node to the node, the "first duration" recited in claim element 1.d is used to detect a first time period (of a first type). (*See* claim elements 1.d and 1.g.) After the first duration is changed to the second duration, the second duration is used to detect a second time period (also of the first type). (*See* claim element 1.h.)

96. Morris is directed to sharing metadata regarding idle time information (e.g., for the second duration) during connection setup and then, after the connection is set up, using the idle time information to detect a time period (e.g., a timeout) that subjects the connection to deactivation. (*See infra* Sections IX.A.1.e-f, IX.A.1.h.) Morris does not disclose detecting such a time period based on the claimed first duration during the connection setup itself. (*See infra* Section X.A.2.) However, a POSITA would have been motivated to consider and implement detection of a time period using the first duration during the connection setup, based on the teachings of SIP (EX1006) and the knowledge of a POSITA.

97. SIP, like Morris, is directed to a protocol in the networking field for enabling communication between two nodes. For example, SIP discloses the Session Initiation Protocol, which is an application-layer protocol for creating, modifying, and terminating sessions (between nodes) such as telephone calls, multimedia distribution, or multimedia conferences. (EX1006, Abstract; *see also id.*

at 8 (describing that "a session is considered an exchange of data between an association of participants").) Like Morris, SIP discloses that it enables Internet endpoints to agree on a characterization of a session (EX1006, 8-9) including "negotiation of session parameters to establish the session" (EX1006, 10-11.) (*See also* EX1005, ¶¶[0005] (discussing enabling cooperation and negotiation between nodes in a connection to detect and manage connections), [0008]-[0011].) Both SIP and Morris also disclose respective three-way handshake procedures to establish communication between nodes. (EX1006, 16 (describing "the INVITE/200/ACK three-way handshake used to establish SIP sessions"), 125 ("The INVITE transaction consists of a three-way handshake."); EX1005 ¶[0076] ("RFC 793 describes a 'three-way handshake' for establishing a TCP connection.").) Thus, SIP discloses networking features in the same technical field as Morris including negotiation of communication parameters and so a POSITA would have had reason to consider the teachings of SIP when implementing Morris.

98. SIP discloses that it is a layered protocol including:

(i) a syntax and encoding layer (first layer),

(ii) a transport layer (second layer above first layer) used to transmit requests/responses between a client and server,[12]

(iii) a transaction layer (above the transport layer) which handles transactions like requests/responses (sent using the transport layer) using a client transaction instance, and

(iv) a transaction user (TU) layer above the transaction layer, where SIP entities (e.g., nodes or endpoints like client and server) are generally referred to as a TU. TUs create a client transaction instance when wishing to communicate. (EX1006, 18-19; *see also id.*, 219-232 (describing first layer), 141-147 describing transport layer), 122-141 (describing transactions).)

99. As part of its protocol, SIP discloses an INVITE Client Transaction (ICT). (EX1006, 125-129 (section 17.1.1).) SIP discloses that this ICT involves setting up the session using a three-way handshake where the client transaction sends

[12] SIP can use either a reliable transport protocol (such as TCP) or an unreliable transport protocol (such as UDP). (EX1006, 142 ("All SIP elements MUST implement UDP and TCP."), 125 (sections 17.1.1.1-2); *see also id.*, 34 (Section 7.5), 52 (describing a SIP response that may "specify additional transport parameters such as … a change of SIP transport from UDP to TCP or vice versa."), 146 (section 18.2.2).).

an INVITE, the server transaction sends responses, and the client transaction sends an acknowledgment (ACK). (EX1006, 125; *see also id.*, 10-16 (section 4 describing session setup by way of Figure 1).) Figure 5 shows the state machine for the ICT. (EX1006, 125.)

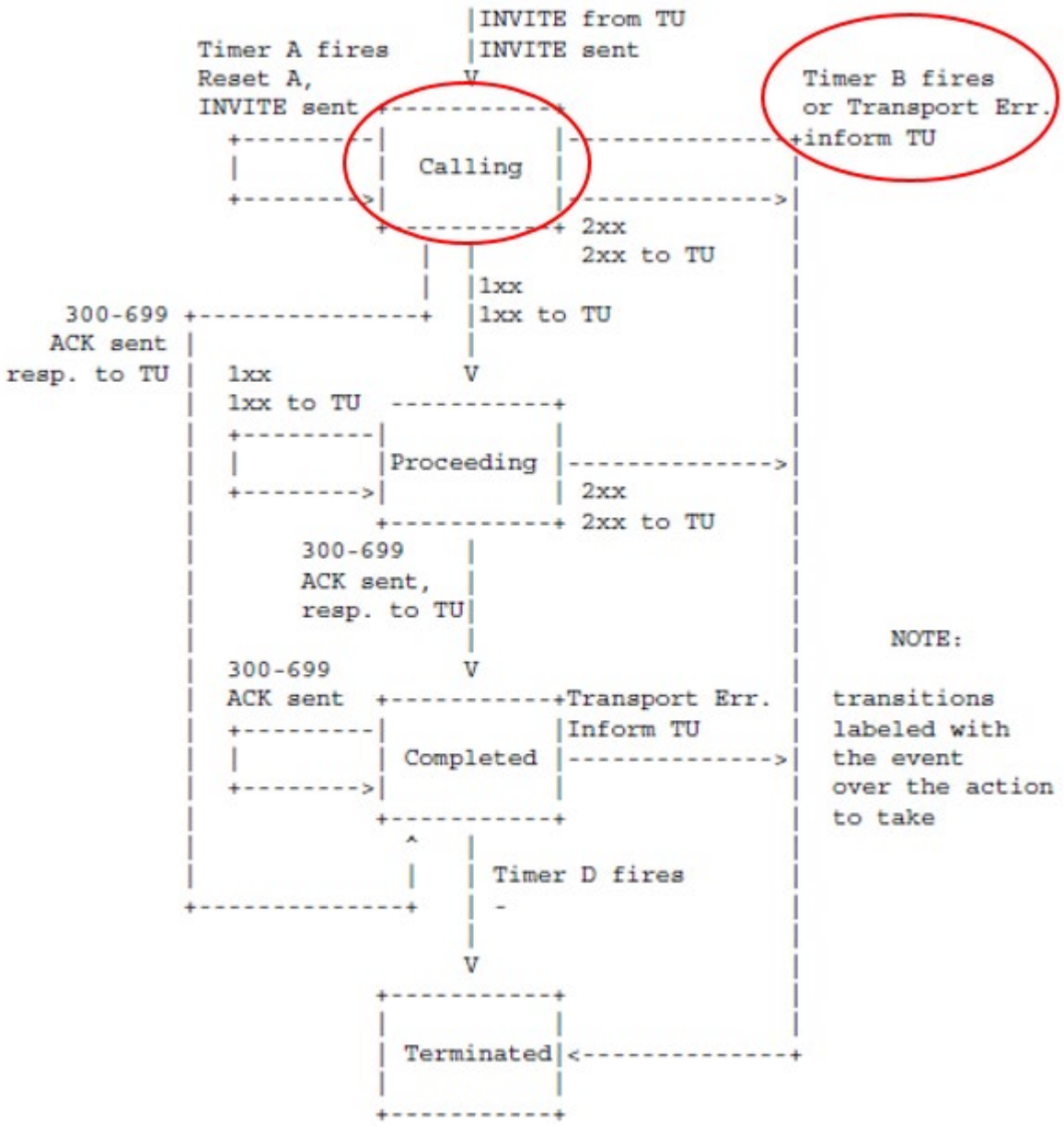


Figure 5: INVITE client transaction

(EX1006, 128 (FIG. 5 (annotated)).)

100. As shown, the ICT includes an initial "calling" state that a transaction user (TU) enters when the TU initiates a new transaction by sending an INVITE request (i.e., begin the handshake procedure). (EX1006, 125.) Upon the sending of the INVITE request, the client transaction instance for the TU starts timer B which controls transaction timeouts and optional timer A (which controls retransmissions). (*Id.*)[13] Timer A is set to a value of T1, and timer B is set to a value of "64*T1 seconds," where "T1 is an estimate of the round-trip time (RTT)" between the client and server. (*Id.*, 125-126) The value of timer B allows for a certain number of retransmissions in case an unreliable transport protocol like UDP is used. (*Id.*, 126 ("The value of 64*T1 is equal to the amount of time required to send seven requests in the case of an unreliable transport.").)

[13] Depending upon the implementation of the transport layer, timer A (which controls retransmissions of the INVITE message) is also started. (EX1006, 125-126.) For example, when the transport layer is implemented using an unreliable transport protocol like UDP, timer A must be started. (*Id.*) Timer A is not required for a transport layer implemented using a reliable transport protocol like TCP. (*Id.*; *see also id.*, 10; EX1035, 1.)

101. When no response message to the INVITE request is received by the requesting TU for the duration of timer B such that the response message would have caused the calling state to advance to a subsequent state like a "proceeding" state or a "completion" state, timer B expires. (EX1006, 126, FIG. 5.) Upon expiration of timer B, the client transaction informs the TU that a timeout has occurred and proceeds to the "terminated" state. (*Id.*) The client transaction is destroyed "the instant it enters the 'Terminated' state." (*Id.*, 128.)

102. As such, SIP discloses a mechanism to detect a timeout during the setup of a communication between two nodes using timer B, and closing the communication upon the timeout detection. A POSITA would have been motivated to consider and draw from such teachings of SIP when implementing Morris.

103. For example, Morris discloses negotiating the value of an idle time period (ITP) during connection setup and modifying timeout attributes for the established connection based on the negotiated ITP to detect timeouts. (*See, e.g.*, EX1005, FIGS. 1-7, ¶¶[0035]-[0040], [0052]-[0059], [0069], [0082]-[0083], [0088], [0096], [0104], [0106]-[0109].) Morris does not disclose detecting any timeouts during connection setup. However, Morris is concerned with wasting network bandwidth and resources by unnecessarily keeping communication between two nodes open due to idleness or underactivity or a connectivity problem (e.g., dead connection). (EX1005, ¶¶[0003]-[0005].) Thus, a POSITA would have been

motivated to draw from the teachings of SIP, which discloses detecting a timeout when no response to an initial setup message is received and terminating the communication between the nodes.[14]

104. In light of SIP's teachings a POSITA would have been motivated to configure Morris' apparatus, including the instructions stored in its non-transitory memory (*see* claim elements 1.b-c), such that the instructions included receiving a value ("first information") on which at least preliminary ITP information ("first

[14] A POSITA would have been further motivated to draw from the teachings of SIP when implementing Morris' system/method for the following reason(s). Morris discloses the processing of data streams and describes its execution environments include a video camera (i.e., producing video data). (EX1005 [0030], [0051]-[0052]) A POSITA would have known to consider sending video streams over more robust and reliable media streams and not over simple data streams. The SIP protocol is a media stream protocol. (EX1006, Abstract, 20 ("A session is a collection of participants, and streams of media between them, for the purposes of communication").) Thus, if a POSITA would have implemented Morris's system/method for transferring video data in a multimedia stream, a POSITA would have had reason to consider and consult a media stream protocol such as SIP which would make more sense for video data transfer.

duration for detecting a first type of time period") is based. As Morris explains, ITP information may be received from various sources including an IPU memory and/or secondary storage where such information is stored or via an application protocol layer, sockets component and/or directly from a network application. (EX1005, [0055]-[0059]; *see also id.*, [0050]-[0054], claims 11, 20, 22.)

105. Like SIP, a POSITA would have considered configuring Morris' apparatus such that the received value for the preliminary ITP information is based on a round-trip time between the communicating nodes (e.g., first/second nodes 602/604 in Figure 7) like the value 64*T1 on which the timer B is based in SIP. (*Id.*; EX1006, 125-126.) Alternatively, a POSITA would have considered configuring Morris' apparatus such that the received value for the first duration is based on a timeout parameter from a prior connection or connection attempt between the nodes.

106. For example, it was known in the networking field to save parameters from a previous connection between two nodes and use those saved parameters when the two nodes establish a new connection. Chang (EX1083), for instance, discloses that if a communication link between a mobile station and a base station is disconnected while communication is in progress, the mobile and base stations save connection parameters from the connection so they can be reused when the stations try to reconnect. (EX1083, Abstract, [0013], [0026], [0034], [0039].) Similarly, Stille (EX1084) teaches that data related to previous connection registration between

User Equipment and IP Multimedia Subsystem is stored and used to determine a timer duration for a current connection. (EX1084, 7:5-17, 6:33-7:3.) Cheng (EX1085) also teaches using timer values from previous connection attempts to determine timer values for current connection attempts. (EX1085, 2:1-17, 2:33-44, 5:15-29, 6:37-57, claims 1, 22). As a POSITA would have understood, the parameters from the previous connection could have been negotiated during the setup of the previous connection, as both Morris and SIP teach connection parameter negotiation during connection setup as discussed above.

107. In either case, i.e., whether the preliminary ITP information is based on a round-trip time or based on parameters from a previous connection, a POSITA would have considered a value for the preliminary ITP information which would have allowed for a certain number of retransmissions of the initial setup packet in case the initial transmission was not successful. (EX1006, 126; EX1008, 45 (describing retransmission timeout in the context of an established TCP connection).) A POSITA would have been motivated to try such a modification to Morris as there were known implementations of detecting timeouts during connection setup as discussed above with respect to a non-TCP implementation in SIP. Indeed, such implementations were also known in a TCP-based syste. For example, Bahk (EX1086) describes a TCP implementation with a connection setup timer that allows for three retransmissions of the initial setup packet before the timer

expires. (EX1086, [0064] ("The BSD-derived implementation allows a total of three retransmissions of the SYN packet because a connection setup timer expires at 75 seconds, that is, 3 seconds before the fourth retransmission of the SYN packet."); *see also* Abstract.)

108. All of the above teachings would thus have further motivated a POSITA to add the ability of detecting a timeout based on the preliminary ITP when no response to an initial setup message is received and terminating the communication between the nodes to the disclosures of Morris.

109. For example, with respect to the alternate approach related to using a parameter from a prior connection, Morris teaches that ITP information negotiated during the connection setup and used, e.g., to set a timeout for the established connection may be based on information in an ITP header in a previously received packet in the same connection. (EX1005, [0058], [0094], claims 7, 12; *see also id.*, [0095] (disclosing the ITP may be dynamic).) Based on these teachings, a POSITA would have been motivated to further configure Morris such that an ITP negotiated during a previous connection between the nodes would have been saved and used during a new connection setup between the nodes until a new ITP value is negotiated during the new connection setup, as this was a known concept at the time of the alleged invention. (EX1083, Abstract, [0013], [0026], [0034], [0039]; EX1084, 7:5-17, 6:33-7:3; EX1085, 2:1-17, 2:33-44, 5:15-29, 6:37-57, claims 1, 22).)

110. A POSITA would have understood that if a connection between the nodes existed before, some recognizable ways to implement a timeout detection mechanism during connection setup between the same two nodes would have involved using a value based on the previously calculated round-trip time between the communicating nodes (e.g., 64*T1 as taught by SIP) or a function of the previously negotiated ITP between the two nodes (e.g., a multiple of the previous ITP to allow for the other node to read, process, and make a decision regarding establishing a new connection based on the proposed parameter(s) in the setup packet). This way, the sending node that is waiting for a response (e.g., acknowledgment) from the other node would have known if the response to the initial setup packet is taking longer than expected. (*Id.*)

111. And like SIP, the proposed modification to Morris would have involved starting the preliminary ITP in response to detecting at least one of a received packet or a sent packet such as in response to detecting a packet sent to initiate connection setup in Morris. (EX1005 ¶[0076].) And once a value for the ITP is negotiated during connection setup and the connection is established, the Morris-SIP system would have updated the preliminary ITP information to the negotiated ITP information such that the negotiated ITP information would have been used to detect a similar timeout in the established connection, as discussed below. (*Id.*)

112. For example, in the proposed modification, the preliminary ITP information (set, e.g., based on the round-trip time or a saved parameter from a previous connection) would have been used to configure a user-timeout type of timeout similar to Morris, i.e., SIP's Timer B ("first type of time period"), such that it would have been initiated upon sending of the initial connection setup packet in Morris' TCP-based connection establishment procedure (three-way handshake). (EX1005 ¶¶[0069], [0076], [0120].)

113. As a POSITA would have understood, such a user-timeout type of timeout would have been initiated at the sending node upon the sending of a non-empty packet and would run at least until an acknowledgment (ACK) associated with the sent packet is successfully received at the sending node. (EX1005 ¶[0069]; EX1018, 3.)[15] This type of user timeout operates similar to the Timer B in SIP, as detection of both of them leads to the closing of the connection. For example, as discussed above, SIP's timer B starts upon the sending of the Invite message and runs until a response to the Invite message is received. (EX1006, 126-128, FIG. 5.)

[15] While RFC 793, which describes TCP and upon which the Morris disclosure is based, does not describe initiating a timeout similar to a user timeout during connection setup, a POSITA would have been motivated to modify Morris to include such a mechanism in the Morris-SIP system, as discussed below.

Upon expiration of timer B, the connection is terminated. (*Id.*) Similarly, a user timeout in Morris starts upon the sending of a non-empty packet and runs until an acknowledgment to the sent packet is successfully received. (EX1005, [0010], [0106]-[0109], FIG. 2 (block 210), FIG. 7 (message 712), claims 11, 13, 22.) Upon expiration of the user timeout, the connection is terminated. (*Id.*)

114. Thus, while SIP's Timer B runs during connection setup and Morris's TCP user timeout runs after the connection is set up, a POSITA would have considered both SIP's Timer B and Morris' user timeout to be similar types of timeouts as they function similarly (e.g., they are both initiated upon the sending of a packet and if no response to the packet is received for the timeout duration, the connection is deactivated, as recited in claim elements 1.g-h). As such, when implementing SIP's teachings regarding Timer B used for timeout detection during connection setup in Morris' system/method, as discussed herein, a POSITA would have also drawn from Morris's teachings regarding implementation of the similar TCP user timeout in Morris. And a POSITA would have understood that the "first type of time period" (e.g., claim elements 1.g-h) in the Morris-SIP combination is a type of time period characterized by, e.g., starting upon the sending of a response-eliciting packet (like Morris's ACK-eliciting packets or SIP's Invite message) and

if no response to the sent packet is received for the duration of the time period, leading to deactivation of the connection. (*See infra* Sections IX.A.1.g-h.)[16]

115. A POSITA would have been motivated and would have known how to modify Morris to incorporate SIP's teachings, for the reasons discussed above and below. (*See, e.g.*, EX1006, 125-127, FIG. 5.) For example, the use of detecting a timeout during setup or establishment of communication between two nodes and closing down the communication upon timeout detection was well-known at the time of the alleged invention. Farrell (EX1019), for instance, discloses initiation of similar timeout detection mechanism during connection setup upon sending of an initial connection setup message N-CONNECT in a UDP environment. (*See, e.g.*, EX1019 ¶¶[0182]-[0183], [0189], [0410], FIGS. 8G, 21; EX1034 ¶¶[0010], [0019]-[0026] (disclosing initiation of similar timeout detection mechanism during connection setup upon sending of a request for connection establishment, where the timer is based on round-trip time between the two nodes); *see also* EX1086 (Bahk), [0064] (describing a TCP implementation with a connection setup timer).) Indeed, a POSITA would have recognized that networking concepts implemented on one layer of the network stack (such as the timeout detection mechanism during session

[16] As discussed below, the Morris-SIP system also teaches that the first type of time period is started in response to a received packet. (*See supra* Section IX.A.4.)

establishment setup in the application layer in SIP) could have been implemented on another layer of the network stack, as evidenced by Farrell (EX1019) which discloses a similar timeout detection mechanism on the network layer in a UDP-based environment. (EX1019 ¶¶[0182]-[0183], [0189], [0410], FIGS. 8G, 21.) Indeed, based on the knowledge of a POSITA discussed above, a POSITA would have understood that the timeout detection mechanisms (like those discussed below in claim elements 1.g-1.h) could have been implemented on an application layer (or another layer other than a transport layer). (*See infra* Section IX.A.1.f.)

116. In this regard, a POSITA would have known that a connection or session between two communicating nodes is not limited to the transport layer, but can exist on other layers as well, and the problems associated with such connections like wasted bandwidth apply regardless of the layer. A POSITA would have also known that in the networking field, protocols often mix and match concepts from other protocols including the use of timeout mechanisms. (EX1009, 101-102 (discussing use of keep-alive in TCP context on transport layer); EX1007, 1:14-28, 1:49-50, 1:63-2:7 (discussing use of keep-alive at application layer for application level timeouts), 4:8-11, 4:25-37, 5:21-28; EX1028, 16:66-7:2, 20:36-41 (discussing null segment timer, which operates similar to keep-alive, in UDP context), 23:45-59.). Thus, a POSITA would have understood that certain problems and mechanisms regarding the establishment and management of connections between

two nodes may be common regardless of whether the connection is at the transport layer or the application layer, and therefore a POSITA would have considered solutions to such problems regardless of the layer.

117. In view of the foregoing, a POSITA would have had the capability and knowledge to implement, and would have tried the above modification to include a timeout detection mechanism during connection setup in Morris (as taught by SIP) while ensuring that Morris' apparatus performed properly. A POSITA would have understood based on the above discussion that there would be a limited number of options for implementing the type of timeout discussed above—either after connection establishment, before connection establishment (i.e., during the connection setup), or during both the connection setup and after connection establishment. Therefore, a POSITA would have had a reasonable expectation of success that the combined Morris-SIP system would have been operated as expected (e.g., establish connection between nodes including negotiation of communication parameters during connection setup along with the ability to detect a lack of activity during the connection setup, and applying the negotiated communication

parameter(s) to detect a lack of activity in the connection after connection establishment).[17]

118. The above modification would have been a mere combination of known technologies (e.g., SIP's disclosed concept of detecting a timeout during communication establishment between nodes incorporated into Morris' disclosed systems and methods for establishing connections while allowing for negotiation of connection parameters), according to known methods (e.g., timeout detection mechanisms during connection setup as widely known in the networking field), to achieve predictable results (e.g., the ability to detect and shut down idle, underactive, or dead link during connection setup).

119. Therefore, the Morris-SIP system discloses or suggests one or more processors executing the instructions for receiving a value ("first information") on which at least preliminary ITP information ("first duration") for detecting a user-

[17] The proposed modification would have resulted in the Morris-SIP system implementing a protocol that results in the first connection that is not a TCP extension connection that involves a TCP extension. (*See infra* Section IX.A.5.)

timeout type of timeout similar to Morris, i.e., SIP's Timer B ("first type of time period") is based.[18]

e) generate a first packet including a first parameter field identifying first metadata for use in determining a second duration for detecting the first type of time period;

120. The Morris-SIP combination discloses or suggests this limitation. For example, the Morris-SIP combination discloses generating a packet ("first packet") including an ITP header identifying first metadata (e.g., a value) ("first parameter field identifying first metadata") for use in determining negotiable ITP information ("second duration") for detecting the user-timeout type of timeout similar to Morris, i.e., SIP's Timer B ("first type of time period"). (EX1005, [0073]-[0081] (describing this feature with respect to block 204 in Figure 2 and exemplary packet with ITP header 820 in Figure 8), [0082] (similarly describing this feature with reference to

[18] A POSITA would have had the same motivation, skills, capabilities, and expectation of success, as described above for element 1.d in configuring the one or more processors in the combined Morris-SIP system/process to execute the instructions to perform the features recited in the remaining claim elements. (*See*, *e.g.*, *infra* Sections IX.A.1.e-k, IX.A.2-5.)

message 704 in Figure 7), claims 11, 22; *see also supra* Section IX.A.1.d (discussing "first type of time period").)[19]

[19] I reference the ITP information included in the first packet in Morris being mapped to the "second duration" as "negotiable" ITP information to distinguish it from the "preliminary" ITP information being mapped to the "first duration" in element 1.d. (EX1005, [0005], [0094], [0105], [0124].)

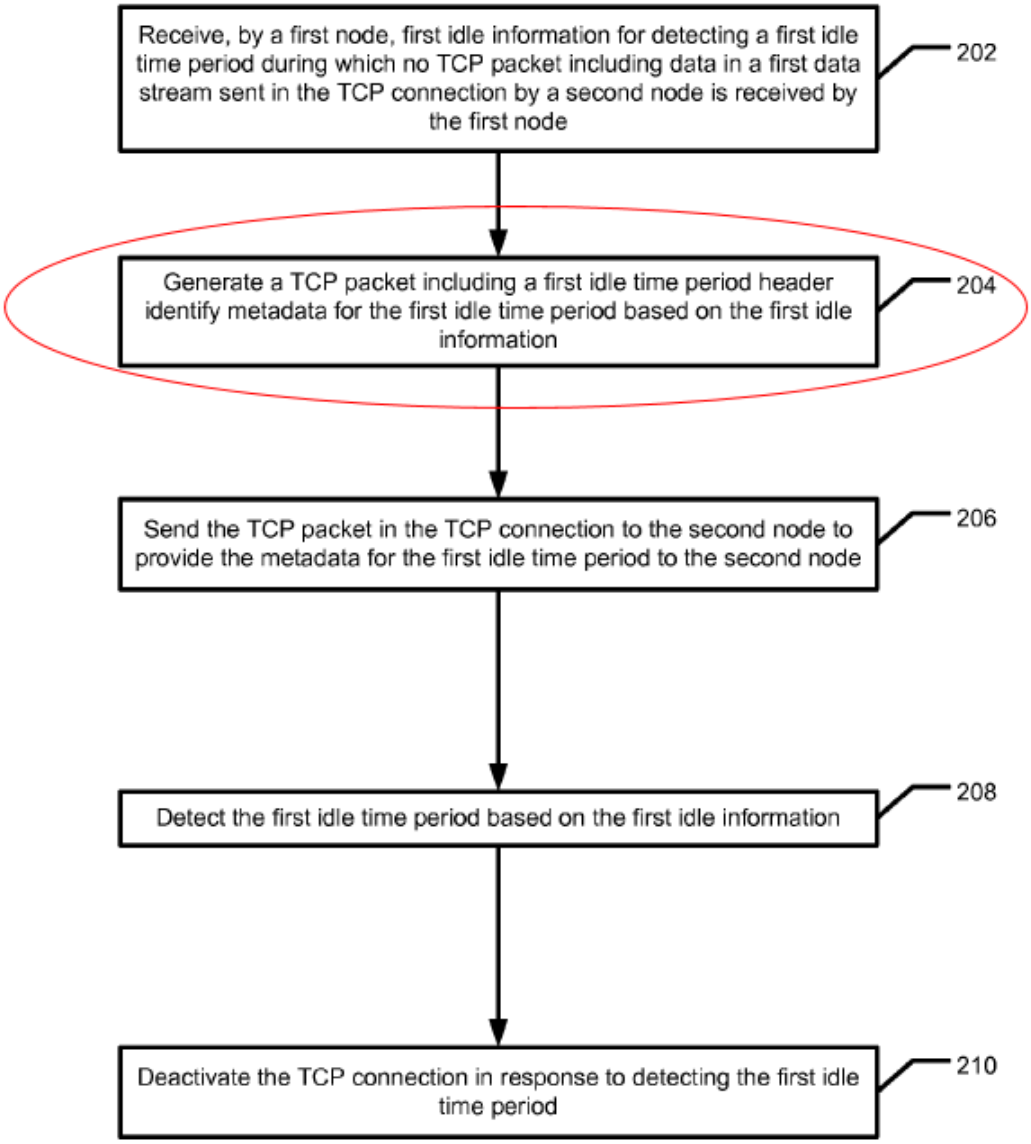


Fig. 2

(EX1005, FIG. 2 (annotated).)

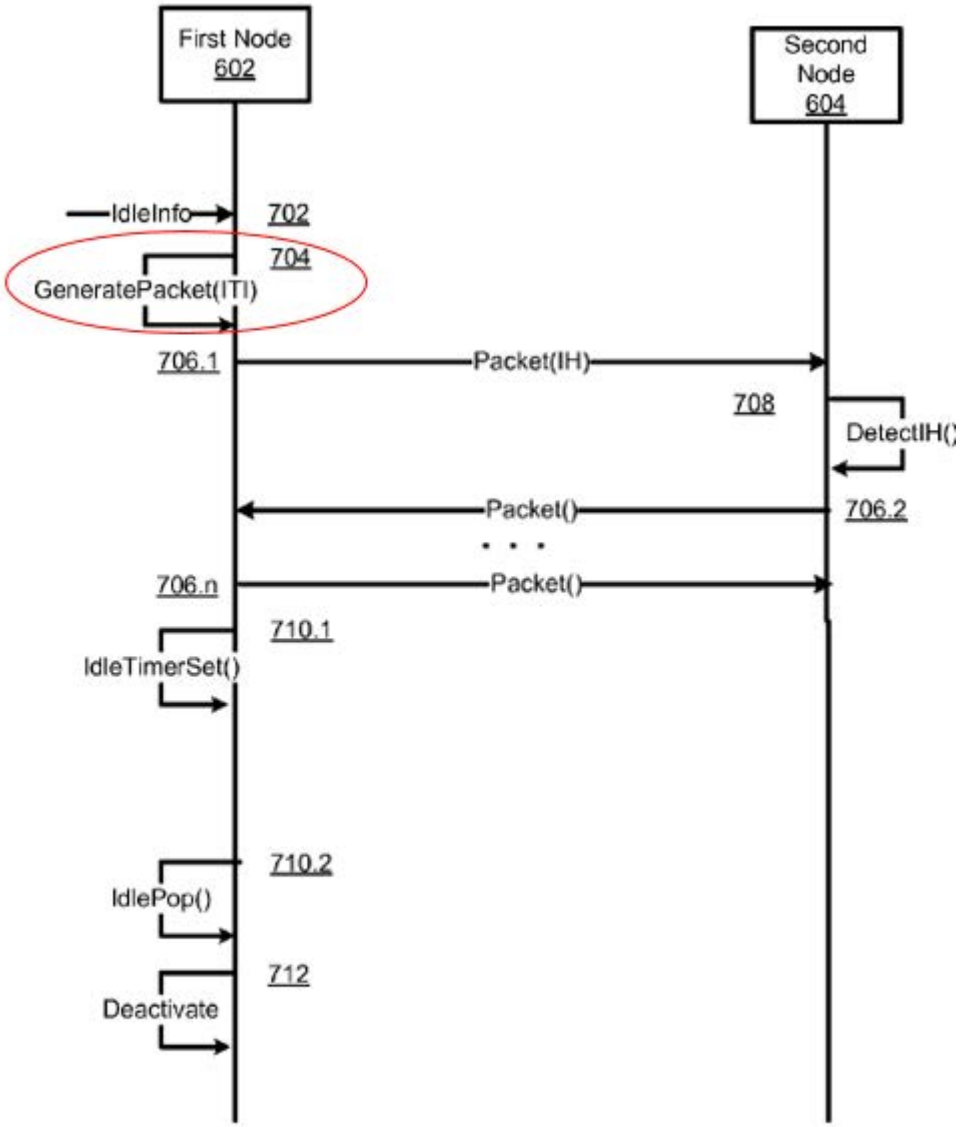


Fig. 7

(EX1005, FIG. 7 (annotated).)

121. Therefore, the Morris-SIP combination discloses or suggests generating a packet ("first packet") including an ITP header identifying first metadata (e.g., a value) for use in determining negotiable ITP information ("second duration") for

detecting the user-timeout type of timeout similar to Morris, i.e., SIP's Timer B ("first type of time period").

f) set up a first connection, by sending, from a first node to a second node, the first packet to provide the first metadata to the second node, for use by the second node in determining the second duration for detecting the first type of time period;

122. The Morris-SIP combination discloses or suggests this limitation. For example, the Morris-SIP combination discloses setting up the first connection, by sending, from the first node 602 ("first node") to a second node 604 ("second node"), the first packet to provide the first metadata to the second node 604, for use by the second node 604, in determining the negotiable ITP information ("second duration") for detecting the user-timeout type of timeout similar to Morris, i.e., SIP's Timer B ("first type of time period"). (EX1005, [0083]-[0084] (describing this feature with respect to block 206 in Figure 2 and message 706.1 in Figure 7), claims 11, 22, ¶[0076] (describing the three-way handshake to set up a connection involving sending a first packet from one node to another); EX1008, 27-28 (RFC 793 describing three-way handshake); *see also* n.10 (discussing the mapping of the first node 602 to both the "apparatus" and "first node," and explaining that nothing in the '217 patent or its priority applications suggests an "apparatus" separate from a "first node," as claimed).)

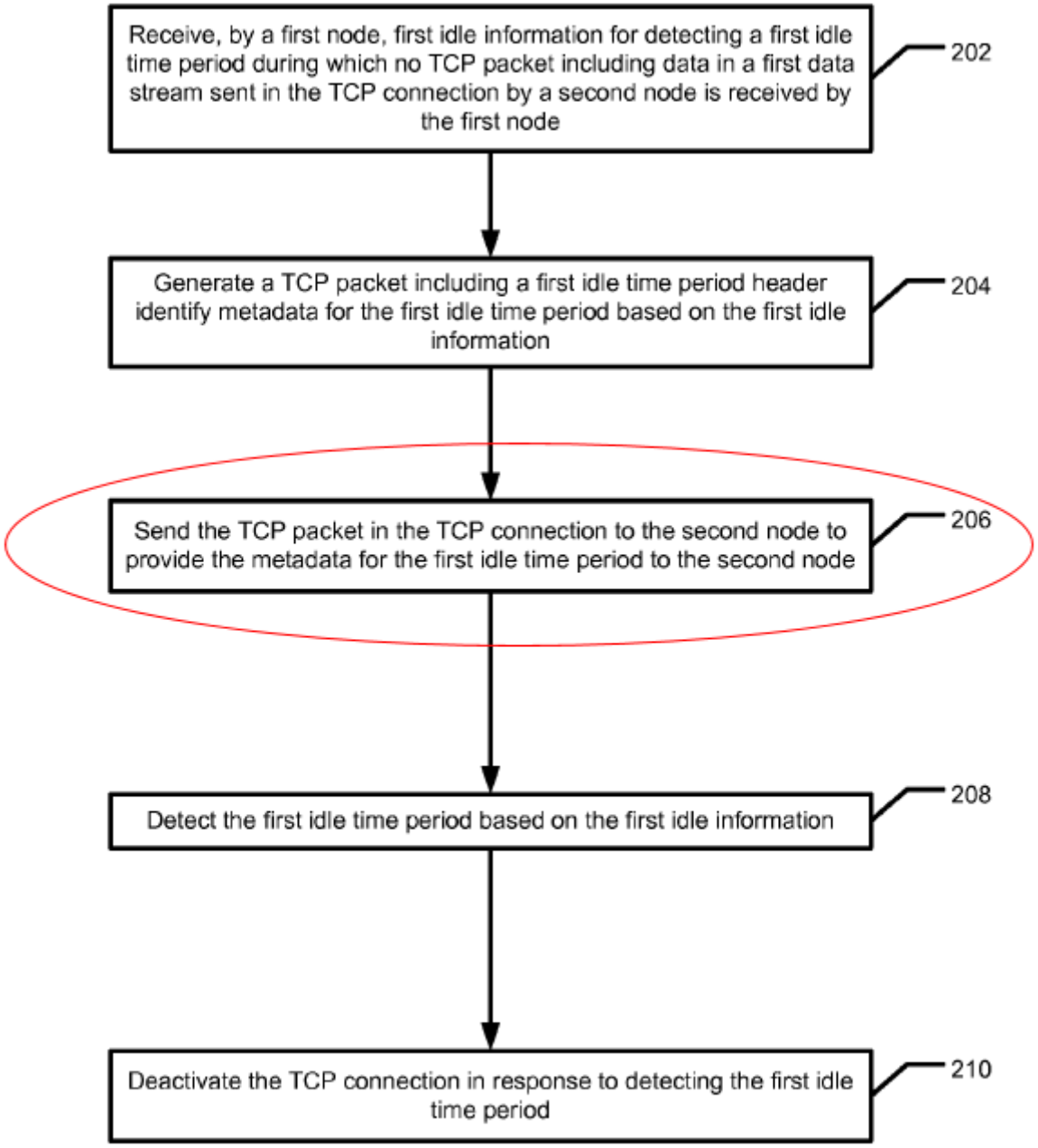


Fig. 2

(EX1005, FIG. 2 (annotated).)

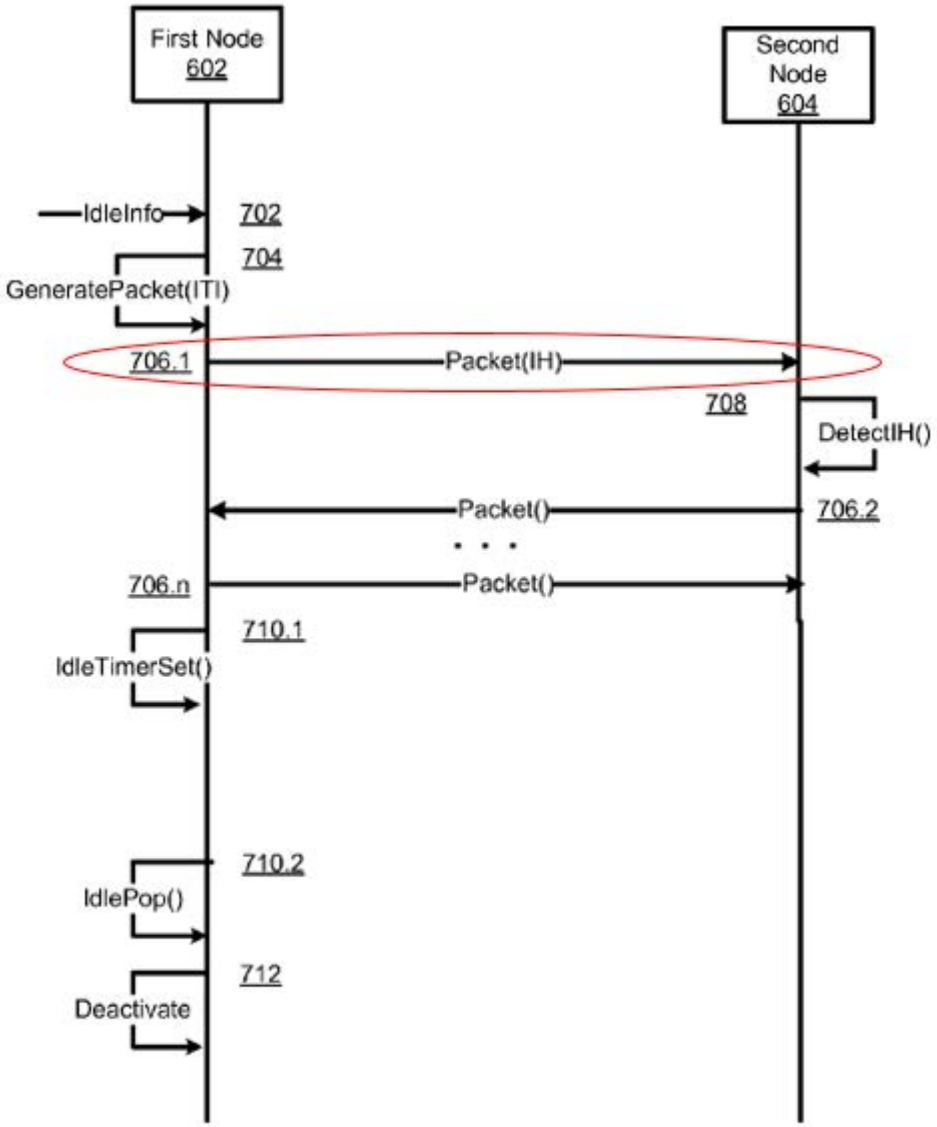


Fig. 7

(EX1005, FIG. 7 (annotated).)

g) in response to detecting, based on the first duration and by the first node during at least a portion of the first connection including at least a portion of the first connection set up, a first time period of the first type of

time period, at least partially close the first connection; and

123. The Morris-SIP combination discloses or suggests this limitation. This feature is taught for at least the reasons explained in claim element 1.d. (*See also supra* Section IX.A.1.d.) For example, the Morris-SIP system discloses that in response to detecting, based on the preliminary ITP information ("first duration") and an absence of an ACK packet acknowledging the sent first packet in element 1.f and by the first node 602 ("first node") during at least a portion of the first connection including at least a portion of the first connection setup, a preliminary ITP ("first time period") of the user-timeout type of timeout similar to Morris, i.e., SIP's Timer B ("first type of time period"), at least partially close the first connection. (EX1005, [0088]-[0089] (describing block 208 in Figure 2 relating to detection of the ITP at first node), [0092]-[0093] (similar), [0096]-[0097] (describing detection of ITP at first node 602 by way of messages 710.1 and 710.2 in Figure 7), [0106]-[0109] (describing block 210 in Figure 2 and message 712 in Figure 7 which relate to first node 602 at least partially closing the connection by releasing resources allocated for the connection).) While the cited Morris disclosures concern at least partially closing a connection in response to the detection of an ITP during an established connection, these teachings would equally apply to the detection of a preliminary

ITP during connection setup disclosed in the Morris-SIP combination discussed above. (*See supra* Section IX.A.1.d.)

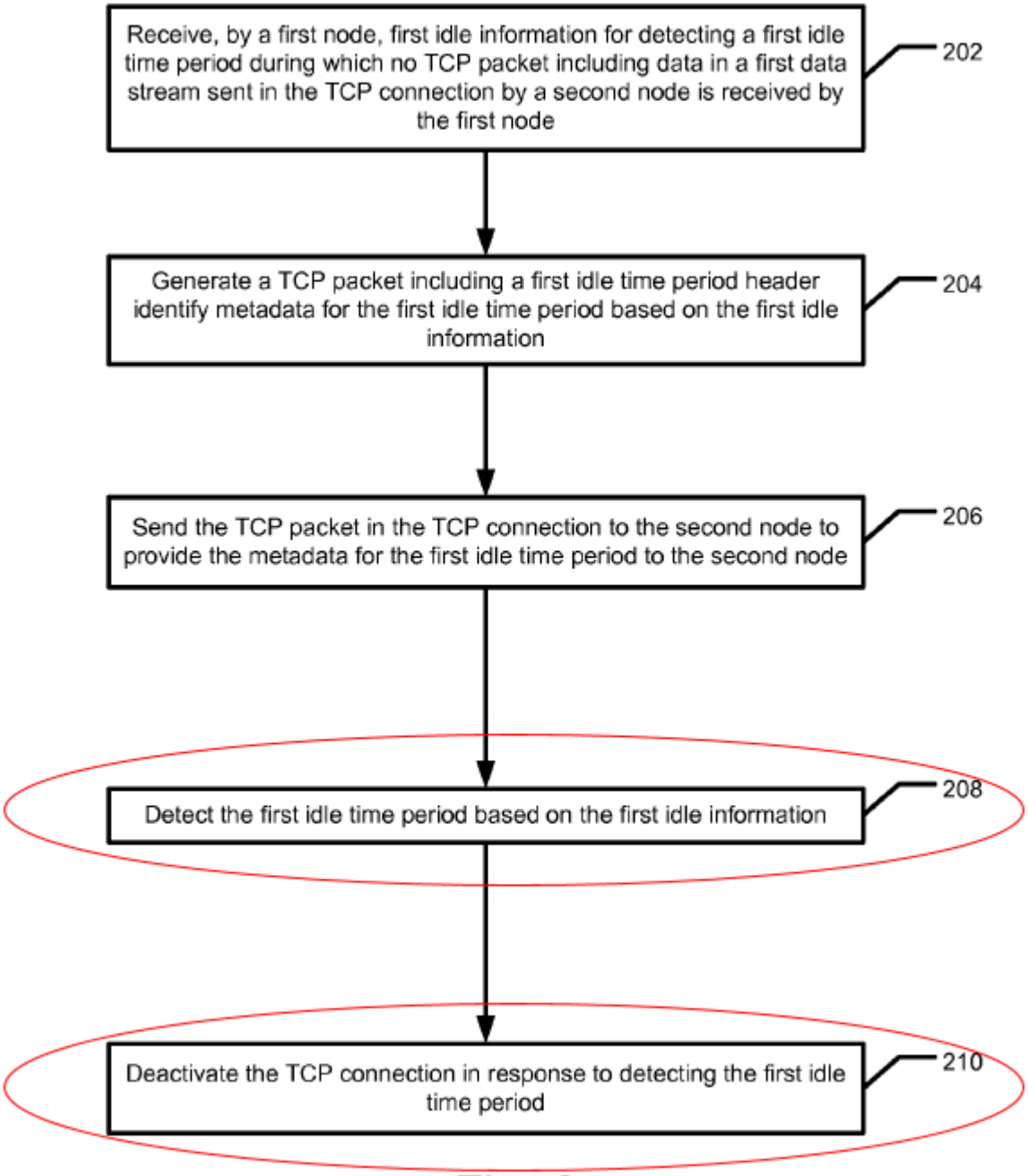


Fig. 2

(EX1005, FIG. 2 (annotated).)

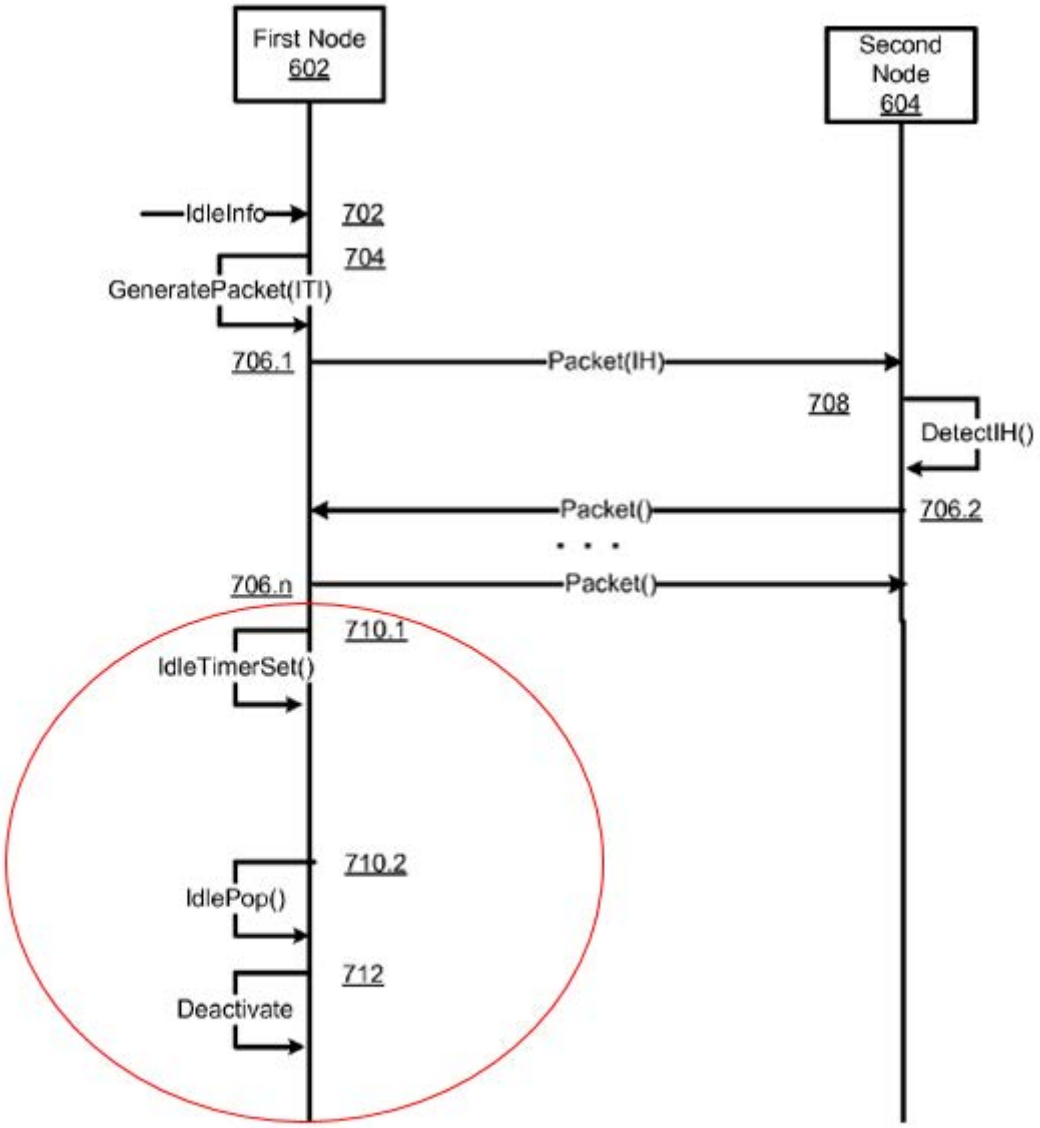


Fig. 7

(EX1005, FIG. 7 (annotated).)

h) in response to detecting, based on the second duration and by the first node after the first duration is changed to the second duration, a second time period of the first type of time period, at least partially close the first connection;

124. The Morris-SIP combination discloses or suggests this limitation. For example, the Morris-SIP combination discloses that in response to detecting, based on the negotiable ITP information ("second duration") and an absence of at least one ACK packet acknowledging a previously sent ACK-eliciting packet and by the first node 602 ("first node") after the first connection setup (e.g., after the three-way handshake) and after the preliminary ITP information ("first duration") is changed to the negotiable ITP information ("second duration"), a negotiated ITP ("second time period") of the user-timeout type of timeout similar to Morris, i.e., SIP's Timer B ("first type of time period"), at least partially close the first connection. (EX1005, [0088]-[0089] (describing block 208 in Figure 2 relating to detection of the ITP at first node), [0092]-[0093] (similar), [0095] (describing negotiated ITP), [0096]-[0097] (describing detection of ITP at first node 602 by way of messages 710.1 and 710.2 in Figure 7), [0106]-[0109] (describing block 210 in Figure 2 and message 712 in Figure 7 which relate to first node 602 at least partially closing the connection by releasing resources allocated for the connection), [0105] (negotiated ITP), [0124] (similar), claims 11, 22.)

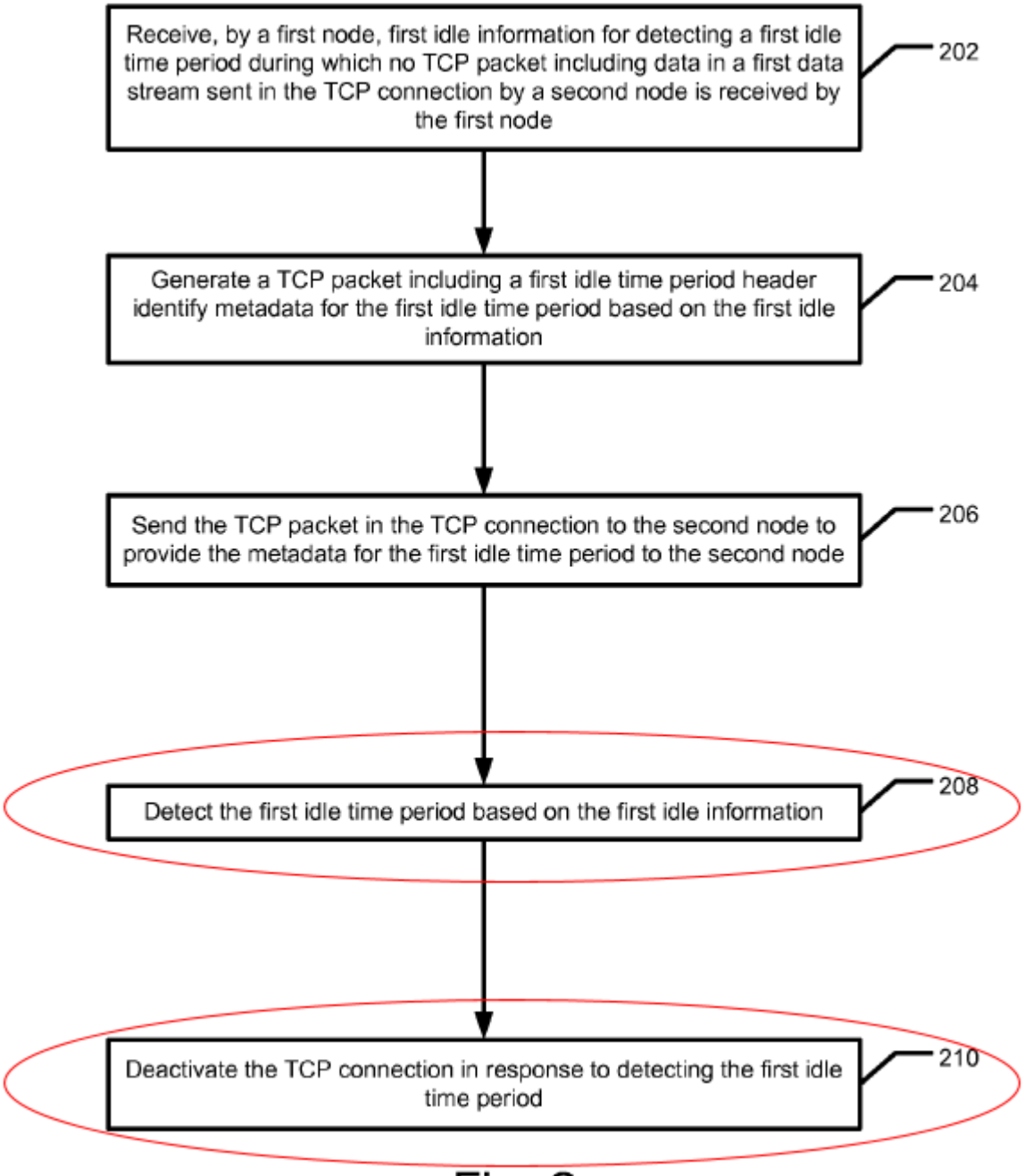


Fig. 2

(EX1005, FIG. 2 (annotated).)

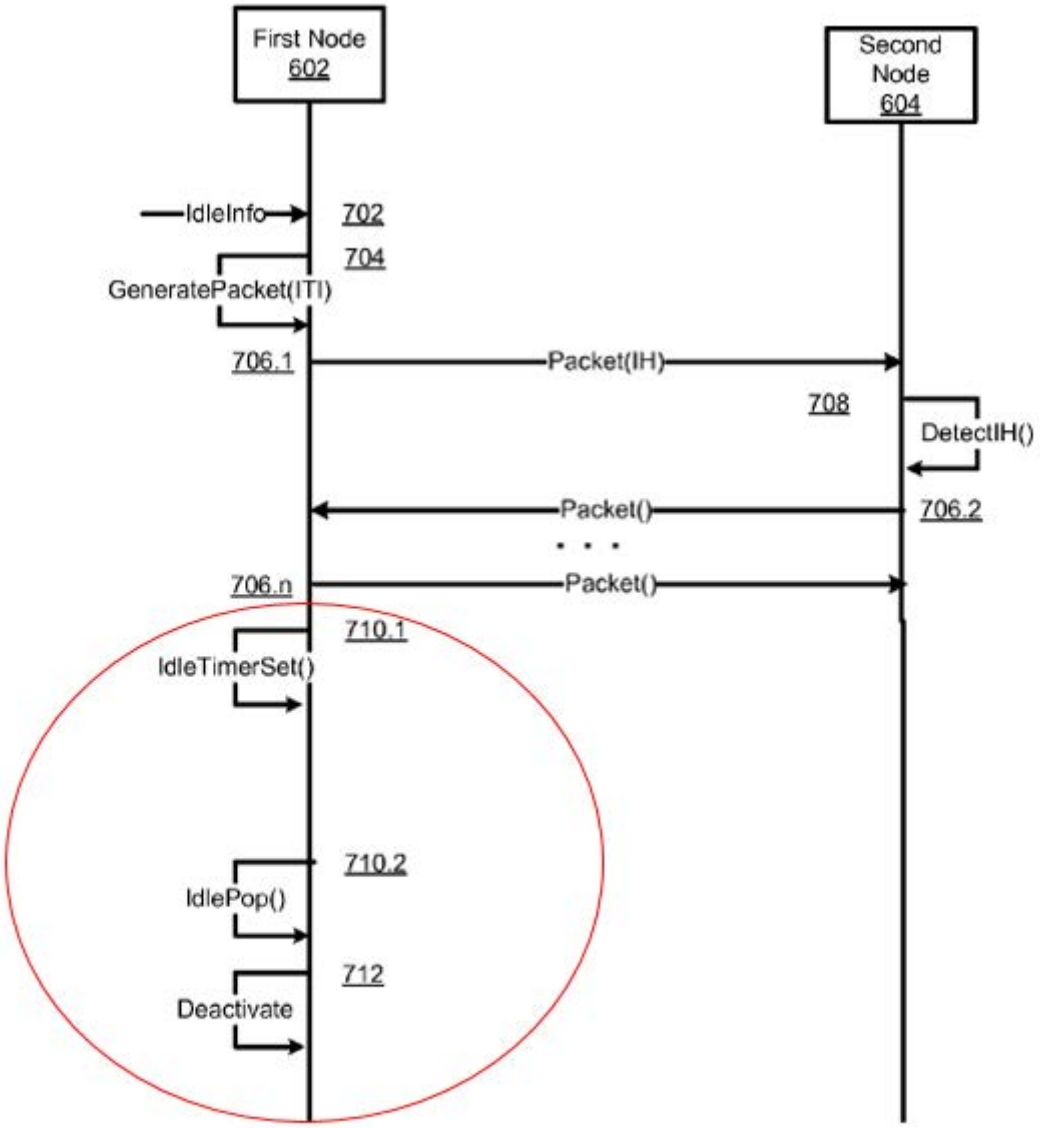


Fig. 7

(EX1005, FIG. 7 (annotated).)

125. In the Morris-SIP combination, after the connection is set up which included sharing the negotiable ITP information with the other node using the first packet (claim elements 1.e-1.f), the preliminary ITP information ("first duration")

being used to detect a timeout during the connection setup would have been updated ("changed") to the negotiable ITP information ("second duration") included in the sent packet. (*See supra* Section IX.A.1.d, IX.A.1.e-g.) A POSITA would have understood that in the proposed combination, the reason for sharing the negotiable ITP information in the packet sent to the second node is to set a timeout attribute like the user timeout (and like SIP's Timer B) to a negotiated value. (*Id.*; EX1005, [0008], [0069], [0072], [0086].) Indeed, as discussed above, Morris teaches that the ITP can be dynamically changed. (EX1005, [0095]; *see also id.*, [0076] (disclosing that the ITP can be exchanged in packets communicated in an established connection, i.e., after the connection setup procedure).)

i) wherein the apparatus is configured such that: during the second time period of the first type of time period, an empty packet is sent without causing a timer utilized to detect the second time period to be reset;

126. The Morris-SIP combination discloses or suggests this limitation. For example, the apparatus in the Morris-SIP system is configured such that during the negotiated ITP ("second time period"), an empty packet is sent without a timer associated with the negotiated ITP ("timer utilized to detect the second time period") being reset. (EX1005, [0101].) For example, the Morris-SIP system discloses that "an empty TCP packet may be received **while** a potential idle time period is being monitored." (*Id.*) The Morris-SIP system further discloses that in response to

receiving an empty packet (that must have been sent by another device, e.g., the second node 604), the timer associated with the negotiated ITP may or may not be reset "based on whether or not a received empty TCP packet matches a specified condition." (EX1005, [0102].) Thus, if the empty packet does not match a specified condition, it is an empty packet that is sent without causing the timer associated with the negotiated ITP to be reset, as claimed.[20]

j) wherein the apparatus is configured such that:

127. As discussed below, the apparatus in the Morris-SIP system is configured as set out in claim elements 1.j.(1)-(3).

(1) the first time period and the second time period are started in response to at least one of a received packet or a sent packet,

128. The Morris-SIP combination discloses or suggests this limitation. For example, the apparatus in the Morris-SIP system is configured such that the preliminary ITP ("first time period") and the negotiated ITP ("second duration") are

[20] Claim element 1.i does not specify the sender of the empty packet. In any event, even if the first node sent such an empty packet, it would not have caused the timer associated with the negotiated ITP to be reset since such an empty packet would, e.g., reset another timer such as the keep-alive timer. (EX1005, [0086].)

started in response to at least a sent packet. (*See supra* Sections IX.A.1.d, IX.A.1.g-h.)[21]

129. In the Morris-SIP system, the preliminary ITP information ("first time period") would have been used to configure a user-timeout type of timeout similar to Morris, i.e., SIP's Timer B such that it would have been initiated upon sending of the initial connection setup packet in Morris' TCP-based connection establishment procedure. (*Id.*; EX1005 ¶¶[0069], [0076], [0092], [0120].) The system would then wait for a response to the sent packet and the timeout would expire if no response is received. (*See supra* Section IX.A.1.g.)

130. Similarly, in the Morris-SIP system, the negotiated ITP ("second time period") would have been used to configure a user-timeout type of timeout similar to Morris, i.e., SIP's Timer B such that it would have been initiated upon sending of an ACK-eliciting packet. (*See supra* Sections IX.A.1.d, IX.A.1.h; EX1005 ¶¶[0083], [0088], [0092], [0096]-[0097], [0120], FIG. 2 (steps 206 and 208), FIG. 7 (elements 706.n and 710.1).) The system would then wait for a response to the sent packet and the timeout would expire if no response is received. (*See supra* Section IX.A.1.)

(2) during the at least portion of the first connection including the at least portion of the

[21] *See also* n.16.

first connection set up, the detection of the first time period of the first type of time period is based on the first duration, by being based on whether, during the first duration after the first time period is started, no packet is received and processed to keep the first connection active, and

131. The Morris-SIP combination discloses or suggests this limitation in at least three ways.

132. **First**, as explained, in the Morris-SIP system, during at least the portion of the first connection setup, the detection of the preliminary ITP ("first time period") of the first type of time period is based on the preliminary ITP information ("first duration"). (*See supra* Section IX.A.1.g; *see also supra* Section IX.A.1.d (discussing "first type of time period" in Morris-SIP combination).) Such detection of the preliminary ITP ("first time period") would have been based on whether, for the duration indicated by the preliminary ITP information, no ACK packet acknowledging a previously sent data packet is received and processed to keep the connection active (e.g., after being received, the sequence number indicated in the ACK packet is processed by checking to see if it covers the sequence number included in the previously sent data segment). (EX1005, [0088]-[0089], [0092]-[0093], [0095]-[0097], [0106]-[0109]; EX1008, 28-29, 45; E.) A POSITA would have been motivated to implement the Morris-SIP system this way since, as discussed above, the "first type of time period" would have functioned similarly in

the system both during the connection setup and after the connection is set up (i.e., during an established connection). (*See supra* Section IX.A.1.d.) Indeed, as discussed above and below, Morris already implements use of ITP during the established connection so a POSITA would have been motivated to apply a similar implementation during the connection setup.

133. **Second**, a POSITA would have been motivated to further configure the Morris-SIP system such that the first and second time periods of the first type of time period are detected based on the absence of a received packet type that is not limited to a received ACK packet type. (*See supra* Section IX.A.4.c.2.) Thus, under this theory, claim element 1.j.2 (and element 1.j.3 below) is met such that the detection of the first time period is based on whether no packet of any kind is received and processed to keep the connection active. In particular, while the analysis of claim elements 1.g-h relied on the absence of at least one acknowledgment packet (ACK packet) received in response to a previously sent data segment, a POSITA would have been motivated to configure the Morris-SIP system such that the first/second time periods would have been detected based on an absence of any received packet type. (*Id.*) By configuring the first and second timeouts to be detected based on the absence of additional packet types (e.g., a received data packet or keep-alive packet or a received retransmission packet), a POSITA would have recognized that the Morris-SIP system would have been allowed to survive intermittent periods of

disconnection or other network connectivity issues where the acknowledgment packet is not received but other packets may be received. (EX1018, 1-8 (describing negotiation of abort timeout value to allow nodes to survive periods of disconnection); *see also* Section IX.A.1.d.)

134. Put another way, in the Morris-SIP system, if the first node 602 (which sends the initial connection setup packet to second node 604) detects any activity in the connection based on various received packet types (not limited to a received ACK packet), the first/second time periods would not be detected. Any packet received by the first node 602 from second node 604 would indicate to the first node 602 that packets are being communicated over the communication link between the two nodes. Accordingly, the first node 602 could continue waiting for an acknowledgment for the data packet it sent, given it is aware that the connection is not entirely idle (due to the presence of the other packet types it detected). A POSITA would have had a reasonable expectation of success in configuring the Morris-SIP system this way as Morris already discloses other timeout types like the keep-alive timeout which is configured in a similar way (e.g., to generally detect idleness in a connection based on detecting different packet types). (EX1005, [0002]-[0003], [0069], [0085]-[0086], [0120], claims 2, 13.)

135. **Third**, this limitation is taught by the Morris-SIP system during a disconnection scenario. For example, when an interruption in communication

occurs, such as a disconnection in the network service or in a physical network medium (EX1005, [0004]-[0005], [0062]), packets from one node do not reach the other node in either direction. Thus, when the first node 602 and/or the second node 604 are experiencing connectivity issues such that one or both of them are disconnected, the absence of packets would have included an absence of all packet types (and not limited to a received ACK packet type) and such an absence of packets would have led to detection of the first/second time periods (assuming any other conditions specified in claim elements 1.g-h are met). (EX1005, [0004]-[0005], [0062].) As an exemplary operational scenario, the Morris-SIP system would have been configured such that the first node 602 would have detected the first/second time periods based on the first/second durations, respectively, along with an absence of any received packet type (e.g., a data packet, a keep-alive packet, a retransmission packet, or an acknowledgment packet), since none of these packets could have been received by the first node 602 if it was disconnected from the network. (EX1005, [0004]-[0005], [0010], [0062], [0106], [0108]-[0109] (discussing deactivation of connection upon ITP detection).)

136. If such a disconnection scenario occurred during a portion of the first connection setup, then detection of the preliminary ITP ("first time period") would have been based on whether, for the duration indicated by the preliminary ITP

information, no packet of any kind is received and processed to keep the connection active.

(3) after the first duration is changed to the second duration, the detection of the second time period of the first type of time period is based on the second duration, by being based on whether, during the second duration after the second time period is started, no packet is received and processed to keep the first connection active;

137. The Morris-SIP combination discloses or suggests this limitation in at least three ways, similar to those discussed above for claim element 1.j.(2). (*See supra* Section IX.A.1.j.(2).)

138. **First**, as explained, in the Morris-SIP system, during the first connection (e.g., after the connection is established and the first duration is changed to the second duration), the detection of the negotiated ITP ("second time period") is based on the negotiable ITP information ("second duration"). (*See supra* Section IX.A.1.h.) Such detection of the negotiated ITP ("second time period") would have been based on whether, for the duration indicated by the negotiable ITP information, no ACK packet acknowledging a previously sent data packet is received and processed to keep the connection active (e.g., after being received, the sequence number indicated in the ACK packet is processed by checking to see if it covers the

sequence number included in the previously sent data segment). (EX1005, [0088]-[0089], [0092]-[0093], [0095]-[0097], [0106]-[0109]; EX1008, 28-29, 45.)

139. **Second**, as explained for claim element 1.j.(2), a POSITA would have been motivated to further configure the Morris-SIP system such that the first and second time periods of the first type of time period are detected based on the absence of a received packet type that is not limited to a received ACK packet type. (*See supra* Section IX.A.1.j.(2).) Thus, under this theory, claim element 1.j.(3) is met such that the detection of the second time period is based on whether no packet of any kind is received and processed to keep the connection active. (*Id.*.)

140. **Third**, this limitation is taught by the Morris-SIP system during a disconnection scenario such as the one discussed above for claim element 1.j.2. (*See supra* Section IX.A.1.j.(2).) As explained, during a disconnection (e.g., when first node 602 is detected from the network such that it can no longer communicate with second node 604), no packet of any kind would have been received by first node 602. (*Id.*) If such a disconnection scenario occurred during the first connection, then detection of the negotiated ITP ("second time period") would have been based on whether, for the duration indicated by the negotiable ITP information, no packet of any kind is received and processed to keep the connection active. (*Id.*.)

k) wherein the apparatus is configured such that the first connection set up further includes, in addition to (a-1) sending the first packet:

(1) (a-2) after sending the first packet and without any other communication between the first node and the second node after (a-1) and before (a-2), receiving, at the first node from the second node, a second packet, and

(2) (a-3) after receiving the second packet and without any other communication between the first node and the second node after (a-2) and before (a-3), sending, from the first node to the second node, a third packet.

141. Regarding limitations 1.k.(1)-(2), the Morris-SIP combination discloses performing a three-way handshake to set up the connection between the first node 602 and the second node 604, as discussed above for limitation 1.f (discussing the first node 602 (a-1) sending the "first packet" as part of the three-way handshake). (*See supra* Section IX.A.1.f; *see also supra* Section IX.A.1.e (discussing the generation of the first packet for use in the three-way handshake).) Both Morris and SIP describe that such a three-way handshake includes, after sending the first packet, (a-2) receiving at the first node 602 from the second node 604, a responsive packet ("second packet") to the first packet as required by limitation 1.k.(1). (EX1005, [0076] (describing the ITP header being exchanged in the RFC 793 three-way handshake); EX1008, 27-28 (RFC 793 describing its three-way handshake involving a second packet including an ACK and a SYN in steps 2-

3); EX1006, 16 (describing "the INVITE/**200**/ACK three-way handshake used to establish SIP sessions"), 125 (describing a second packet in the form of one or more responses to the INVITE message).) Morris and SIP do not disclose any other communication between the first node 602 and the second node after (a-1) and before (a-2) during the connection setup procedure. (EX1005, [0076]; EX1008, 27-28, 44 (RFC 793 disclosing that data cannot be exchanged between nodes until after connection establishment); EX1006, 16, 125.)

142. And (a-3) after receiving the second packet, both Morris and SIP disclose sending, from the first node 602 to the second node 604, an acknowledgment packet ("third packet"). (EX1005, [0076]; EX1008, 27-28 (RFC 793 describing its three-way handshake involving a third packet including an ACK in step 4); EX1006, 16 (describing "the INVITE/200/**ACK** three-way handshake used to establish SIP sessions"), 125 (describing a third packet in the form of an ACK message).) Morris and SIP do not disclose any other communication between the first node 602 and the second node after (a-2) and before (a-3) during the connection setup procedure. (EX1005, [0076]; EX1008, 27-28, 44; EX1006, 16, 125.)

2. Claim 6

a) The apparatus of claim 1, wherein the apparatus is configured such that the first duration is used and the second duration is not used, for detecting the first type

of time period until being changed to the second duration.

143. The Morris-SIP combination discloses or suggests this for at least the reasons given above with respect to claim elements 1.d, 1.g, and 1.h. (*Supra* Sections IX.A.1.d, IX.A.1.g-h.) As discussed above, the apparatus is configured such that the first node uses the preliminary ITP information ("first duration") to detect a preliminary ITP ("first time period") of the user-timeout type of timeout similar to Morris, i.e., SIP's Timer B ("first type of time period") until the preliminary ITP information is updated ("changed") to the negotiable ITP information ("second duration"). (*See* Sections IX.A.1.d, IX.A.1.g.) As discussed, after the first connection setup occurs (i.e., after the three-way handshake), and after the preliminary ITP information ("first duration") is changed to the negotiable ITP information ("second duration"), the first node uses the negotiable ITP information ("second duration") to detect a negotiated ITP ("second time period") also of the user-timeout type of timeout similar to Morris, i.e., SIP's Timer B ("first type of time period"). (*See* Section IX.A.1.h.)

3. Claims 12, 14

a) **12: The apparatus of claim 1, wherein the apparatus is configured such that the second duration is capable of being different from the first duration.**

b) **14: The apparatus of claim 12, wherein the apparatus is configured such that the second duration is the same as the first duration, despite being capable of being different from the first duration.**

144. The Morris-SIP combination discloses or suggests the limitations of claims 12 and 14. As discussed above, the preliminary ITP information is set based on, e.g., the round-trip time or a saved parameter from a previous connection. (*See supra* Section IX.A.1.d.) The negotiable ITP information ("second duration") is determined using the first metadata included in the first packet sent from the apparatus to the node. (*See supra* Sections IX.A.1.e-f.) Indeed, Morris teaches that "a duration of an [ITP] **may be static and/or may be dynamic**." (EX1005, [0095].) Thus, the second duration is "capable" of being different from (or the same as) the first duration.

145. For instance, the apparatus in the Morris-SIP system may identify first data in the first packet that reflects a new round-trip-time between the apparatus and the node than the previously calculated one or may determine to use the same round-trip-time as before. (*See supra* Section IX.A.1.d.) Similarly, the apparatus in the Morris-SIP system may decide to propose the same or a different second duration than the first duration based on other attributes recognized by Morris such as "a

measure of network latency, a measure of network congestion, an indication of the availability of a particular resource, a user specified attribute, a security attribute, an energy usage attribute, a user attribute such as role of the user, and/or a measure of bandwidth supported by NIC 508 and/or a physical network medium operatively coupled to NIC 508." (EX1005, [0062].)

146. Morris further discloses that the exchange of the ITP information in the packets between the first node 602 and second node 604 "may simply be informational" (as opposed to a negotiation). (*Id.*, [0105].) In this scenario, the first node 602 (apparatus) would have the ability to propose a new value for the second duration but may determine to keep the same value for the second duration as the first duration, and then share that same value with the second node 604 for informational purposes (so second node 604 is aware of the time period being used by first node 602 that would lead to potential connection closure). A POSITA would have further been motivated to set second duration the same as the first duration (despite having the capability to change it) in certain scenarios such as reconnection between nodes after a first connection between the nodes dropped momentarily. (EX1083, Abstract, [0013], [0026], [0034], [0039].) In such a scenario, the nodes may agree to use the same duration agreed upon before if, e.g., the connection environment has not changed and/or the nodes wish to avoid another round of

negotiation which would consume further network resources. (*See supra* Section IX.A.1.d.)

147. Thus, the Morris-SIP combined system discloses or suggests an apparatus that is configured such that the negotiable ITP information ("second duration") is the same as the preliminary ITP information ("first duration"), despite being capable of being different from the first duration.

4. Claim 17

a) The apparatus of claim 1, wherein the apparatus is configured such that the first type of time period is started in response to the received packet.

148. The Morris-SIP combination discloses or suggests this limitation. As discussed above, the Morris-SIP combination teaches a user-timeout type of timeout similar to Morris, i.e., SIP's Timer B, which is a first type of time period characterized by, e.g., starting in response to the sending of a response-eliciting packet (like Morris's ACK-eliciting packets or SIP's Invite message). (*See supra* Sections IX.A.1.d, IX.A.1.g-h, IX.A.1.j.(1) (explaining how the preliminary ITP and the negotiated ITP, which are both of a "first type of time period," are started in response to at least a sent packet).)

149. The Morris-SIP combination further teaches that the apparatus is configured such that "first type of time period" is started in response to a received packet, as recited in claim 17. For example, Morris discloses that in addition to

starting an ITP in response to a sent packet, the ITP can be started in response to a received packet.

> In FIG. 5, ITP policy component 550 and ITP monitor component 556 may operate to **reset and/or initiate detection of an idle time period in response to receiving an empty TCP packet**. First node 602 may receive an empty packet. In response, ITP monitor component 556 may receive an event and/or other indication to reset detection of an idle time period. Resetting the detecting process may be based on whether or not a received empty TCP packet matches a specified condition. ITP option handler component 562 may be configured to determine whether a received empty TCP packet matches the condition. If ITP option handler component 562 determines the empty packet matches the condition, ITP monitor component 556 may be instructed to **reset and/or restart detection of the first idle time period including detecting the beginning of a next potential time period**.

(EX1005 [0102]; *see also id.* [0101] ("Receiving a packet from second node 604 included in the TCP connection is an event that, in various aspects, may directly and/or indirectly indicate the beginning of a potential idle time period…"); *see also* EX1006, 134-137 (SIP disclosing an Invite Server Transaction corresponding to its

ICT, where in response to the server receiving the INVITE message from the client, the server sends a response along with starting a timer H), FIG. 7.)

150. A POSITA would have been motivated to further configure the Morris-SIP system to restart the time period detection in response to receiving a packet. For example, a POSITA would have understood that in a system such as the Morris-SIP system which may include the keep-alive mechanism, the apparatus/first node of claim 1 would have been expected to receive keep-alive packets from the second node from time to time (e.g., based on a keep-alive period). (*See supra* Section IX.A.1.j.(2), *infra* Section IX.A.7.) In this case, a POSITA would have been motivated to configure the Morris-SIP system such that it would restart the "first type of time period" upon receipt of a received packet, to start counting down a certain amount of time until another packet is received (such a time period would have been based on, e.g., a multiple of the keep-alive period to allow for the second node to send or attempt to send a keep-alive packet two or more times). (EX1009, 101.) Upon expiration of the time period where no further packets are received (e.g., when the system detects expiration of the "first time period" or "second time period" which are both of the "first type of time period" in elements 1.g-h), the system would have determined that the second node is no longer active and thus partially closed the connection. A POSITA would have had a reasonable expectation of success in modifying the Morris-SIP system this way since Morris itself teaches that the ITP

can be restarted upon receipt of a packet and based on the knowledge of a POSITA discussed herein. Accordingly, the Morris-SIP combination teaches a system configured such that the first type of time period is started in response to the received packet, as claimed.

5. Claim 20

a) The apparatus of claim 1, wherein the apparatus is configured such that the first connection is not a Transmission Control Protocol (TCP) extension connection, and the first connection is not a TCP extension connection that involves a TCP extension.

151. The Morris-SIP combination discloses or suggests this limitation. As an initial matter, the Morris-SIP combination teaches that the first connection is not a transmission control protocol (TCP) for at least two reasons.

152. **First**, the combined Morris-SIP system implements a protocol that is not TCP by inclusion of a mechanism to detect inactivity during connection setup in the Morris system based on SIP's teachings, as explained above with respect to claim element 1.d. (*Supra* Section IX.A.1.d.) This is because TCP (as defined by RFC 793) does not disclose the option (e.g., by using the options field in a TCP header) to introduce the mechanism taught by the Morris-SIP system. (EX1005, [0002], [0051], [0078], [0109]; *see generally* EX1008; *see also infra* Section X.A.1.) Thus, the Morris-SIP system implements a protocol that is not TCP for establishing a connection. (*See generally* EX1005; EX1008; *see also infra* Section X.) For

example, RFC 793 does not include the transmission of a setup packet (e.g., like the "INVITE" message in SIP) during connection setup that, upon being sent or received, would have initiated the mechanism (e.g., a timer) to detect inactivity during the connection setup portion (including the possibility of the in-progress connection to time out during the connection setup portion) as taught by the Morris-SIP system.

153. Indeed, a POSITA would have understood that a TCP extension typically refers to the use of TCP options in a TCP header. (EX1005, [0080] (describing a TCP extension in the context of TCP options in a TCP packet); EX1090, 7.) On the other hand, the Morris-SIP system differs from TCP by including a mechanism to detect inactivity during connection setup, regardless of whether such a mechanism is implemented using TCP options (although Morris or RFC 793 do not disclose this possibility). Thus, the first connection in the Morris-SIP system is not a TCP extension connection involving a TCP extension.

154. **Second**, the combined Morris-SIP system implements a protocol that is not TCP based on PO's understanding of the claims. According to PO, the "the first connection is not a TCP extension connection that involves a TCP extension" in claim 20 encompasses a TCP-variant protocol. In addition, I have been informed that in patents related to Morris which share substantially the same specification as Morris, Patent Owner has characterized and claimed Morris' above-noted TCP-

based disclosures in the context of a TCP-variant protocol using TCP-variant packets in a TCP-variant connection. (*See, e.g.*, EX1037, claim 1; EX1036, claim 1; EX1087, claim 78.) In addition, I have been informed that the Patent Owner contends that its TCP-variant protocol encompasses a UDP-based protocol, which is not a TCP-based protocol. Thus, in light of Patent Owner's views including Patent Owner's understanding that Morris teaches TCP-variant protocols, to the extent the Morris-SIP combination is found to encompass a TCP-variant protocol, it is my opinion that the combination still teaches that the apparatus is configured such that the first connection is not a TCP extension connection, and the first connection is not a TCP extension connection that involves a TCP extension.

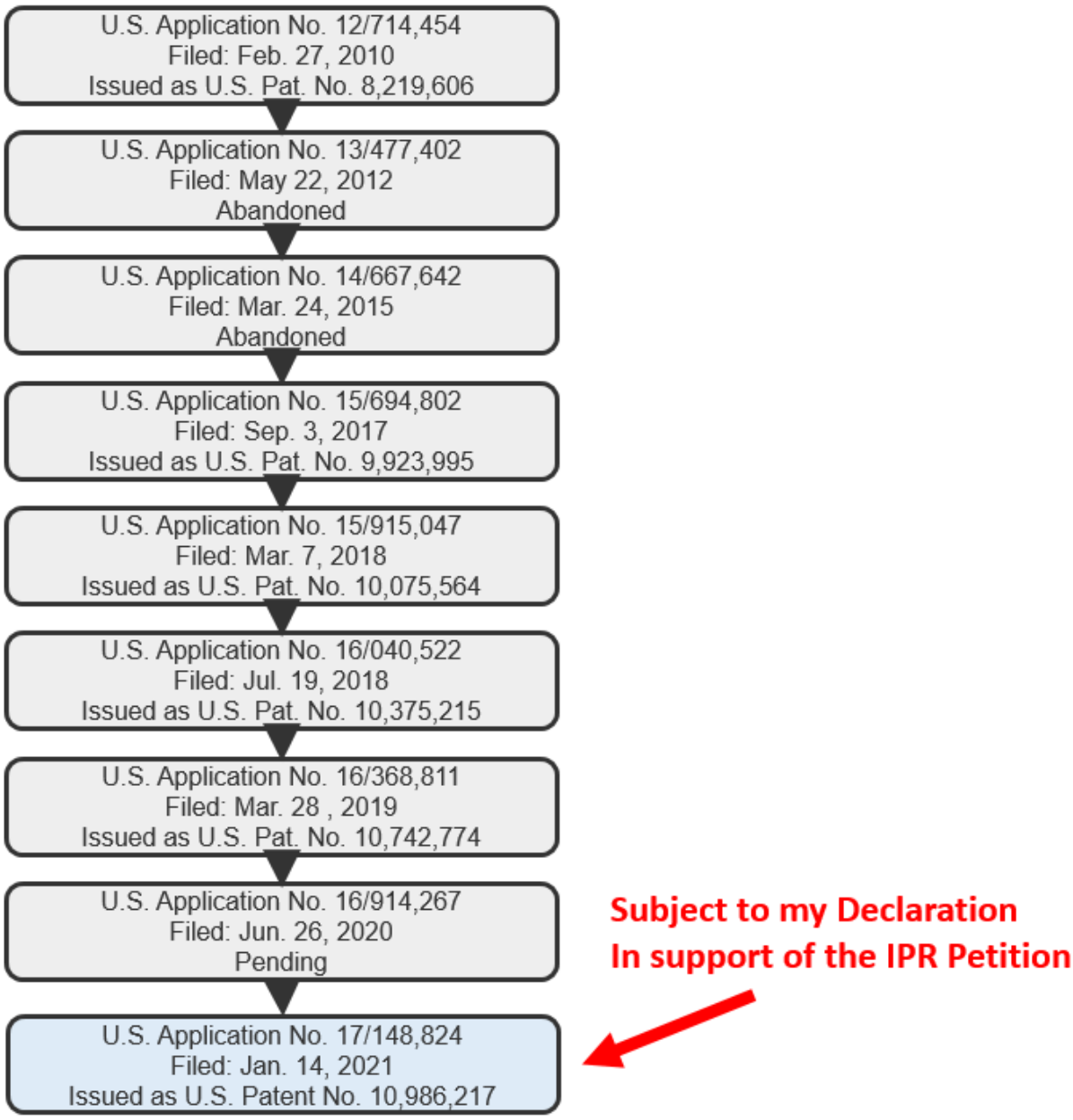
155. For example, in light of PO's suggestion that a non-TCP protocol includes a TCP-variant protocol and PO's understanding that Morris teaches TCP-variant protocols, the apparatus of the Morris-SIP combination is configured such that it implements a non-TCP protocol, resulting in the nodes establishing a connection that is different from TCP using packets that are different from TCP packets. Thus, a connection that is established based on the Morris-SIP combination, which involves detection of a timeout during the connection setup (based on the first duration) and a change of the timeout value after connection setup (based on the second duration) is not a TCP extension connection involving a TCP

extension, as claimed. (*Id.*) Morris and RFC 793 do not describe any such TCP extension connection or TCP extension.

X. DISCLOSURE OF CERTAIN CLAIM LIMITATONS IN THE '217 PATENT AND ITS PARENT APPLICATIONS/PATENTS

156. In my opinion, a person of skill in the art would not have understood that the named inventor of the '217 patent possessed an invention with all of the features recited in or encompassed by claim 1 at the time of the alleged invention.

157. The '217 patent relates to several patents and/or patent applications, as shown in the purported priority chain below:



158. However, as I discussed below, the priority applications do not convey to a person of ordinary skill in the art that the named inventor was in possession of at least two claim features.

A. Claim 1: No Support for Non-TCP Features Encompassed by "First Connection"

159. Independent claim 1 of the ’217 patent recites:

> set up **a first connection**, by sending, from a first node to a second node, the first packet…

(EX1001, 21:40-44.)

160. Independent claim 1 broadly recites features relating to setting up a first connection (e.g., between the first and second node) without specifying the type of connection. For example, the scope of the “first connection” term encompasses setting up a TCP connection or a non-TCP connection, e.g., setting up a connection using a protocol that does not utilize TCP and using packets that are not TCP packets (generally referred to herein as the “non-TCP” features).

161. This understanding is confirmed by other claims in the ’217 patent such as claim 20. For example, claim 20 recites that “the first connection is not a Transmission Control Protocol (TCP) extension connection, and the first connection is not a TCP extension connection that involves a TCP extension.” (EX1001, 23:42-46.) Claim 20 confirms that the “first connection” in claim 1 is not limited to TCP-based connections. I have been informed that the Patent Owner contends that the non-TCP features, as encompassed by the scope of claim 1, encompass protocols that do not utilize TCP at all such as UDP-based protocols. As I have explained above, UDP relates to the User Datagram Protocol (*see* my discussion above in Section V.A.4.c), and UDP is specified in RFC 768. (Ex. 1060, 1.) It is well known

in the art that UDP does not utilize TCP at all. For example, UDP does not utilize a TCP connection or TCP packets. (*See* my discussion above in Section V.A.4.c; *see generally* Ex. 1060.)

162. In my opinion, a person of ordinary skill in the art would not have understood that the inventor of the ’217 patent had possession of the claimed invention, including the non-TCP features encompassed by the scope of claim 1 at the time the ’824 application was filed because the as-filed specification of the ’824 application and the specifications of the ’217 patent’s priority applications (shown below) do not support these features.

1. The ’824 Application and Its Priority Applications Are Limited to TCP-Based Embodiments

163. The ’824 application and its priority applications (except for the ’642 application) share substantially the same as-filed specifications.[22] Exhibits 1040,

[22] The ’642 application is a continuation-in-part of the ’402 application. I have reviewed the ’642 application. In my opinion, a person of ordinary skill in the art reviewing the ’642 application would have determined that the ’642 application has little in common with the subject matter of the ’855 patent family concerning sharing of information for detecting an idle TCP connection. Based on my review, it is my opinion that the specification of the ’642 application was significantly amended such

1050-1054, and 1077 are computer-generated redlines comparing the as-filed specifications of these priority applications including the ’824 application (EX1004, EXS1038-1039, EXS1044-1048). The computer-generated redlines show the similarities between the compared documents in BLACK and differences in BLUE for what is only present in the later-filed application and RED for what is only present in the earlier-filed application. As shown in exhibits 1040, 1050-1054, and 1077, other than the changes to the summary sections in each of the priority

---

that it was directed to “selecting a resource based on a processing cost,” and the disclosures regarding the sharing of information for detecting an idle TCP connection were removed. (Ex. 1043, 108 (title), 110–111 (¶¶[0005]–[0006]); Ex. 1049 (computer-generated redline showing the similarities between the ’402 and ’642 applications in BLACK and differences (BLUE for what is only present in the ’642 application and RED for what is only present in the ’402 application)), 1 (title), 3 (¶¶[0005]–[0006]).) While the ’642 application claims priority to (or is related) to a number of U.S. patent applications (Ex. 1043, 108–110 (¶¶[0001]–[0004]), 162–164 (¶[0128])), none of these applications (Exs. 1038–1039, 1041, 1062–1074) describe any non-TCP features at-issue here.

applications (which were updated to reflect the claims being presented for examination), the specifications of the applications are substantially similar.[23]

164. The disclosures of these priority applications (including the '824 application), however, do not describe any non-TCP-based features (e.g., the use of a non-TCP-based protocol to set up a non-TCP connection using non-TCP packets)—the disclosures are limited to using the options field in a TCP header defined by RFC 793. Instead, as discussed below, these disclosures only describe embodiments based on TCP. The discussion below primarily focuses on how the '454 application does not convey to a person of ordinary skill in the art that the named inventor had possession of the non-TCP features as it is the earliest application in the priority chain. The remaining applications in the priority chain are similarly defective given their specifications are substantially similar.

165. With regard to the '454 application, the application uses the term "TCP" nearly 400 times, including, for example, in the specification (310 times),

[23] The only change to the detailed description portion of the '802 application specification, relative to the originally filed '454 application, is the addition of a brief explanation of the well-known TCP three-way handshake already described in RFC 793 decades earlier. (EX1050, 15 (¶¶[0082]-[0087]); EX1008 (RFC 793), 31-32.)

claims (72 times), abstract (3 times), and figures (14 times). (*See generally* EX1038 ('454 application), 99-106, 111-118, 120-165.) Based on my review, not once does the '454 application refer to a protocol that is not TCP (e.g., non-TCP) or a non-TCP connection/packet. (*Id.*)

166. As an example, both the Abstract and "Summary" of the as-filed '454 application describes the disclosed invention as relating to a "**TCP connection**" and "**TCP packet**" (as opposed to a **non-TCP** connection or packet required by claim 1):

> Methods and systems are described for sharing information for detecting an idle **TCP connection**. In one aspect, a method includes receiving, by a second node from a first node, a first **transmission control protocol (TCP) packet** in a **TCP connection**. The method further includes … identifying metadata for a first idle time period, detectable by the first node, during which no **TCP packet** including data in a first TCP data stream sent in the **TCP connection** by the second node is received by the first node. The method still further includes

> modifying … a timeout attribute associated with the **TCP connection**.

(EX1038, 122 (¶[0008]), 99 (Abstract) (describing same); *see also id.*, 122-124 (¶¶[0009]-[0011] (describing additional aspects in the summary section in context of a "TCP packet" and "TCP connection")).)

167. Additionally, the detailed description in the '454 application also focuses on a "**TCP connection**." (EX1038, 125-126 (¶¶[0013]-[0020]).) For example, with reference to figures 2-4b, the '454 application explains:

> FIG. 2 is a flow diagram illustrating a first method for sharing information for detecting an idle **TCP connection** according to an exemplary aspect of the subject matter described herein. FIG. 3 is a flow diagram illustrating a second method for sharing information for detecting an idle **TCP connection** according to an exemplary aspect of the subject matter described herein. FIG. 4a is a block diagram illustrating a system for sharing information for detecting an idle **TCP connection** according to the first method in FIG. 2. FIG. 4b is a block diagram illustrating a system for sharing information for detecting an idle **TCP connection** according to the second method in FIG. 3.

(*Id.*, 131-132 (¶[0035]); *see also, e.g., id.* (133-137 (¶¶[0042]-[0051] (describing in connection with figure 5, "TCP packets" and "TCP connection" and "TCP layer" features), 141 (¶[0064] (discussing a "TCP connection" as one that "may be

identified by its endpoints" and "may include an endpoint of the TCP connection"), 143 (¶[0069] (discussing a "TCP keep-alive option, a TCP user timeout" and "another timeout associated with a TCP connection").)

168. Based on my review, the lack of support for the non-TCP features encompassed by claim 1 is exemplified by claim 20's language requiring setup of a "first connection" that is "**not a Transmission Control Protocol (TCP) extension connection**" and "**not a TCP extension connection that involves a TCP extension**" while the '454 application's disclosure is limited to describing setup of a TCP connection using TCP packets. In particular, the '454 application only discloses setting up a **TCP connection** (i) using the three-way handshake described in RFC 793 or (ii) in other unspecified ways:

> In an aspect, an ITP header may be included in a packet exchanged during setup of a TCP connection. **RFC 793 describes a "three-way handshake" for establishing a TCP connection. Other message exchanges may be used in setting up a TCP connection as those skilled in the art will understand. Such other exchanges are not currently supported by the TCP as described in RFC 793.** The specified "three-way handshake" and other patterns of message exchange for setting up a TCP connection include packets that are

> considered to be in the TCP connection for purposes of this disclosure.

(EX1038, 147 (¶[0076]).)

169. Based on my review, nowhere does the ’454 application describe setting up a connection in a protocol that is not TCP (e.g., a UDP-based protocol which does not utilize TCP at all). (*See generally* EX1038.)

170. In my opinion, a person of ordinary skill in the art would not have understood that the disclosures in the ’454 application limited to TCP embodiments would have been interchangeable with non-TCP embodiments. For example, a person of ordinary skill in the art would have understood that a protocol that does not utilize TCP at all (like the UDP-based protocol) would operate in a different environment with its own set of rules regarding at least (i) the setup of a connection between two nodes in a network, (ii) how the devices would communicate with each other, (iii) the packet structure used for communication, and (iv) detection of timeouts that would cause the connection to close. (EX1060, 1, EX1027, 1, EX1061, 1.) However, based on my review, the ’454 application does not provide any explanation regarding how its disclosed techniques for sharing idle time information in a TCP options field of a TCP packet during setup of a TCP connection would have extended to an altogether different environment with its own rules regarding connection setup, connection operation, packet structure, and timeout detection.

171. For example, nowhere does the ’454 application describe (or inform a POSITA) how its disclosures limited to TCP embodiments may be extended to the non-TCP features encompassed by claim 1 like setting up a non-TCP connection (e.g., using a protocols for establishing a UDP connection). Thus, the ’454 application does not convey to a person of ordinary skill in the art that the named inventor had possession of the non-TCP features encompassed by claim 1.

172. Indeed, even the discussion in the reproduced portion above regarding “[o]ther message exchanges may be used in setting up a TCP connection as those skilled in the art will understand” and “[s]uch other exchanges are not currently supported by the TCP as described in RFC 793” (EX1038, 147 (¶[0076]) would have been understood to only disclose a TCP embodiment as this portion states that such exchanges would still “be used in setting up a TCP connection” (*id.*). Moreover, this portion also fails to disclose the other requirements with regards to the set up packet.

173. While the ’454 application discusses that “[a]n equivalent or analog of an ITP header may be included in a footer of a protocol packet in **an extension and/or variant of the current TCP**” (EX1038, 147 (¶[0080]), such extensions or variants are mentioned solely within the context of a TCP packet:

> Those skilled in the art will recognize given this disclosure that an **ITP header** may have other suitable formats and **may be included in a TCP packet in structures and locations other**

> **than those specified for TCP options in RFC 793**. An equivalent or analog of an ITP header may be included in a footer of a **protocol packet in an extension and/or variant of the current TCP**.

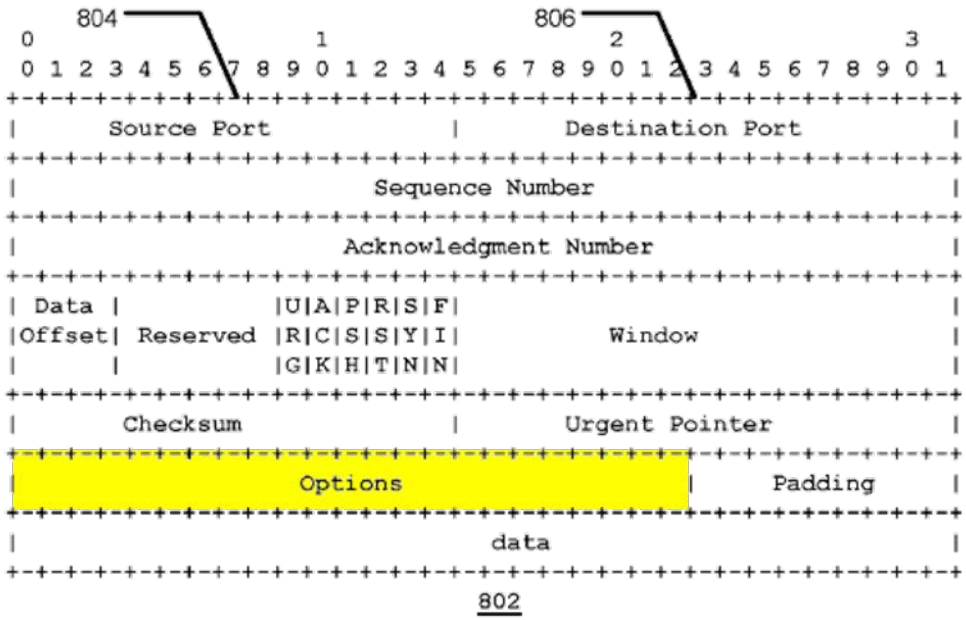
(*Id*., 147 (¶[0080]).)

174. Even if a TCP extension or variant as vaguely described here is encompassed by the non-TCP features, the scope of the non-TCP features encompasses protocols that do not utilize TCP at all (e.g., a UDP-based protocol), as discussed above. Thus, the scant description of a TCP extension or variant in the priority applications, even if considered to be within the scope of non-TCP, is not sufficient to convey to a person of ordinary skill in the art that the named inventor had possession of these features because the full scope of the non-TCP features is not supported.[24]

---

[24] As I noted above, I understand that Patent Owner's interpretation of the non-TCP features encompasses UDP-based protocols. Such an interpretation of non-TCP would encompass various other protocols (e.g., protocols based on XTP or IPX/SPX along with UDP-based protocols), none of which are described in the '217 patent family disclosures, much less in the context of the features set forth in the '217 patent claims. (*See, e.g.*, Ex. 1016, 1, 7–8 (describing the XTP protocol); *see also* my discussion above in Section V.A.4.b (discussing the IPX/SPX protocols).)

175. In fact, to the extent any embodiment is disclosed in relation to the above features, it relates to using a TCP options field in a **conventional TCP packet structure provided by RFC 793** to share idle time information between nodes. (EX1038, 147 (¶¶[0076]-[0079]).) Figure 8 below, "adapted from RFC 793," illustrates this configuration. (*Id.*, FIG. 8, ¶¶[0078]-[0079].)



(*Id.*, FIG. 8 (cropped and annotated); *see also* EX1008, FIG. 3.)

176. In my opinion, this disclosed extension or variant of "the current TCP" in the '454 application does not describe any **non-TCP** functionality, especially when viewed in the context of surrounding language in the specification that, as explained above, is entirely focused on TCP-based features without any discussion of non-TCP-based features. (*See also supra* IX.A.2.f.1, IX.A.4.f.1.)

177. Moreover, based on my review, none of the applications in the priority chain describe any embodiment that shows a connection between nodes that is setup using a protocol that is not TCP along with the other claim features (e.g., time period detection causing partial deactivation of the connection or performance of other actions) associated with such a non-TCP protocol. For example, there is no description of setting up a connection that does not utilize TCP (e.g., a UDP-based connection) using packets not specified by TCP (e.g., UDP-based packets) to generate a setup packet for such a UDP-based connection, set up the UDP-based connection, and detect time periods in the UDP-based connection. (*Id.*) Instead, as noted above, the '454 application's disclosure is limited to TCP-based embodiments including setting up the connection using TCP's three-way handshake (as specified in RFC 793) and employing TCP-based packets to share idle time information for a TCP-based connection. (*See generally* EX1038.) And as I discussed above, the TCP-based teachings of the '454 application cannot be extended to non-TCP protocols.

178. Thus, for the reasons as I discussed above, the '454 application and the other applications in the priority chain sharing substantially similar specifications do not provide disclosure of the non-TCP features encompassed by the "first connection" of claim 1 of the '217 patent.

2. Materials Incorporated by Reference in the Priority Applications Do Not Support the Non-TCP Features

179. The priority applications purport to incorporate U.S. Application No. 12/714,063 ("the '063 application") (EX1041) and RFC 793 (EX1008). (*See, e.g.*., EX1038 ('454 application), 120 (¶¶[0001]-[0002]), 90 (IDS citing RFC 793).) Based on my review, however, as with the priority applications, neither the '063 application nor RFC 793 discloses or mentions the non-TCP features encompassed by claim 1 of the '217 patent.

180. The '063 application shares a similar specification to the '454 application and only describes "TCP connections" and "TCP packets" (including extensions and variants in a similar fashion as the '454 application). (*See generally* EX1041, 137-201; *see also* EX1042 (computer-generated redline comparing the '063 and '454 applications).)

181. RFC 793 dates back to 1981 (EX1008, 1) and defined TCP at that time (*id.*). Indeed, RFC 793 provides:

> This document describes the DoD ***Standard Transmission Control Protocol (TCP)***. There have been nine earlier editions

> of the ARPA TCP specification on which this standard is based, and the present text draws heavily from them.

(EX1008, 4.) RFC 793 likewise does not describe the non-TCP features.[25]

182. Thus, even if the material disclosed in RFC 793 and/or the '063 application is incorporated into priority application disclosures, in my opinion, none of the material incorporated by reference supports the non-TCP features encompassed by claim 1.

3. Mere Introduction of Non-TCP Type Terms In Summaries and Claims of Some Priority Applications Does Not Support the Non-TCP Features

183. Terms such as "non-TCP," "different from TCP," or "separate from TCP" started appearing in the summaries and claims of some of the priority

[25] The '217 patent further acknowledges that there are "**various** implementations" of TCP (EX1001, 1:66-2:2), but only describes RFC 793 as the "TCP specification" (*id.*, 2:12-19). The TCP was defined by way of other RFC documents published after RFC 793 and before February 2010 (the earliest filing date to which the '217 patent purports to claim priority), such as RFC 1122 (EX1009) and RFC 3168 (EX1056). (*See also* EX1059 (excerpt of RFC 793 webpage on IETF website, noting RFC 793 was updated by at least RFCs 1122 and 3168 along with RFCs 6093 (EX1057) and 6528 (EX1058)).)

applications, starting from the ’802 application filed September 3, 2017. As I discussed below, the introduction of such terms this way, without more, does not convey to a person of ordinary skill in the art that the named inventor had possession of the non-TCP features encompassed by claim 1. My discussion below focuses on how the ’802 application does not convey to a person of ordinary skill in the art that the named inventor had possession of the non-TCP features, as it is the earliest application in the priority chain where non-TCP terms were introduced (the other later filed applications in the priority chain are defective for similar reasons).

184. Based on my review, the ’802 application introduced the non-TCP connection and non-TCP packet terms only (i) in its summary section of the specification and (ii) in the claims. (EX1044, 17-18 (¶¶[0012]-[0013]), 70-77; *see also id.*, 100 (substitute specification contains no new matter), 105-106 (¶¶[0012]-[0013]).) In fact, other than the changes to the summary of the ’802 application, the specifications of the ’802 application and the earliest priority application, i.e., the ’454 application, are substantially similar. (*See generally* EX1050 (computer-generated comparing the ’454 and ’802 applications)); *see also id.*, 2-5 (¶¶[0009]-[0017]).)

185. Notably, the introduction of the “non-TCP” terms in the summary section and claims of the as-filed ’802 application simply replaces the term “TCP” with “non-TCP” but otherwise generally discloses and claims systems and methods

similar to those disclosed in the earlier TCP-based applications. (*Compare, e.g.*, EX1044, 17-18 (¶¶[0012]-[0013]) (the new "non-TCP" embodiments in summary section) *with* EX1038, 122-123 (¶¶[0008]-[0009]) (the TCP embodiments disclosed in the pre-AIA '454 application); *compare* EX1044, 70-77 (the "non-TCP" claims) *with* EX1038, 100-106 (the TCP-based claims in the'454 application).) In other words, other than inserting the term "non-TCP" in the summary and claims of the '802 application, the '802 application includes **no explanation** regarding what is meant by "non-TCP" and how the disclosed techniques for sharing of information for detecting an idle TCP connection could be applied to non-TCP protocols for establishing non-TCP connections using non-TCP packets.

186. Thus, in my opinion, a person of ordinary skill in the art would not have understood based on the disclosure of the '802 application that the named inventor of the '217 patent had possession of the subject matter encompassed by claim 1 because the detailed description of the specification of the '802 application (which is substantially similar to the other priority applications as noted earlier) is exclusively focused on TCP-based connections and packets, as explained above in Section X.A.1.

187. Based on my review, the as-filed specification of the '802 application does not disclose any embodiment where time periods are detected during the time

a connection that is not TCP is being set up or after such a connection is set up (e.g., a non-TCP connection).[26]

188. For example, as I discussed above, the "first connection" in claim 1 encompasses a non-TCP connection. However, the detailed description of the '802 application discusses a method based on a **TCP connection** only—indeed, the term "TCP connection" appears 109 times in the detailed description. (*E.g.*, EX1044, [0057] ("Detailed information on the operation of TCP is included in RFC 793"), [0082] ("In an aspect, an ITP header may be included in a packet exchanged during setup of TCP connection. RFC 793 describes a "three-way handshake" for establishing a ***TCP connection***."), [0088] ("Other message exchanges may be used in setting up a ***TCP connection*** as those skilled in the art will understand. Such other exchanges are not currently supported by the TCP as described in RFC 793. … An equivalent or analog of an ITP header may be included in a footer of a protocol packet in an extension and/or variant of the current TCP.").)

[26] As noted above in Section X.A.1, whether the "non-TCP" terms encompass TCP extensions or variants is irrelevant because the '217 patent disclosure must convey to a person of ordinary skill in the art that the named inventor had possession of the full scope of these terms, which includes protocols that do not utilize TCP at all.

189. The only mention of the word "non-TCP" in the specification is in the summary section, which spans two paragraphs. (EX1044, 16-18 (¶¶[0011]-[0012]).) The summary section, however, explicitly states that any disclosure in this section is a "simplified summary of the disclosure in order to provide a basic understanding to the reader" and that "it **does not** identify key/critical elements of the invention or delineate the scope of the invention." (EX1044, 16 (¶[0009]).) It further states that "[i]ts **sole purpose** is to present some concepts disclosed herein in a simplified form as a prelude to the more detailed description that is presented later." (*Id.*) Thus, the mere fact that two paragraphs in the summary section of the specification for the '802 application refer to a "non-TCP connection" does not show that the named inventor had possession of the "non-TCP connection" elements of the claimed invention at the time of the '802 application.

190. In particular, as noted above, a person of ordinary skill in the art would have understood that a non-TCP protocol (like the UDP-based protocol) would operate in a different environment with its own set of rules regarding at least (i) the setup of a connection between two nodes in a network, (ii) how the devices would communicate with each other, (iii) the packet structure used for communication, and (iv) detection of timeouts that would cause the connection to close. (*See supra* Section X.A.1; EX1060, 1, EX1027, 1, EX1061, 1.) Thus, the generic description of "non-TCP" in the summary section, which the disclosure admits "**does not**

identify key/critical elements of the invention or delineate the scope of the invention" (EX1044, [0009]), without any further explanation of how the invention would operate even in a single exemplary non-TCP protocol, is insufficient to convey to a person of ordinary skill in the art that the named inventor had possession of such feature(s).

191. As I discussed above and further discussed below, far from describing the non-TCP features encompassed by the scope of claim 1 in any detail, none of the disclosures of the '217 patent family even include language mirroring the language of claim 20 which explicitly excludes TCP-based connections from the "first connection" (which, in any case, would be insufficient to convey to a person of ordinary skill in the art that the named inventor had possession of such features).

192. Additionally, a person of ordinary skill in the art would not have understood based on the disclosure of the '802 application that the setup and configuration of a TCP connection as recited in the detailed description was interchangeable with the setup and configuration of a **non**-TCP connection for the same reasons discussed above. (S*ee supra* Section X.A.1; *see also* EX1044, 17-18 (¶¶[0012]-[0013]), 70-77; *see also* Section X.A.1 (discussing similar limited disclosure in the '454 application regarding connection setup in TCP context).) In particular, as I discussed above in Section X.A.1, a person of ordinary skill in the art would have understood that a protocol that does not utilize TCP at all (like the UDP-

based protocol) would operate in a different environment with its own set of rules regarding at least (i) the setup of a connection between two nodes in a network, (ii) how the devices would communicate with each other, (iii) the packet structure used for communication, and (iv) detection of timeouts that would cause the connection to close. (EX1060, 1, EX1027, 1, EX1061, 1.) However, other than generally mentioning the term "non-TCP" in the summary section and the claims, the '802 application does not provide any explanation regarding how its disclosed techniques for sharing idle time information in a TCP options field of a TCP packet during setup of a TCP connection would have extended to an altogether different environment with its own rules regarding connection setup, connection operation, packet structure, and timeout detection. Hence, a person of ordinary skill in the art would not have understood that the disclosure in the detailed section of the '802 application convey to a person of ordinary skill in the art that the named inventor had possession of establishing a non-TCP connection.

193. The originally-filed claims of the '802 application reciting "non-TCP" features do not convey to a person of ordinary skill in the art that the named inventor had possession of the "first connection" term of claim 1 of the '217 patent, which encompasses non-TCP connections, for similar reasons. In particular, they recite "non-TCP" features in the context of nodes connected via a "non-TCP" connection and exchanging information in "non-TCP" packets. (EX1044, 70-77.) However,

the claims fail to provide any detail regarding any connections that do not use TCP. And as I discussed above, other than generally mentioning the term "non-TCP" in the summary section, the '802 application does not provide any explanation regarding how its disclosed techniques for sharing idle time information in a TCP options field of a TCP packet during setup of a TCP connection would have extended to an altogether different environment with its own rules regarding connection setup, connection operation, packet structure, and timeout detection.

194. Accordingly, nothing in the four corners of the specifications of the '802 application and the later filed applications in the priority chain (including the '824 application), which all share substantially similar specifications, demonstrates that the named inventor was in possession of the non-TCP-based features encompassed by claim 1. (EX1045, 84-86 (¶¶[0012]-[0013]), 137-144 (original-filed claims); EX1046, 168-173 (¶¶[0010]-[0013]), 221-226 (original-filed claims); EX1047, 229-234 (¶¶[0010]-[0013]), 220-225 (original-filed claims); EX1048, 30-34 (¶¶[0010]-[0013]), 83-88 (original-filed claims); EX1004, 208 (¶[0010]), 251-258 (original-filed claims).)

195. Thus, the originally-filed disclosures of the '217 patent family, including the '217 patent itself, do not disclose the non-TCP features encompassed in the scope of claim 1 (let alone the explicit features set forth in claim 20).

4. Dependent Claims 6, 12, 14, 17, and 20 Are Not Supported

196. Claims 6, 12, 14, 17, and 20 depend from claim 1. Thus, these claims incorporate the features of claim 1 and are not supported by the as-filed disclosure of the ’217 patent and its priority applications.

B. Claim 1: No Support for Detecting a Time Period During At Least A Portion of the First Connection Setup

197. Claim 1 recites:

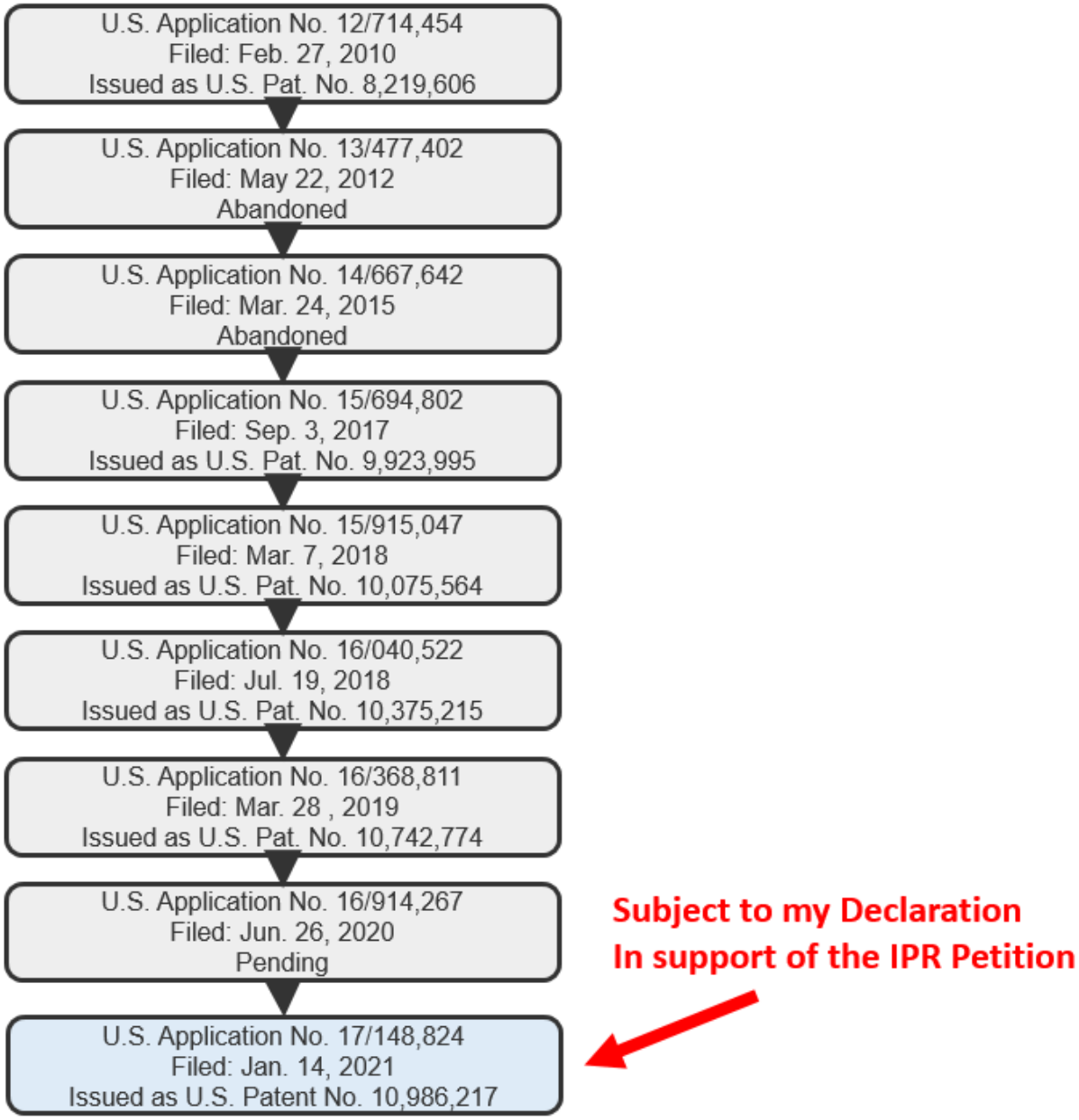
> in response to detecting, based on the first duration and by the first node during at least a portion of the first **connection including at least a portion of the first connection set up**, a first time period of the first type of time period, at least partially close the first connection.

(EX1001, 21:45-50.)

198. Claim 1 recites features relating to (i) detecting a time period during a portion of the connection setup and (ii) at least partially closing the connection based on the detection (generally referred to herein as the “*time period detection during connection setup*” features). The *time period detection during connection setup* feature encompasses scenarios where the time period (e.g., a timeout) is detected and the connection is partially closed during the connection setup time.

199. In my opinion, a person of ordinary skill in the art would not have understood that the named inventor of the ’217 patent had possession of the claimed invention, including the *time period detection during connection setup* features set

forth in claim 1 at the time the ’824 application was filed because the as-filed specification of the ’824 application and the specifications of the ’217 patent’s priority applications (shown below) do not support these features.



1. The ’824 Application Does Not Support the Time Period Detection During Connection Setup Features of Claim 1

200. A variation of the *time period detection during connection setup* features set forth in the challenged claims were introduced in the priority chain of the ’824 application by way of the original claims presented in the ’824 application filed January 14, 2021. (*See, e.g.*, EX1004, 251-252.) The original claims in the ’824 application included a variation of the *time period detection during connection setup* feature—they differed in that they did not include the condition of “an absence of one or more packets” as part of the basis for detecting the first time period and “releasing…the resource” as part of partially closing the connection. (*See, e.g.*, *id.*, 254-255 (claim 1 reciting “in response to detecting, based on the first duration and by the first node during at least a portion of the first connection including at least a portion of the first connection setup, a first time period of the first type of time period, at least partially close the first connection”), 207-208 (similar language in summary section of specification).) However, the ’824 application does not provide support for the *time period detection during connection setup* features set forth in the challenged claims.[27]

[27] The ’824 application claims priority to the ’267 application. (*See supra* Section VI.) U.S. patent application no. 17/079,397 (“the ’397 application”) filed October

201. The ’824 application discloses detection of time periods **after** a TCP connection is set up (established) for purposes of detecting idle time periods and closing such idle TCP connections.[28] For example, like the priority applications, the ’824 application discloses by way of its background that “there exists a need for methods, systems, and computer program products for sharing information for detecting **an idle TCP connection**.” (EX1004 (’824 application), 207 (¶[0008]);

23, 2020 also claims priority to the ’267 application. (*See* EX1076, 471-472.) The ’397 application issued as U.S. Patent No. 10,751,742 (“the ’742 patent”). (*See* EX1076, 1.) I have been informed that the ’397 application is not part of the priority chain of the ’824 application and thus not directly relevant to my analysis here. Nonetheless, I note that features similar to the *time period detection during connection setup* features were introduced in the ’397 application via a preliminary amendment filed December 2, 2020 (i.e., after the application was filed). (EX1076, 427-444). The ’397 application does not provide written description support for the *time period detection during connection setup* features for all the reasons I explain herein. (*See*, *e.g.*, EX1055.)

[28] As discussed above in Section X.A, the ’217 patent does not disclose any embodiments related to a protocol that is not TCP.

EX1048 ('267 application), 27 (¶[0008]); EX1047 ('811 application), 229 (¶[0008]); EX1046 ('522 application), 168 (¶[0008]); EX1045 ('047 application), 83 (¶[0008]); EX1044 ('802 application), 154 (¶[0008]); EX1039 ('402 application), EX1038 ('454 application), 122 (¶[0008]).)[29] A POSITA would have understood that a "TCP connection" is only established **after two nodes exchange initial messages to set up the TCP connection**. (EX1004, 229 (¶¶[0075]-[0081] (describing that the idle time period (ITP) header is "exchanged **during setup of [a] TCP connection**" and "RFC 793 describes a 'three-way handshake' **for establishing a TCP connection**")), 221-222 ((¶[0051] (describing that the ITP header includes idle information specifying a value used to detect an idle TCP connection)).)

202. The '824 application discloses examples of timeouts in a TCP connection that may be modified based on the idle information shared during the connection setup. (EX1004, 226 (¶[0068]) ("In an aspect, a TCP keep-alive option, a TCP user timeout, a retransmission timeout, an acknowledgment timeout, and/or another timeout associated with a TCP connection may be modified based on the first idle information."), 246 (¶[0119]).) And while the '824 application discloses that "[t]he specified 'three-way handshake' and other patterns of message exchange

[29] As noted earlier, the priority '642 application is directed to different subject matter. (*See supra* Section X.A.1.)

for setting up a TCP connection include packets that are considered to be in the TCP connection for purposes of this disclosure" (EX1004, 229 (¶[0081])), the '824 application does not describe when its disclosed timeouts are detected and does not provide written description support for any detection of such timeouts and partial deactivation of the connection during the connection setup, as claimed.

203. As discussed below, a POSITA would have understood that these timeouts described by the '824 application disclosure, such as the TCP keep-alive option, TCP user timeout, and retransmission timeout, are initiated based on data being sent or received, and RFC 793 makes clear that data cannot be exchanged between nodes until after connection establishment (i.e., **after** the connection is set up). Thus, detection of any of these timeouts would occur only after the connection is established since data exchange would occur only after the connection is established. A POSITA, therefore, would not have understood that the '824 application discloses the claimed invention, including the *time period detection during connection setup* features set forth in claim 1 at the time the '824 application was filed.

204. In particular, the '217 patent family, including the '824 application, relies on RFC 793 for TCP implementation details. (*See, e.g.*, EX1004, 206 (¶[0004]), 221 (¶[0050] ("Detailed information on the operation of TCP is included in RFC 793")), 229 (¶[0075]), 230 (¶[0083]).) RFC 793 discloses that "[w]hen two

processes wish to communicate, their TCP's **must first establish a connection**." (EX1008, 9; *see also id.*, 31-32 (describing that "[f]or a connection to be established or initialized, the two TCPs must first synchronize on each other's initial sequence numbers" and the "three-way handshake" which is also described in the '824 application); EX1004, 229 (¶¶[0075]-[0080]).) RFC 793 discloses that data is sent or received between nodes after the TCP connection is established. (EX1008, 44 ("Once the connection is established data is communicated by the exchange of segments").) Moreover, RFC 793 discloses that a retransmission timeout is initiated based on data being sent or received. (*Id.*, 14 ("**When the TCP transmits a segment containing data, it puts a copy on a retransmission queue and starts a timer**; when the acknowledgment for that data is received, the segment is deleted from the queue. If the acknowledgment is not received before the timer runs out, the segment is retransmitted."), 45 (describing "An Example Retransmission Timeout Procedure" involving "[m]easur[ing] the elapsed time between **sending a data octet with a particular sequence number and receiving an acknowledgment that covers that sequence number**").)

205. In other words, RFC 793, upon which the '824 application relies for TCP implementation details, does not disclose detecting any time period (let alone a time period associated with closing the TCP connection) during the time the TCP connection is being set up (e.g., during the time the messages in the three-way

handshake are being exchanged). As such, a POSITA would have understood that when the ’824 application discloses modifying timeout attributes of a TCP connection, such as the TCP keep-alive option, user timeout, or retransmission timeouts, any detection of time periods associated with such timeout attributes would have been initiated **after** establishment of the connection and not during the connection setup, as claimed.

206. Additionally, independent claim 1 recites the *time period detection during connection setup* features in the context of a connection that may be a non-TCP connection, as discussed above. Thus, the ’824 application needs to disclose the *time period detection during connection setup* features in the context of a non-TCP protocol/connection. But as noted above, the ’824 application is limited to TCP-based embodiments and does not disclose any non-TCP embodiments which include the *time period detection during connection setup* features. Indeed, even if the priority applications supported the *time period detection during connection setup* features during the three-way handshake procedure for setting up a TCP connection (they do not), that still would be insufficient disclosure since all non-TCP protocols do not necessarily use such a handshake procedure to establish a connection.

2. The Priority Applications Do Not Support the Time Period Detection During Connection Setup Features of Claim 1

207. As I discussed above, the disclosures of the priority applications (other than the ’642 application) are substantially similar to each other. (*See supra* Sections X.A.1, X.A.3.) Thus, these priority applications, namely the ’454, ’402, ’802, ’047, ’522, ’811, and ’267 applications, do not provide any additional explanation regarding the *time period detection during connection setup* features set forth in the challenged claims. And as discussed above, the ’642 application was significantly amended such that it is directed to different subject matter—the ’642 application also does not provide any additional explanation regarding the *time period detection during connection setup* features set forth in the challenged claims.[30] (*See supra* Section X.A.1.)

208. Accordingly, a POSITA would not have understood that the named inventor had possession of the claimed invention, including the *time period detection during connection setup* features at the time the ’824 application was filed because their as-filed specifications and the specifications of the ’217 patent’s remaining priority applications do not support these features as set forth in claim 1.

[30] The ’063 application also does not disclose these features. (*See generally* EXS1041-1042.)

3. Dependent Claims 6, 12, 14, 17, and 20 Are Not Supported

209. Claims 6, 12, 14, 17, and 20 depend from claim 1. Thus, these claims incorporate the features of claim 1 and are not supported by the as-filed disclosure of the ’217 patent and its priority applications.

XI. CONCLUSION

210. I declare that all statements made herein of my knowledge are true, and that all statements made on information and belief are believed to be true, and that these statements were made with the 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: May 18, 2023 By: _____
Bill Lin, Ph.D.