UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

GOOGLE LLC
Petitioner

v.

JENAM TECH, LLC
Patent Owner

Patent No. 10,742,774

DECLARATION OF DR. BILL LIN

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. 10,742,774 ("the '774 patent") (Ex. 1001). I have been asked to consider whether certain references disclose or suggest the claim elements recited in claim 1 ("the challenged claim") of the '774 patent. I have also been asked to consider whether a person of ordinary skill in the art would have understood the scope of claim 1 with reasonable certainty when this claim is read in light of the specification, drawings, and prosecution history of the '774 patent. I have also been asked to consider whether a person of ordinary skill in the art would have understood that the named inventor of the '774 patent was in possession of the subject matter recited claim 42 of the '774 patent based on the disclosures provided by the U.S. Patent Application No. 12/714,454 ("the '454 application") or the U.S. Patent Application No. 13/477,402 ("the '402 application"). My opinions are set forth below.

2. I am being compensated at my rate of $600 per hour 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 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):

Ex. 1001	U.S. Patent No. 10,742,774
Ex. 1004	File History of U.S. Patent Application No. 16/368,811 (U.S. Patent No. 10,742,774)
Ex. 1005	U.S. Pre-Grant Publication No. 2007/0171921 to Wookey *et al*.
Ex. 1006	L. Eggert, TCP Abort Timeout Option, draft-eggert-tcm-tcp-abort-timeout-option-00, Network Working Group, Internet-Draft (April 14, 2004) ("*Eggert*")
Ex. 1007	U.S. Patent No. 6,981,048 to Abdolbaghian ("*Abdolbaghian*")
Ex. 1008	DARPA RFC 793 TRANSMISSION CONTROL PROTOCOL
Ex. 1009	RFC 1122 "Requirements for Internet Hosts -- Communication Layers"
Ex. 1010	U.S. Patent No. 9,923,995
Ex. 1011	RFC 1001
Ex. 1012	U.S. Patent No. 6,212,175 to Harsch
Ex. 1013	U.S. Patent No. 6,665,727 to Hayden
Ex. 1014	U.S. Patent No. 7,636,805 to Rosenberg

Ex. 1015	RFC 5440
Ex. 1020	U.S. Pre-Grant Publication No. 2004/0093376 to De Boor *et al.*
Ex. 1021	U.S. Patent 7,535,913 to Minami *et al.*
Ex. 1022	U.S. Pre-Grant Publication No. 2005/0204013 to Raghunath *et al.*
Ex. 1023	U.S. Pre-Grant Publication No. 2007/0005804 to Rideout
Ex. 1024	U.S. Pre-Grant Publication No. 2004/0098748 to Bo *et al.*
Ex. 1025	IETF RFC 2616 Hypertext Transfer Protocol -- HTTP/1.1
Ex. 1026	U.S. Patent No. 8,259,716 to Diab
Ex. 1027	Bova et al., RELIABLE UDP PROTOCOL <draft-ietf-sigtran-reliable-udp-00.txt> 25 February 1999
Ex. 1028	U.S. Patent 6,674,713 to Berg et al.
Ex. 1036	U.S. Patent No. 10,075,564
Ex. 1038	File History of U.S. Patent Application No. 12/714,454 (U.S. Patent No. 8,219,606)
Ex. 1039	File History of U.S. Patent Application No. 13/477,402
Ex. 1040	Computer-generated Redline Between U.S. Patent Application No. 12/714,454 and U.S. Patent Application No. 13/477,402
Ex. 1041	File History of U.S. Patent Application No. 12/714,063
Ex. 1042	Computer-generated Redline Between U.S. Patent Application No. 12/714,454 and U.S. Patent Application No. 12/714,063
Ex. 1043	File History of U.S. Patent Application No. 15/694,802 (U.S. Patent No. 9,923,995)
Ex. 1044	File History of U.S. Patent Application No. 14/667,642
Ex. 1045	File History of U.S. Patent Application No. 16/040,522 (U.S. Patent No. 10,375,215)
Ex. 1046	RFC 3168
Ex. 1047	RFC 6093

Ex. 1048	RFC 6528
Ex. 1049	Excerpt of IETF webpage hosting RFC 793, retrieved May 2, 2021
Ex. 1050	U.S. Patent No. 6,584,546 to Kavipurapu

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 ’774 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 ’774 patent, the prior art to the patent, and the state of the art at the time of the alleged invention.

15. As I discuss in detail below, it is my opinion that certain references disclose or suggest all the claim elements recited in claim 1 of the ’774 patent.

IV. PERSON OF ORDINARY SKILL IN THE ART

16. I have been asked to consider the time of the alleged invention for the ’774 patent to be the early 2010 time frame, including and up to February 27, 2010, which is the earliest filing date of an application that is associated with the ’774 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 ’774 patent.

17. Based on my knowledge and experience, I understand what a person of ordinary skill in the art would have known at the time of the alleged invention of the ’774 patent. In my opinion, based on my review of the ’774 patent, such a person 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, 1:55–2:39 (discussing the background of the ’774 patent), 2:43–6:6 (discussing a “summary of the disclosure” (*id.*, 2:45–50)).)

18. My analysis of the ’774 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.

V. TECHNICAL BACKGROUND

19. In this section, I discuss the state of the art relevant to the subject matter of the ’774 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 ’774 patent (e.g., early 2010 time frame, including and up to February 27, 2010, as I mentioned above in Section IV). I rely on and incorporate the understandings below regarding such known features and technologies to support my opinions regarding the knowledge of a person of ordinary skill in the art at the relevant timeframe and the motivations relating to how the prior art discloses and/or suggests the limitations of claim 1 of the ’774 patent. In my opinion, the understandings regarding the technologies, processes, and features as I discussed in this section (Section V) would have been in the mind of a person of ordinary skill in the art when contemplating the disclosures and/or suggestions in the prior art and applications related to the ’774 patent that I address below in Sections IX, X, and XI and when considering combining features from that prior art in the manner that I explain below in Section XI.

A. Computer Networks and Communication Protocols

20. 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.

21. 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

22. 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.

23. 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 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	

24. 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.

25. 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).

26. In the TCP/IP model, the "Internet" layer was also known to 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 includes guaranteed delivery and congestion control. UDP was known to provide a simpler connectionless service that does not provide for reliability or flow control.

27. 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.

28. 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;

29. It was known prior to and at the time of the alleged invention for the '774 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.").)

30. 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

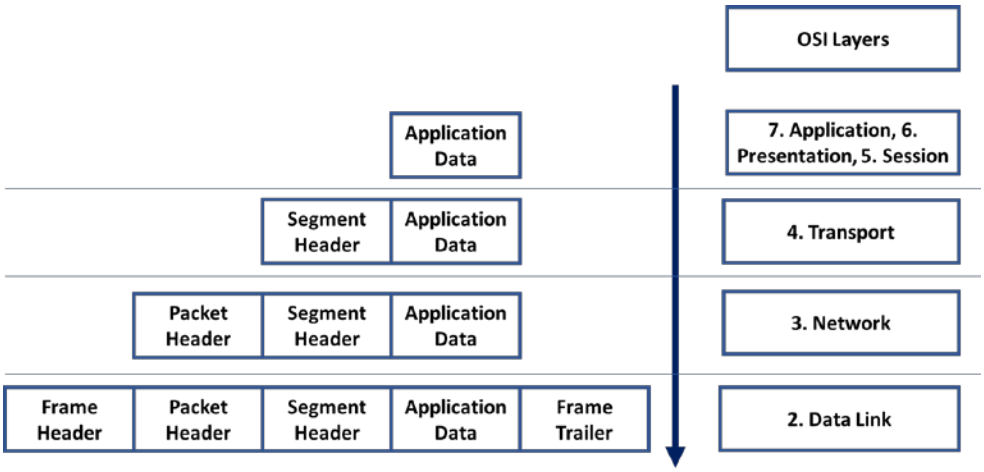
masking any differences in data formats that may occur between dissimilar or disparate systems. (*Id*., 2:21–26.) The presentation layer specifies architecture-independent data transfer formats. (*Id*.) The OSI model's Layer 5, the session layer, was known to be typically responsible for managing user 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*.)

31. 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*.)

32. 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.)

33. 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:



34. 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.

35. 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

36. 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].)

37. 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. 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).)

38. 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.)

39. 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 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, 2:23–3.) 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

40. 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).)

41. 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).)

42. 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

43. 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).)

44. 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.)

a) **TCP/IP**

45. 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

46. 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 it's 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).)

47. 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

48. 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

49. 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.

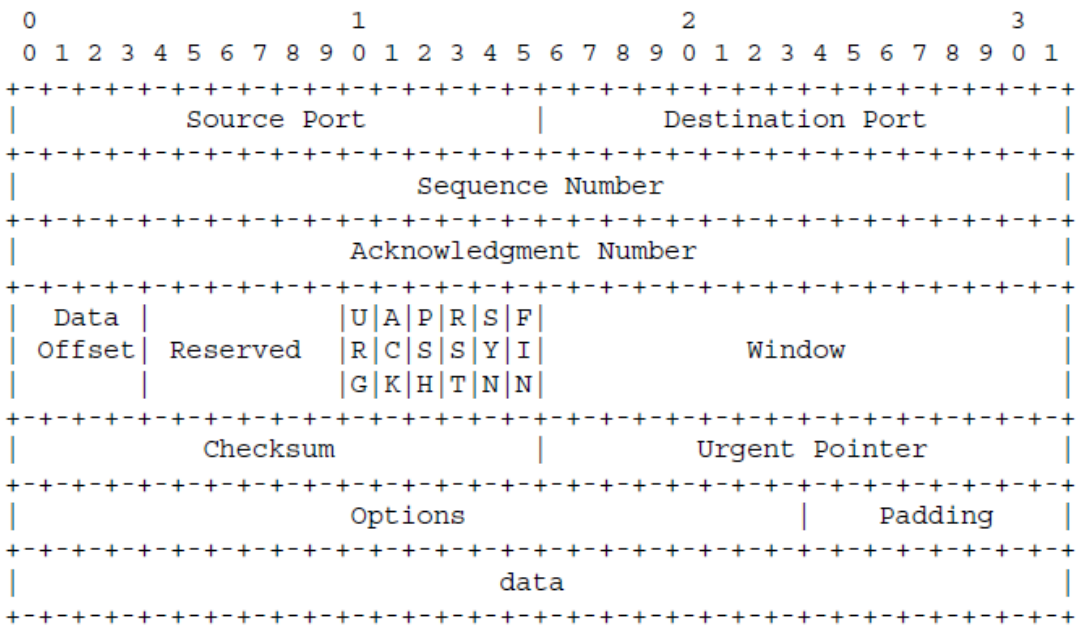
(*Id.*, 77 (excerpted with emphasis added).)

50. 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

51. 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



TCP Header Format

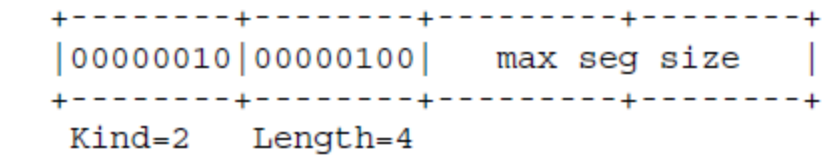
Note that one tick mark represents one bit position.

Figure 3.

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

52. 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



(*Id.*, 18.)

53. 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**

54. 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**

55. 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**

56. 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.))

[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.)

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

58. 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:

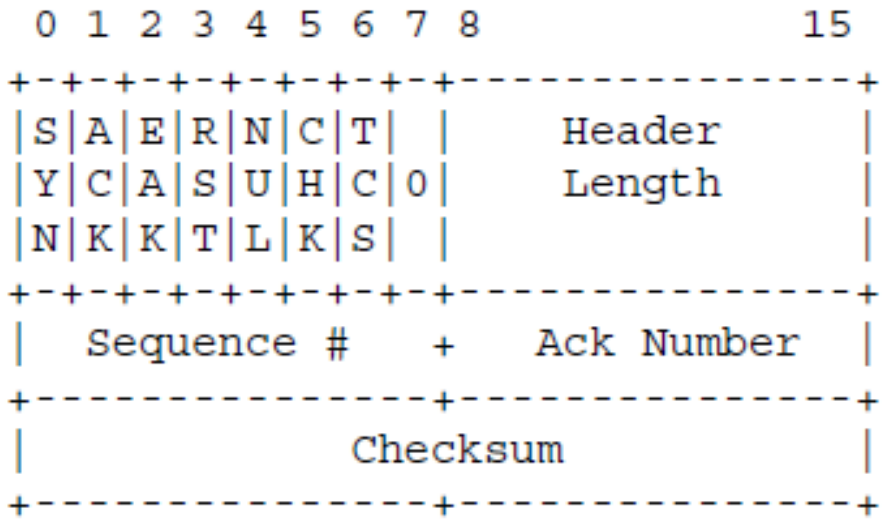


Figure 1, RUDP Header

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

59. 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).)

60. 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)

61. 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).)

62. 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)

63. 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 ’774 PATENT

64. The ’774 patent is directed to networking, and in particular, to the sharing of information for detecting an idle TCP connection. (*See*, *e.g.*, Ex. 1001, Title, 2:27–29, 8:6–27.) Figure 7 below depicts a method for sharing this information.

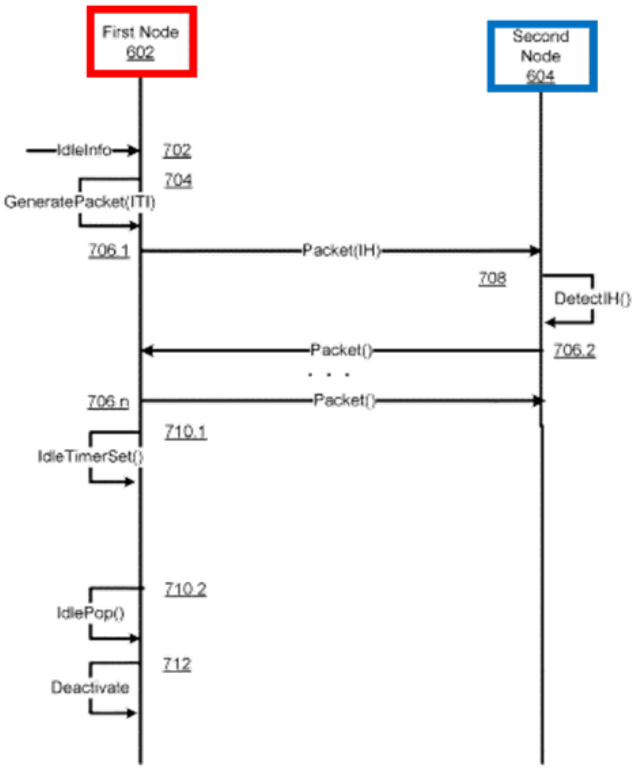


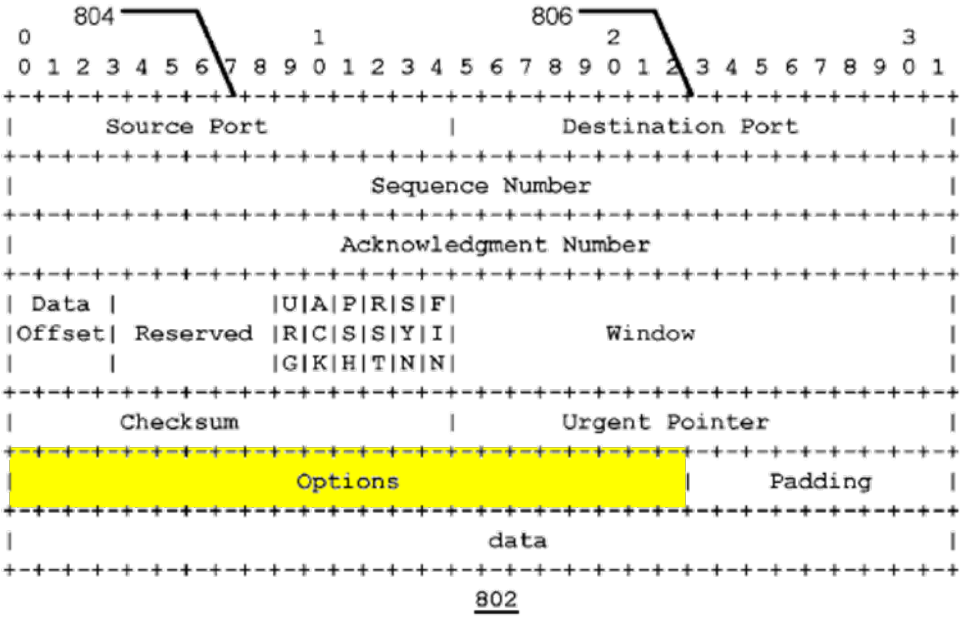
Fig. 7

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

65. 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.*, 11:30–39.) Message 702 may take various forms, and thus can be "a message received via a network." (*Id.*, 11:34–39.) The idle information "may include and/or identify a duration of time for detecting an idle time period." (*Id.*, 12:12–18.) The duration "may be specified according to various measures of time[,] including seconds, minutes, hours, and/or days." (*Id.*)

66. 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.*, 15:45–48, 16:3–5.) The ITP header can be in the TCP options field of a TCP packet. (*Id.*, 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. (*Id.*, 6:5–8, 15:20–28.)



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

67. 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.*, 15:22–28.)

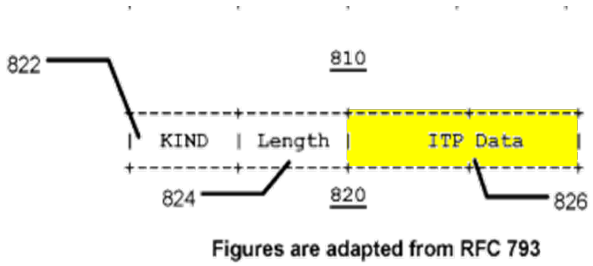


Fig. 8

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

68. The value representing the ITP can be part of the ITP header exchanged during the three-way handshake (*Id.*, 14:25–39) or "in a TCP packet in structures and locations other than those specified for TCP options in RFC 793" (*id.*, 15:29–35). For example, as shown in Figure 7 above, the **first node 602** transmits a message 706.1, which is a TCP packet including an ITP header (IH) that contains ITP information, to the **second node 604**. (*Id.*, 16:3–5.) Message 708 exemplifies the detection of the ITP header in the TCP packet received from the **first node 602** by the **second node 604**. (*Id.*, 21:25–29.)

69. The ’774 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.” (EX1001, 13:62–65.)

70. Claim 1 recites limitations relating to the above features but in context of “TCP-variant” packet and connection. (EX1001, claim 1.) However, the ’774 patent describes its features in terms of a TCP connection and packet and never uses the term “TCP-variant” or explains the use of a “TCP-variant” packet or connection in the way recited in claim 1. (*See my discussions and explanations below* in Section X; *see also generally* EX1001, 2:43–6:6 (Summary discussing aspects relating to a “TCP” packet and “TCP” connection)[2], 6:48–24:20 (Detailed Description section discussing aspects in terms of a “TCP” packet and “TCP” connection)[3].)

[2] While the Summary section also refers to a second protocol that is “separate” or “different” from “the TCP” (EX1001, Abstract, 3:6–7, 3:39–40, 4:9–10, 4:50–51), the detailed description provides no disclosures of these features at all. (*Id.*, 6:48–24:20).

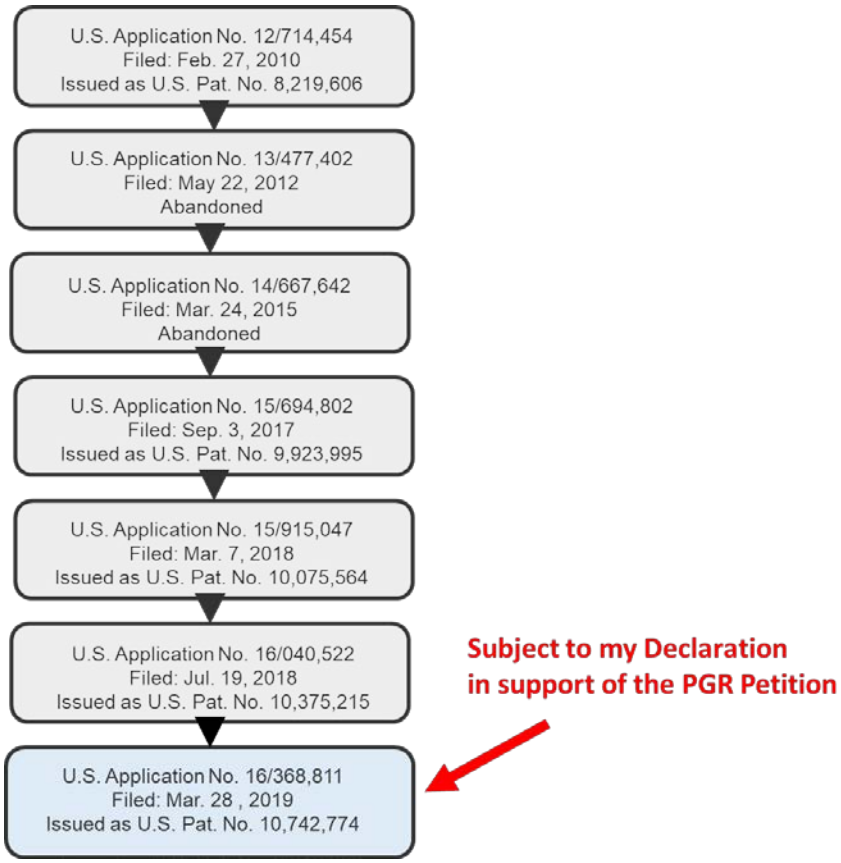
[3] As I discuss below, the specification does mention a “variant of the current TCP,” but does so in terms of the location and structure of an ITP header that “may be

71. As I also discuss below in Sections VIII and XI, it is my opinion that all of these features were known in the art.

72. I understand that '774 patent issued on August 11, 2020 from U.S. Patent Application No. 16/368,811 ("the '811 application") (Ex. 1004), which was filed on March 28, 2019. (Ex. 1001 at Cover.) I also understand that the '811 application is a continuation application that claims priority to U.S. Patent Application No. 16/040,522 ("the '522 application"), filed on July 19, 2018. (*Id.*) I understand that the '522 application is a continuation application that claims priority to U.S. Patent Application No. 15/915,047 ("the '047 application"), filed on March 7, 2018. (*Id.*) I understand that the '047 application is a continuation application that claims priority to U.S. Patent Application No. 15/694,802 ("the '802 application"), which issued as U.S. Pat. No. 9,923,995 ("the '995 patent"), filed on September 3, 2017. (*Id.*) I understand that the '802 application is a continuation-in-part application that claims priority to U.S. Patent Application No. 15/694,642 ("the '642 application"), filed on March 24, 2015. (*Id.*) I understand that the '642 application is a continuation-in-part application that claims priority to U.S. Patent Application No. 13/477,402 ("the '402 application"), filed May 22,

included in a TCP packet." (EX1001, 16:1–7; *see my discussions below* in Section X.) There is no discussion of a "TCP-variant" connection as recited in claim 1.

2012. (*Id.*) I understand that the '642 application is a continuation application that claims priority to U.S. Patent Application No. 12/714,454 ("the '454 application"), filed Feb. 27, 2010. (*Id.*) I have created the demonstrative below to help visualize the purported priority chain for the '774 patent.



VII. CLAIM CONSTRUCTION

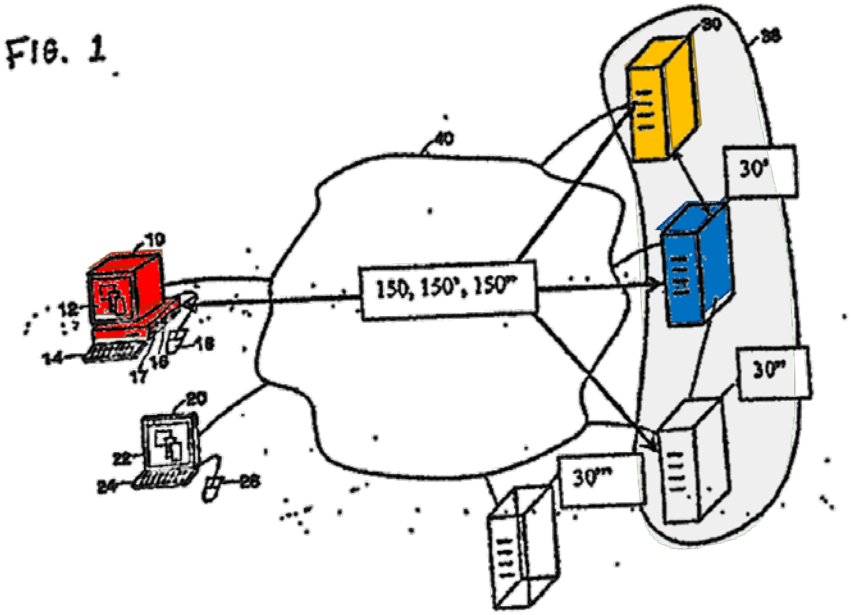
73. I have been asked to give the terms in the challenged claim their plain and ordinary meanings, as would have been understood by a person of ordinary skill in the art at the time of the alleged invention, taking into consideration the language of the claims, the specification, and the prosecution history of the ’774 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[4]

A. Overview of *Wookey* (Ex. 1005)

74. *Wookey* discloses "a method for making a hypermedium page interactive." (Ex. 1005, Abstract; *see also id.*, ¶[0002] ("methods and systems for making a hypermedium page interactive").) For example, *Wookey* explains that a "client machine executes a browser application" (*id.*, ¶[0016]) for "displaying a hypermedium page including a hyperlink identifying a desired computing resource" (*id.*). (*See also id.*, Abstract, ¶¶[0013], [0196] (providing that a remote machine, such as a web server, sends the hypermedium page (i.e., Webpage) to the client machine).) Figure 1, below, shows an exemplary environment in which a **client machine 10** accesses a computer resource provided by a network server (e.g., **remote machine 30**. (*Id.*, ¶[0020], FIG. 1.)

[4] These summaries of the prior art references in Section VIII are applicable to my analysis below in Section XI. 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 XI below).



(*Id.*, FIG. 1 (emphasis added).)

75. For instance, Figure 2C below shows an embodiment where **remote machine 30** is a web server that hosts an HTML page. (*Id.*, ¶¶[0192]–[0193], FIG. 2C; *see also id.*, ¶[0024].) The **client machine 10** executes a web browser application 280 (*id.*, ¶[0192]) that "transmits a request 282 to access a Uniform Resource Locator (URL) address corresponding to an HTML page residing on remote machine [30]" (*id.*, ¶[0193].) The request may be for "an enumeration of available computing environment[s]." (*Id.*, ¶[0213].)

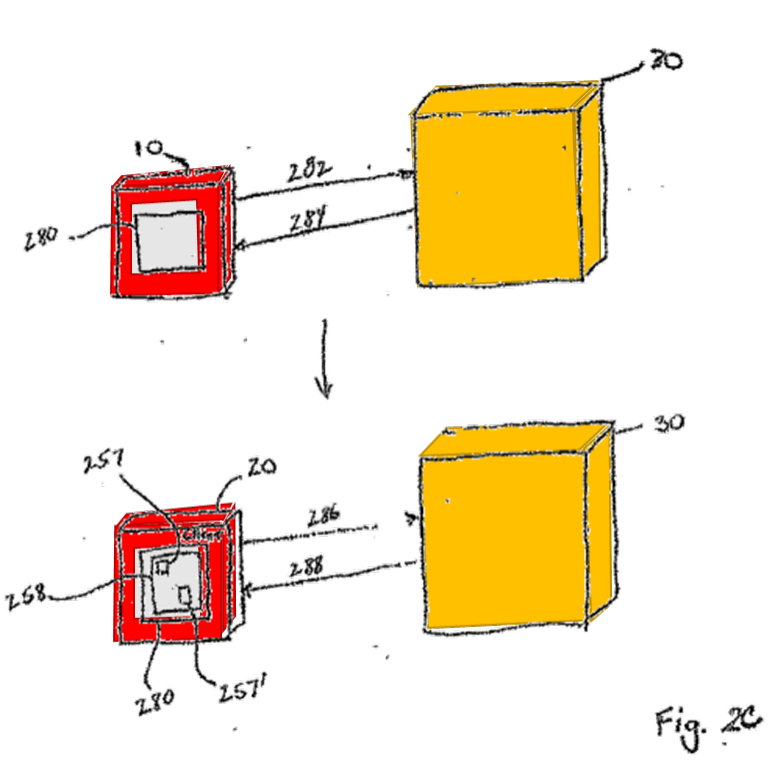


Fig. 2C

(*Id.*, FIG. 2C (annotated); *see also id.*, ¶[0021] (depicting a block diagram of an embodiment in which a client machine uses a web browser application to determine its resource neighborhood).)

76. Following this request from the **client machine 10**, the **remote machine 30** returns an HTML page to the **client machine 10**. (*Id.*, ¶[0193].) For example, the **remote machine 30** may return "an authentication page that seeks to identify the client machine 10 or the user of the client machine 10." (*Id.*) Or the **remote machine 30** may return "an HTML page 288 that includes a Resource

Neighborhood window 258 in which appears graphical icons 257, 257' representing resources to which the client machine 10 has access." (*Id*., ¶[0196].)

77. Then the **remote machine 30** *transmits* this data to the **client machine 10**, as shown in Figure 2C. (*Id*.) The **client machine 10** displays the received "HTML page 288 that includes a Resource Neighborhood window 258 in which appears graphical icons 257, 257' representing resources to which the client machine 10 has access." (*Id*.; *see also id*., ¶[0215] ("Once the output display is generated, it is transmitted to client machine 10 for display by the browser program 80.").)

78. The "HTML page 288 that includes a Resource Neighborhood window 258 in which appears graphical icons 257, 257' representing resources to which the client machine 10 has access." (*Id*., ¶[0196].) And "[a] user of the client machine 10 can access a resource by clicking an icon 257, 257' displayed in the Resource Neighborhood web page." (*Id*.; *see also id*., ¶¶[0196] ("A user of client machine 10 requests access to a resource represented by icon 257 by clicking that icon 257."), [0216].)

79. "[E]ach icon 257, 257' is associated with an encoded URL that specifies: the location of the resource (i.e., on which remote machines it is hosted or, alternatively, the address of a master remote machine, a gateway, or other remote machine 30); a launch command associated with the resource." (*Id*.,

¶[0216].) And "the URL includes a file, or a reference to a file, that contains the information necessary for the client to create a connection to the remote machine hosting the resource." (*Id*.)

80. If an icon is selected, "[t]he client machine 10 establishes a connection (arrow 394) with the remote machine 30' identified as hosting the requested resource and exchanges information regarding access to the desired resource." (*Id*.) One example of this result of establishing a second connection 394 is shown below with respect to Figure 3D. (*Id.*, FIG. 3D.)

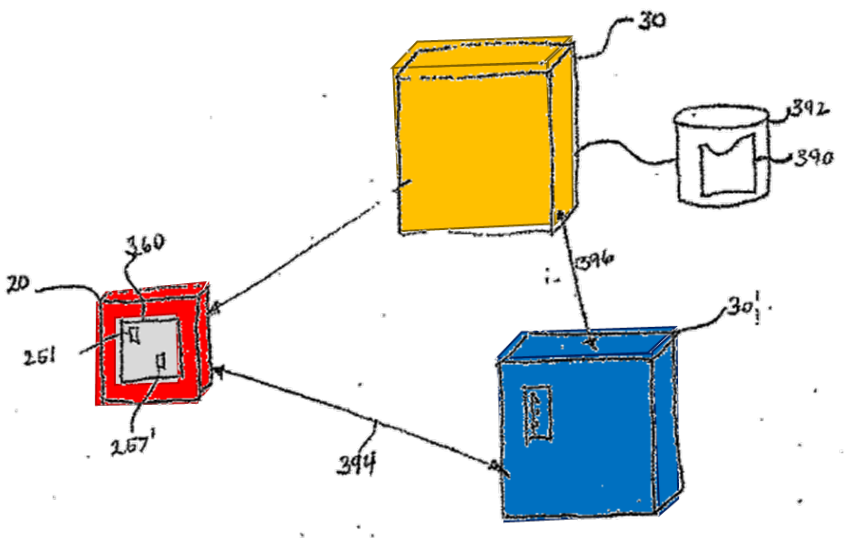


Fig. 3D

(*Id.*, FIG. 3D, ¶[0026] (depicting a block diagram of an embodiment in which a client machine can access a resource from a resource neighborhood web page displayed at that client machine).)

81. *Wookey* explains that "[c]onnections can be established using a variety of communication protocols," such as a TCP/IP connection. (*Id.*, ¶[0156]; *see also id*. ¶[0155] ("[t]he communications link 150 may use a transport layer protocol such as TCP/IP or any application layer protocol, such as the Hypertext Transfer Protocol (HTTP).").) And the connection 394 made be established using other protocols including "the Independent Computing Architecture (ICA) protocol" (*id*., ¶[0216]), "the RDP protocol" (*id*.), "X11 protocol" (*id*.), "the Virtual Network Computing (VNC) protocol" (*id*.).

82. *Wookey* also teaches that "the client machine 10 *periodically* transmits a signal to the remote machine 30 to confirm that a connection is still intact." (*Id.*, ¶[1136].) And "if the server process 7922 detects that a predetermined number of expected confirmation signals from a client machine 10 have not arrived, the server process 7922 determines that the client machine 10 has disconnected." (*Id*.) And upon detecting this disconnection, "if a user fails to connect within that time period, the server process 7922 stalls the application 7916" (*id.*, ¶[1135]) or the server may "disconnect an application session 7918 from the client machine 10 that the user is communicating from" (*id*., ¶[1136]).

83. Additionally, *Wookey* teaches that "[t]he client machine 10 transmits at least one heartbeat message to a remote machine (step 1816)." (*Id.*, ¶[0656].) *Wookey* explains that this "heartbeat message" is transmitted to "retain authorization to execute the plurality of resource files comprising the enumerated resource." (*Id.*, ¶[0656].) And "if a remote machine 30 disconnects from a network on which it replies, the client machine 10 may not receive a reply to a heartbeat message transmitted to the remote machine 30." (*Id.*, ¶[0698].)

B. Overview of *Eggert* (Ex. 1006)

84. Like *Wookey*, *Eggert* similarly relates to establishing a reliable connection between two nodes, where the host may be mobile (e.g., the host experiences "changes [in] network attachment points based on [its] current location"):

> Some hosts are only intermittently connected to the Internet. One example is ***mobile hosts that change network attachment points based on current location***, for example using MobileIP [5] or HIP [6].

(Ex. 1006, 2–3 (excerpted with emphasis added).)

85. *Eggert* describes a modification to the TCP protocol to "allow established TCP connections to survive periods of disconnection." (Ex. 1006, Abstract.) In particular, *Eggert* introduces a "TCP Abort Timeout Option" (ATO) that "allows conforming TCP implementations to negotiate individual, per-connection abort timeouts."

> The TCP Abort Timeout Option allows conforming TCP implementations to negotiate individual, per-connection abort timeouts. Lengthening abort timeouts allows established TCP connections to survive periods of disconnection.

(*Id.*) Thus, *Eggert* discloses a TCP variant protocol that allows established TCP connections to survive periods of disconnection by negotiating abort timeouts. (*Id.*)

86. *Eggert* explains that its TCP-variant protocol solves a problem in the art "where hosts are only intermittently connected to the Internet." (*Id.*, 1–3.) For example, *Eggert* notes that at the time, a mobile host may "experience disconnected periods during which no network service is available" when the "mobile hosts … change network attachment points." (*Id.*, 1–3.) These disconnected periods may lead to the "established TCP connections" being "abort[ed] during periods of disconnection" on the mobile host. (*Id.*)

> In between connected periods, ***mobile hosts may experience disconnected periods during which no network service is available***. When such hosts use the Transmission Control Protocol (TCP) [1], ***their established TCP connections can abort during periods of disconnection***.

(*Id.*, 2–3 (excerpted with emphasis added).)

87. *Eggert* overcomes this problem in the art by "specif[ying] a new TCP option - the Abort Timeout Option [ATO] - that allows conforming hosts to negotiate per-connection abort timeouts." (*Id.*, 3) *Eggert*'s TCP-variant protocol "allow[s] mobile hosts to maintain TCP connections across disconnected periods that are longer than their system's default abort timeout":

> This document specifies a new TCP option - ***the Abort Timeout Option - that allows conforming hosts to negotiate per-connection abort timeouts***. This allows mobile hosts to ***maintain TCP connections across disconnected periods that are longer than their system's default abort timeout***.

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

88. *Eggert* further explains that if "[a] TCP implementation" on a device "does not support the TCP [ATO]," then the device "SHOULD silently ignore it":

> ***A TCP implementation that does not support the TCP Abort Timeout Option SHOULD silently ignore it [2].*** This causes the connection to use the default abort timeout, thus ensuring interoperability.

(*Id.*, 7 (excerpted with emphasis added).)

89. Thus, *Eggert* provides a new TCP option to the traditional TCP implementation where devices configured according to *Eggert* vary from the "traditional TCP implementation" in terms of the ATO negotiation. As a result, a connection using *Eggert*'s ATO would be a TCP-variant connection and at least the segments used for establishing such a TCP-variant connection (e.g., the segments using the TCP ATO) would be TCP-variant packets.

90. Figure 2 of *Eggert*, below, illustrates two different ways that the first node receives a TCP-variant packet during the exchange of three messages (3-way

handshake) in *Eggert*'s TCP-variant protocol for negotiating a per-connection abort timeout.

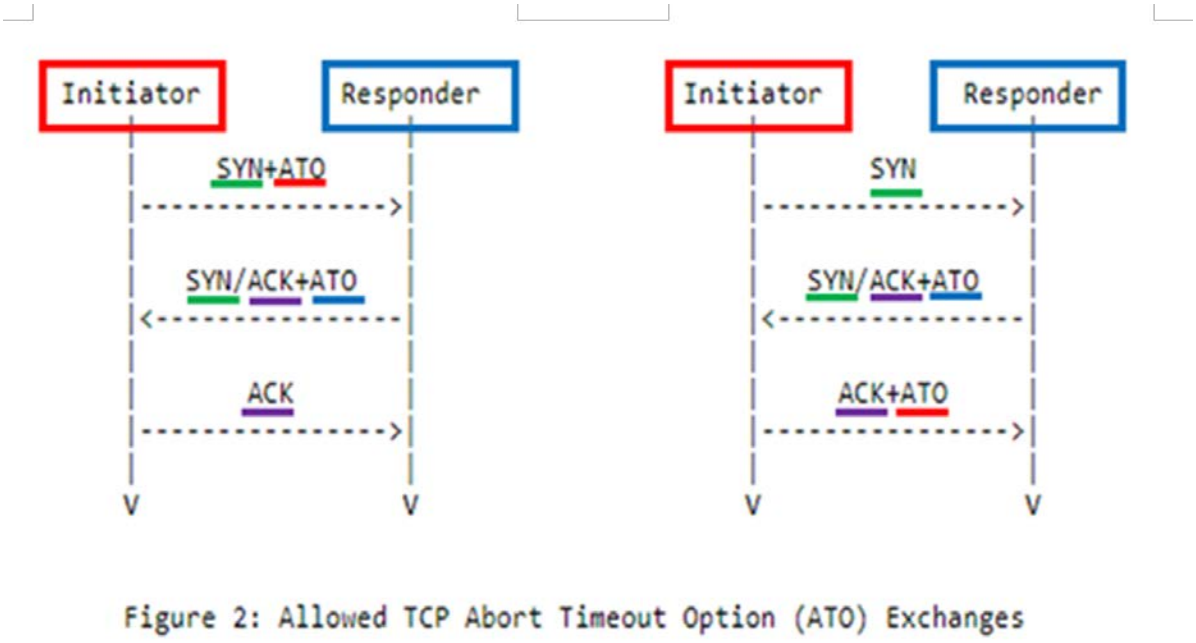


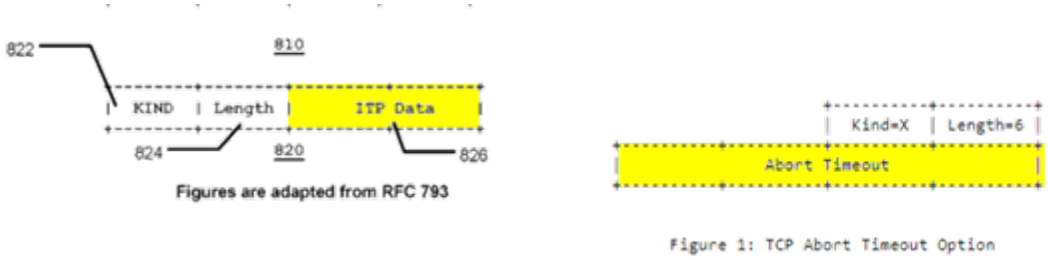
Figure 2: Allowed TCP Abort Timeout Option (ATO) Exchanges

(*Id.*, FIG. 2 (annotated).)

91. In the first way, as shown on the left-hand side of Figure 2 (when the **initiator** initiates an abort timeout negotiation), *Eggert* teaches that the **initiator** (e.g., client node) sends the **responder** (e.g., server node) a "SYN+ATO" segment. (*Id.*, FIG. 2; *see also id.*, 5–7.) The ATO in the segment indicates that this segment contains an abort timeout. (*Id.*, 3.) *Eggert* explains that when the **responder** (e.g., server node) accepts the **initiator's** offered abort timeout, it must echo the offered timeout value in the ATO it sends. (*Id.*, 5–7.) That is, the **initiator** receives an "SYN/ACK+ATO" segment ("TCP-variant packet") from the **responder**. (*Id.*)

92. In the second way, as shown on the right-hand side of Figure 2 (when the **responder** initiates an abort timeout negotiation), *Eggert* teaches that the **initiator** (client node) receives an "SYN/ACK+ATO" segment ("TCP-variant packet") from a **responder** (server node). (*Id.*, 5–7.)

93. Like the '774 patent, *Eggert*'s ATO is also incorporated in the TCP header and arranged in a similar format. (*Compare* Ex. 1001, 15:52-60, FIG. 8 *with* Ex. 1006, 3–5, FIG. 1.) As shown below, both formats use a KIND sub-field to indicate the new option, a TCP option length sub-field, and a data portion (YELLOW) of the TCP option containing the timeout value. (*Compare* Ex. 1001, FIG. 8 *with* Ex. 1006, FIG. 1.)



(Ex. 1001, FIG. 8 (annotated) (left); Ex. 1006, FIG. 1 (annotated) (right).)

IX. THE ’454 APPLICATION AND THE ’402 APPLICATION DO NOT SUPPORT THE FEATURES OF CLAIM 42

94. I have reviewed the U.S. Patent Application No. 12/714,454 (“the ’454 application) and the U.S. Patent Application No. 13/477,402 (“the ’402 application”), which I understand are patent applications to which the ’774 patent claims priority. (Ex. 1001, cover.) I have been asked to consider whether a person of ordinary skill in the art would have understood that the named inventor of ’774 patent was in possession of the subject matter recited in claim 42 of the ’774 patent based on the disclosures provided by the’454 application and the ’402 application. In my opinion, a person of ordinary skill in the art reviewing the’454 application and the ’402 application would *not* have understood that the named inventor was in possession of the subject matter recited in claim 42.

95. Independent claim 42 of the ’774 patent recites features concerning a “non-TCP connection” and a “non-TCP packet.” (Ex., 1001, 29:10–23.) For example, claim 42 recites:

> A method comprising:
>
> using at least a portion of a first node:
>
> receiving information for use in: detecting a time period that results in a ***non-TCP connection*** being subject to at least partial deactivation, and sending a ***non-TCP packet*** that is based on

the information and that includes a time period parameter field identifying metadata; and

sending, to a second node and for setting up the ***non-TCP connection***, the ***non-TCP packet*** to provide the metadata to the second node, for use by the second node in determining a timeout attribute associated with the ***non-TCP connection***.

(Ex. 1001, 29:10–23).[5] Additionally, dependent claim 67, which depends directly from claim 43 and indirectly from claim 42, further explains that "***the non-TCP connection***," which is recited in claim 42, "***is not*** a TCP-extension." (*Id.*, 33:47–48 (claim 67), 29:24-30 (claim 43).)

96. As I discuss below, it is my opinion that neither the '454 application nor the '402 application describe a "non-TCP connection" or "non-TCP packet" in the manner recited by claim 42. (*See generally* Ex. 1038 ('454 application), 99–106 (abstract and claims) 111–118 (figures), 120–165 (specification); Ex. 1039 ('402 application), 85 (abstract), 90–103 (claims and figures), 106–151 (specification).) It is also my opinion that the material incorporated by reference in the '454 and '402 applications also do not describe a "non-TCP connection" or "non-TCP packet" in the manner recited by claim 42.

[5] 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.

A. The ’454 Application (Ex. 1038)

97. In my opinion the disclosure of the ’454 application (including the claims) is limited to describing TCP connections (including extensions and variants thereof). However, the ’454 application does not provide any disclosure that describes to a person of ordinary skill in the art a “non-TCP connection” like that required by claim 42 of the ’774 patent.

98. For example, I determined that the ’454 application uses the term “TCP” nearly 400 times—e.g., about 310 times in the specification, 72 times in claims, 3 times in the Abstract, and 14 times in the corresponding figures/drawings. (*See generally* Ex. 1038 (’454 application), 99–106, 111–118, 120–165.) I was unable to locate a single instance in the ’454 application (e.g., its specification, abstract, drawings and claims) that refers to a “non-TCP” connection or packet or even variants.

99. In fact, both the Abstract and “Summary” of the as-filed ’454 application describes the disclosed invention as relating to a “***TCP connection***” and “***TCP packet***” and not to a ***non-TCP*** connection or packet (which is required by claim 42 of the ’774 patent). For example, the ’454 application recites:

> [0008] 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 detecting a first idle time period header, in the first packet, 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, based on the metadata, by the second node a timeout attribute associated with the ***TCP connection***.

(Ex. 1038, 122 (¶[0008]) (emphasis added), 99 (Abstract) (describing same).)

[0009] Further, a system for sharing information for detecting an idle **TCP connection** is described. The system includes an execution environment including an instruction processing unit configured to process an instruction included in at least one of a net inport component, an idle time period option handler component, and an option attribute handler component. The system includes the net in-port component configured for receiving, by a second node from a first node, a first **transmission control protocol (TCP) packe**t in a **TCP connection**. The system further includes the idle time period option handler component configured for detecting a first idle time period header, in the first packet, 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 system still further includes the option attribute handler component configured for modifying, based

on the metadata, by the second node a timeout attribute associated with the **TCP connection**

(Ex. 1038, 122 (¶[0009]) (emphasis added).)

[0010] In another aspect, a method for sharing information for detecting an idle **TCP connection** is described that includes receiving, by a first node, first idle information for detecting a first idle time period during which no **TCP packet** including data in a first data stream sent in the **TCP connection** by a second node is received by the first node. The method further includes generating a **TCP packet** including a first idle time period header identifying metadata for the first idle time period based on the first idle information. The method still further includes sending the **TCP packet** in the **TCP connection** to the second node to provide the metadata for the first idle time period to the second node. The method also includes detecting the first idle time period based on the first idle information. The method additionally includes deactivating the **TCP connection** in response to detecting the first idle time period.

(Ex. 1038, 122 (¶[0010]) (emphasis added).)

[0011] Still further, a system for sharing information for detecting an idle **TCP connection** is described. The system includes an execution environment including an instruction processing unit configured to process an instruction included in at least one of an idle time period policy component, a packet

> generator component, a net out-port component, an idle time period monitor component, and a connection state component. The system includes the idle time period policy component configured for receiving, by a first node, first idle information for detecting a first idle time period during which no **TCP packet** including data in a first data stream sent in the **TCP connection** by a second node is received by the first node. The system includes the packet generator component configured for generating a **TCP packet** including a first idle time period header identifying metadata for the first idle time period based on the first idle information. The system still further includes the net out-port component configured for sending the **TCP packet** in the **TCP connection** to the second node to provide the metadata for the first idle time period to the second node. The system includes the idle time period monitor component configured for detecting the first idle time period based on the first idle information. The system includes the connection state component configured for deactivating the **TCP connection** in response to detecting the first idle time period.

(Ex. 1038, 122 (¶[0011]) (emphasis added).)

100. Furthermore, the descriptions of the figures in the ’454 application likewise focus on a “TCP connection.” (*Id.*, 125–126 (¶¶[0013]–[0020].) Similarly, the detailed description of the application describes various aspects of

the alleged invention again only in terms of a "TCP connection" and/or "TCP packet." 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").)

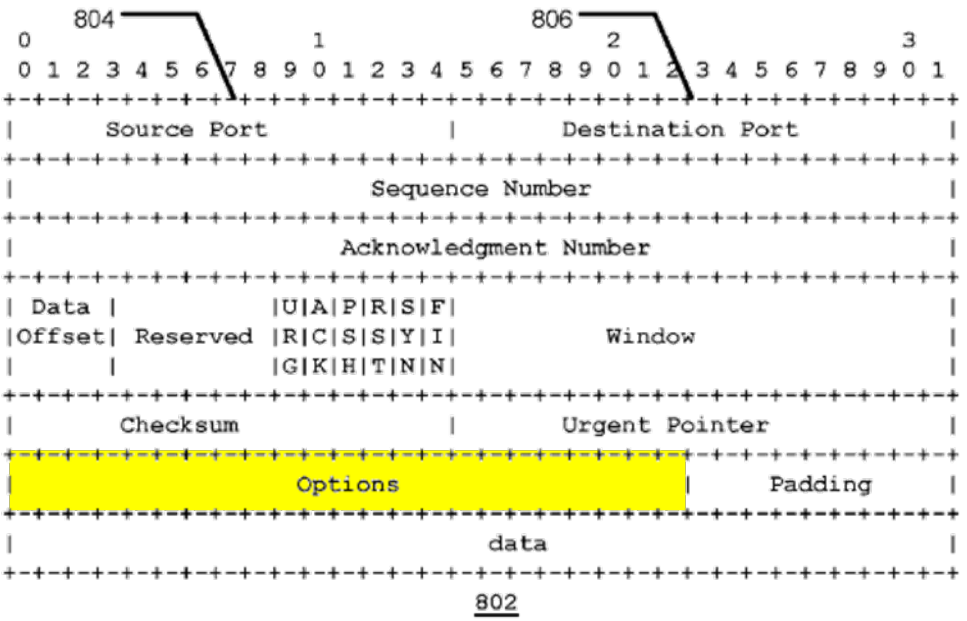
101. The '454 patent also discusses that "[a]n equivalent or analog of an ITP header may be included in a footer of a protocol packet in **an extension**

and/or variant of the current TCP" (Ex. 1038, 147 (¶[0080] (reproduced below)). It is my opinion that a person of ordinary skill in the art would have understood that such extensions or variants are 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]).)

102. Further, if an embodiment is disclosed in relation to these features (e.g., an extension and/or variant of the current TCP), in my opinion a person of ordinary skill in the art would have understood that the embodiment would relate to using the TCP options field in a conventional TCP packet structure provided by RFC 793 in order to share the idle time information between nodes. (Ex. 1038, 15:38–58.) A person of ordinary skill in the art would have understood that Figure 8, annotated and reproduced below, which is "adapted from RFC 793," illustrates such a configuration. (*Id.*, FIG. 8, 6:42–44, 15:38–58.)



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

103. Thus, it is my opinion that the disclosed extension or variant of "the current TCP" do not describe non-TCP aspects, especially when viewed in context of the disclosure in the '454 application, as I explained above, which is entirely focused on TCP-based features and not any "non-TCP" features.

104. Furthermore, it is my opinion that a person of ordinary skill in the art would have understood that "an extension … of the current TCP" as disclosed in ¶[0080] above cannot encompass a "non-TCP" feature. Such an understanding is completely consistent with claim 67 of the '774 patent, which makes clear that "***the non-TCP connection***" of claim 42 "***is not*** a TCP-extension." (*Id.*, 33:47–48 (claim 67).)

105. I have been asked to assume that the "non-TCP" feature may encompass TCP-extensions based on claim 67. I disagree with such an understanding, however in my opinion in this scenario, a person of ordinary skill in the art would have understood that "non-TCP" as recited in claim 42 must be broader in its scope that just covering a TCP-extension. So in such a scenario, a person of ordinary skill in the art would expect that the "non-TCP connection" as recited in claim 42 must covers things other than a TCP-extension. However, as I have discussed above in this section, nowhere does the specification of the '454 application indicate any other protocols (or connections) that could be considered "non-TCP."

106. Similarly, a person of ordinary skill in the art would have understood that the "variant of the current TCP" as disclosed in ¶[0080] above also cannot encompass the claimed "non-TCP" connection or packet features. Such an understanding is also confirmed by the '774 patent that includes claims that separately recite "TCP-variant" and "non-TCP." (*See* Ex. 1001, 24:22–41 (TCP-variant based claim 1), 29:10–23 (non-TCP based claim 42).)

107. It is my opinion that the '774 patent's separate use of these different claim terms demonstrates the distinction the '774 patent makes between a "variant" of TCP and "non-TCP." To be sure, this same distinction between the "TCP-variant" and "non-TCP" terms is also found in the other patents included in the

’774 patent family. (*See e.g.*, Ex. 1010 (’995 patent), 23:4–21 (TCP-variant-based claim 1), 24:33–51 (TCP-variant-based claim 15), 25:36–61 (non-TCP-based claim 25), 26:15–42 (non-TCP-based claim 28); Ex. 1036 (’564 patent), 23:5–21 (TCP-variant-based claim 1), 24:29–47 (TCP-variant-based claim 16), 25:27–53 (non-TCP-based claim 24), 26:6–29 (non-TCP-based claim 28).)

108. Accordingly, it is my opinion for at least the reasons I have discussed above that a person of ordinary skill in the art would have concluded that the ’454 application does not provide any support for the “non-TCP connection” and “non-TCP packet” features recited in claim 42 of the ’774 patent.

B. The ’402 Application (Ex. 1039)

109. In my opinion the disclosure of the ’402 application (including the claims) is limited to describing TCP connections (including extensions and variants thereof). The ’402 application, however, does not provide any disclosure that describes to a person of ordinary skill in the art a “non-TCP connection” like that required by claim 42 of the ’774 patent.

110. For example, the ’402 application is identified to be a continuation of ’454 application, which I have discussed above. (E.g., Ex. 1001, 1:30–39; Ex. 1039 (’402 application), 87, 106 (¶[0001]).) Because these application are related, I also reviewed a computer-generated redline between the between the ’454 and ’402 applications to determine the differences. (Ex. 1040 (the computer-generated redline illustrates the similarities in BLACK and differences (BLUE for what is only present in the ’402 application and RED for what is only present in the ’454 application) between the ’454 and ’402 applications).) Based on this review, it is my opinion that the ’402 application includes the same substantive disclosure as the ’454 application.

111. A person of ordinary skill in the art would have understood that the ’402 application’s specification, figures, abstract, and as-filed claims, just like the ’454 application, do not describe any “non-TCP connection” or “non-TCP packet” features. (*See generally* Ex. 1039 (’402 application).)

112. Thus, for the same reasons I have explained above for the ’454 application, a person of ordinary skill in the art would have concluded that the ’402 application does not provide any support for the “non-TCP connection” and “non-TCP packet” features recited in claim 42 of the ’774 patent.

C. The ’063 Application (Ex. 1041) and RFC 793 (Ex. 1008) Incorporated by Reference

113. In my opinion the disclosures of the U.S. Application No. 12/714,063 (the ’063 application) (Ex. 1041 (’063 application)) (including the claims) and RFC 793 titled “Transmission Control Protocol, DARPA Internet Program Internet Protocol Specification” do not provide any disclosure that describes to a person of ordinary skill in the art a “non-TCP connection” (Ex. 1008 (RFC 793) like that required by claim 42 of the ’774 patent.

114. I reviewed these two disclosures because both the ’454 and ’402 applications appear to incorporate both the ’063 application and RFC 793. (*E.g.*, Ex. 1038 (’454 application), 120 (¶¶[0001]–[0002]) , 90 (IDS specifying RFC 793); Ex. 1039 (’402 application), 106 (¶¶[0002]–[0003]).)

115. Because the ’063 application shares a similar specification to the ’454 application, I also reviewed a computer-generated redline between the between the ’454 and ’063 applications to determine the differences. (*See generally* Ex. 1041, 137–201; Ex. 1042 (the computer-generated redline illustrates the similarities in

BLACK and differences (BLUE for what is only present in the '063 application and RED for what is only present in the '454 application) between the '454 and '063 applications).) Based on this review, it is my opinion that the '063 application does not disclose the "non-TCP connection" or "non-TCP packet" features like that recited in claim 42 of the '774 patent.

116. RFC 793 dates back to 1981 (Ex. 1008, 1) and defined TCP at that time (*id.*). However, based on my review, RFC 793 does not describe or mention the "non-TCP connection" and "non-TCP packet" features like that recited in claim 42. For example, 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.

(Ex. 1008, 4.)

117. Thus, assuming that the material disclosed in RFC 793 and also in the '063 application are incorporated into either the '454 and '402 application disclosures, it is my opinion that a person of ordinary skill in the art would have understood that none of the material incorporated by reference supports the claimed "non-TCP connection" and "non-TCP packet" features recited in claim 42 of the '774 patent.

118. Thus, it is my opinion that neither the ’454 and ’402 applications convey to a person of ordinary skill in the art that the named inventor had possession of the “non-TCP connection” and/or “non-TCP packet” features recited in claim 42 at the relevant time. As I explained above, neither the ’454 nor ’402 application, along with the purported incorporated materials, describes a “non-TCP connection” and/or a “non-TCP packet” in any respect, let alone in the context of the claimed “detecting a time period that results in a *non-TCP connection* being subject to at least partial deactivation,” “sending a *non-TCP packet* that is based on the information and that includes a time period parameter field identifying metadata,” or “sending, to a second node and for setting up *the non-TCP connection*, *the non-TCP packet* to provide the metadata to the second node, for use by the second node in determining timeout attribute associated with *the non-TCP connection*,” as recited in claim 42 of the ’774 patent.

119. The above understanding is consistent with the prosecution history of the ’774 patent family, as the first textual appearance of the term “non-TCP” connection in the family of applications for the ’774 patent was in U.S. Patent Application No. 15/694,802 (“the ’802 application”) filed September 3, 2017. (Ex. 1043 (’802 application).) The ’802 application is a continuation-in-part application of U.S. Patent Application No. 14/667,642 (“the ’642 application”)’ (Ex. 1044 (’642 application)), which is a continuation-in-part of the ’402

application.[6] Based on my review, the ’802 application introduced the non-TCP connection and non-TCP packet terms only in its Summary section of the specification and in the claims. (Ex. 1043 (’802 application), 17–18 (¶¶[0012]–[0013], 70–77; *see also id.*, 100 (substitute specification contains no new matter), 105–106 (¶¶[0012]–[0013]).)

X. A PERSON OF ORDINARY SKILL IN THE ART WOULD NOT HAVE UNDERSTOOD THE SCOPE OF CLAIM 1 OF THE ’774 PATENT WITH REASONABLE CERTAINTY

120. Based on my review of the claims of the ’774 patent in light of the specification, drawings, and prosecution history of the ’811 application (from which the ’774 patent issued), it is apparent that claim 1 of the ’774 patent include a number of terms related to the “TCP-variant” features that are not explained elsewhere in the specification, drawings, and prosecution history of the ’811 application to a degree sufficient to inform a person of ordinary skill in the art as to the scope of those claims with reasonable certainty.

121. For example, the following terms, which play a prominent role in the claims, are not explained in the specification, drawings, and prosecution history of

[6] Based on my review the ’642 application only mentions “TCP” (Ex. 1044, 126 (¶[0048]), *see also id.*, 109 (¶¶[0003]–[0004], 162–164 (¶[0128]), and does not refer to a “non-TCP” connection or packet.

the ’811 application: “TCP-variant connection” and “TCP-variant packet.” Indeed, they appear nowhere outside of the claims of the ’811 application. Because these claim terms are not explained in the disclosure of the ’811 application and its prosecution history, their meaning in the context of the claims is unclear to a person of ordinary skill in the art. As discussed below, a person of ordinary skill in the art would not have been informed about the scope of the claims with reasonable certainty, and the lack of support for many of the claim terms in the specification is a contributing factor to the lack of clarity.

122. As demonstrated below, a person of ordinary skill in the art would not have been able to determine what encompasses the claimed “TCP-variant connection” and “TCP-variant packet” of claim 1 with reasonable certainty when this claim is read in light of the disclosure of the ’811 application and its prosecution history. In my opinion, a person of ordinary skill in the art would have found that such lack of clarity results in the unanswered questions as to the scope of the invention recited in claim 1. For example, a person of ordinary skill in the art would have at least these two questions regarding the scope of the claims:

- From what TCP baseline does a person of ordinary skill in the art measure that undefined variance?
- What qualifies as a “variant” from TCP including what type of variance and how much (or little) variance from the undefined baseline “TCP” is required

to constitute a "TCP-variant"?

Moreover, other claims of the '774 patent and claims in the '774 patent family further demonstrate the ambiguity introduced by the "TCP-variant" terms. For all the reasons that I discuss below, it is my opinion that a person of ordinary skill in the art would not have been able to determine the scope of "TCP-variant connection" and "TCP-variant packet" which are recited in claim 1 with reasonable certainty.

1. A Person Of Ordinary Skill In The Art Would Have Understood That The '774 Patent's Claim Language, Specification, And File History Do Not Describe Or Define The "TCP-Variant" Features

123. In my opinion, based on my review of claim 1 of the '774 patent in light of the specification, drawings, and prosecution history of the '811 application, a person of ordinary skill in the art would not have been able to determine the scope of "TCP-variant" recited in claim 1 with reasonable certainty. The "TCP-variant" term found in claim 1 is recited in the context of a "TCP-variant" packet and connection. Claim 1 recites:

> 1. An apparatus comprising:
>
> a non-transitory memory storing instructions; and
>
> one or more processors in communication with the non-transitory memory, wherein the one or more processors execute the instructions for:

> receiving, by a second node from a first node, a ***transmission control protocol (TCP)-variant packet*** in advance of a ***TCP-variant connection*** being established;
>
> detecting a time period parameter field in the ***TCP-variant packet***;
>
> identifying metadata in the time period parameter field for a time period, during which no packet is received from the first node in a data stream of the ***TCP-variant connection*** and processed by the second node to keep the ***TCP-variant connection*** active; and
>
> calculating, by the second node and based on the metadata, a timeout attribute associated with the ***TCP-variant connection*** for being used to at least partially close the ***TCP-variant connection***.

(Ex. 1001, 24:22–41.) Other than reciting the term ***variant***, the language of claim 1 (or any of its dependents) does not define or describe what is meant by a ***variant*** of a TCP connection or packet. (*Id*.; *see also id.*, 24:42–29:9 (dependent claims 2–41).) Further, a person of ordinary skill in the art would have understood that the specification does not provide the required clarity for the term ***variant***.

124. For example, the ’774 patent specification describes features to address an alleged “need…for sharing information for detecting ***an idle TCP connection***.” (Ex. 1001, 2:37–39 (emphasis added).) Also the ’774 patent’s “Summary” discloses methods and systems concerning an “***idle TCP connection***” that refer to a “***TCP connection***” and “***TCP packet***.” (*Id.*, 2:40–6:6 (emphasis added).) Based on my review, the detailed description of the ’774 patent does the

same (i.e., only refers to TCP connections and TCP packets). As I have explained above, the '774 patent's description describes a process for detecting an idle ***TCP connection*** (not a TCP-variant one). (*See* my discussion above at Section VIII ('774 patent overview); Ex. 1001, FIG. 7, 8:66–9:13, 12:12–18, 12:40–45.) The '774 patent further describes a "TCP packet" including an "ITP header" that "may be included in a packet exchanged during setup of TCP connection" (Ex. 1001, FIG. 8, 14:63–66) that has a configuration "adapted from RFC 793" (*id.*, 15:38–67, FIG. 8).

125. Based on my review, the only place where the specification mentions a TCP variant concept is in the context of the ITP header 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***.

(Ex. 1001, 16:1–8 (emphasis added).)

126. In my opinion the above disclosure (or any other portion of the specification) does not provide any discussion or details that would inform a person of ordinary skill in the art with reasonable certainty as to the scope of a

"TCP-variant packet" (e.g., how much variance from a TCP packet is permitted and what types of packets are included within the scope of a TCP-variant packet). Further the above disclosure (or any other portion of the specification) does not mention or provide any details that would inform a person of ordinary skill in the art with reasonable certainty as to the scope of a "TCP-variant connection."[7] For

[7] In my opinion, a person of ordinary skill in the art would have understood that the whole concept of what qualifies as a "variant" in the context of the specification of the '774 patent leaves questions as to its proper scope. For example, a person of ordinary skill in the art would not have been informed with reasonable certainty whether a "variant" of TCP requires a modification of existing functionality from what is done in "the current TCP" (which itself is undefined as I discuss below). If a "variant" of TCP does not require a modification of existing functionality from what is done in "the current TCP", a person of ordinary skill in the art would not have been informed with reasonable certainty whether a protocol that includes additional functionality not present in the undefined "current TCP" qualifies as a "variant" of the undefined "current TCP" or an "extension" of the undefined "current TCP." In my opinion, such guidance is needed to inform a person of ordinary skill in the art regarding the scope of the "TCP-variant" features, but such

instance, the "variant of the current TCP" language recited above relates to a packet and does not describe or give any guidance to a person of ordinary skill in the art as to the following questions:

- What protocols would encompass a "TCP-variant connection"?
- What types of variances from a TCP connection would encompass a "TCP-variant connection"?
- How much or little variance would encompass a "TCP-variant connection"?

Accordingly, it is my opinion that a person of ordinary skill in the art would have understood that the specification does not provide the required clarity for the term ***variant***.

127. Based on my review, the file history of the '774 patent (and its relevant parent applications) also do not offer any assistance to a person of ordinary skill in the art in ascertaining what is meant by the claimed "TCP-variant" features. (*See generally* Ex. 1004; Ex. 1043; Ex. 1045.) If anything, as I further discuss below in Section X.A.2.c, other claims pursued during prosecution (now issued) further demonstrate the lack of clarity of the claimed "TCP-variant" features in claim 1. (*See* my discussion below in Section X.A.2.c.)

guidance is altogether missing from the '774 patent disclosure as I discuss below in Section X.2.

2. The Intrinsic Record Does Not Inform A Person Of Ordinary Skill In The Art About The Scope Of The Claimed "TCP-Variant" Features Recited With Reasonable Certainty

128. It is my opinion that a person of ordinary skill in the art would not have been able to ascertain with reasonable certainty the scope of the "TCP-variant" features recited in claim 1 for several reasons. First, the intrinsic record for the '774 patent does not inform a person of ordinary skill in the art as to what is meant by the "TCP" aspects of the "TCP-variant" features recited in claim 1. Second, the intrinsic record does not provide guidance to inform a person of ordinary skill in the art with reasonable certainty as to what type of (or how much) "variance" from the "TCP" would encompass the claimed "TCP-variant" features of claim 1. Third, other claims in the '774 patent further demonstrate the ambiguity introduced by the "TCP-variant" terms and confirm that a person of ordinary skill in the art would not be able to ascertain with reasonable certainty the scope of the "TCP-variant" features recited in claim 1.

a) The Intrinsic Record Does Not Inform About the Scope of the [Current] TCP From Which The Claimed TCP-Variant Features Must Be Considered

129. To understand with reasonable certainty the scope of the claimed "TCP-variant" features, a person of ordinary skill in the art would need to understand the scope of the "TCP" from which the variance is based. Here, while the specification mentions "variant of the ***current TCP***" (Ex. 1001, 16:4–7), the

’774 patent does not provide guidance to a person of ordinary skill in the art in determining the scope of claim 1 with reasonable certainty because the ’774 patent does not describe what is meant by “current TCP.” That is, the ’774 patent specification does not identify what the “current TCP” is or explain any mechanism on how to identify what is or is not a “current TCP”, and thus in turn, what is a “variant of the current TCP,” or even a “TCP-variant connection” or “TCP-variant packet.”

130. For example, the ’774 patent specification states there are “various implementations” of TCP, with no further discussion or support (Ex. 1001, 1:55–58), and also references RFC 793 (“Transmission Control Protocol, DARPA Internet Program Internet Protocol Specification”) (*id.*, 1:61–67) to provide details on TCP functionality. (*Id.*, 1:55–2:5 (excerpted and reproduced below), 11:47–48 (“Detailed information on the operation of TCP is included in RFC 793”), 14:63–66, 15:13–14 (“Such other exchanges are not currently supported by the TCP as described in RFC 793.”), 15:42, 16:1–4, 20:51–57.) As such, the specification (and claims) do not inform a person of ordinary skill in the art with reasonable certainty as to whether only components/characteristics specified in RFC 793 define the “current TCP” described in the specification and “TCP” in claim 1 (Ex. 1001, 16:1–8, Claim 1 (“***TCP***-variant”)) or whether the “TCP” in “TCP-variant” encompasses unspecified implementations of TCP at the time.

> **Various implementations of the transmission control protocol (TCP)** in network nodes support a number of options that are not negotiated or even communicated between or among any of the nodes. Some of these options are included in the specification of the TCP while others are not. For example, the TCP keep-alive option is supported by a number of implementations of the TCP. It is not, however, part of the **TCP specification as described in "Request for Comments" (RFC) document RFC 793 edited by John Postel, titled "Transmission Control Protocol, DARPA Internet Program Internet Protocol Specification" (September 1981)**, which is incorporated here in its entirety by reference. …

(Ex. 1001, 1:55–2:5 (excerpted and emphasis added).)

131. Moreover, RFC 793, which is dated September 1981 (Ex. 1008, 1), predates the filing of the application for the '774 patent by over 37 years. (Ex. 1001, cover.) Also, in addition to the "***various*** implementations" of TCP acknowledged by the '774 patent, a person of ordinary skill in the art would have known that TCP implementations were defined by way of other RFC documents published after RFC 793 and before February, 2010 (the earliest filing date to which the '774 patent purports to claims priority), such as RFC 1122 (Ex. 1009) and RFC 3168 (Ex. 1046). (*See also* Ex. 1049 (excerpt of RFC 793 webpage on IETF website noting RFC 793 was updated by at least RFCs 1122 and 3168 along with RFCs 6093 (Ex. 1047) and 6528 (Ex. 1048)).)

132. A person of ordinary skill in the art would have understood that TCP implementations at the relevant time (e.g., before February 2010) also did not have to be fully compliant with any RFC document. Thus, a person of ordinary skill in the art would not have been able to reasonably ascertain the bounds and scope of the "TCP" or "current TCP" features described by the '774 patent with reasonable certainty, which is needed to frame a person of ordinary skill in the art's understanding as to the scope of the claimed "TCP-variant" features recited in claim 1 with reasonable certainty.

133. For example, the '774 patent does not provide any guidance as to whether RFC 793 and/or some other document(s) or source provides the requisite details needed to frame a person of ordinary skill in the art's understanding as to the components/characteristics of the "TCP" in a "TCP-variant connection" or "TCP-variant packet" at the relevant time of the '774 patent. To be sure, none of these details are explained in the specification, claims, or file history. (*See generally* Ex. 1004; Ex. 1043; Ex. 1045.)

134. Accordingly, for at least this reason, it is my opinion that a person of ordinary skill in the art would not have been able to ascertain with reasonable certainty the scope of the "TCP-variant" features recited in claim 1.

b) **The Intrinsic Record Does Not Inform A Person Of Ordinary Skill In The Art About The Scope Of Variance Required To Encompass The Claimed**

"TCP-Variant" Features Of Claim 1

135. In addition to, or separate from, the reasons I discuss above, it is my opinion that the '774 patent also does not provide clarity to a person of ordinary skill in the art as to *what* variation or how *much* variation from the "[current] TCP" a packet or connection needs to include (or have to be within) such that the packet or connection is considered within the scope of the claimed "TCP-variation" features of claim 1.

136. The '774 patent specification lacks objective guidance to inform a person of ordinary skill in the art about what connections and packets would qualify as a "variant" of a [current] TCP connection/packet. For example, there are simply no objective boundaries in the '774 patent to inform a person of ordinary skill in the art with reasonable certainty as to when a TCP connection is considered a "TCP-variant" versus a [current] TCP in context of the patent.

137. It is my opinion that could the example in the specification that "an ITP header may be included in a footer of a protocol packet" be considered an "extension and/or variant of the current TCP" (Ex. 1001, 16:4–8), this examples still does not inform a person of ordinary skill in the art on the objective boundaries between a current TCP, an extension of the current TCP, or a variant of the current TCP, nor based on review is there any such guidance elsewhere in the specification.

138. Further, a person of ordinary skill in the art would have understood that the "variant" term does not describe on its face what, where, or which components/characteristics are necessary or absent for a connection/packet to be a "TCP-variant connection"/TCP-variant packet" in comparison to the other identified forms of TCP-based connections or packets. And even if that example from the portion of the specification is considered, it does not provide any insight to a person of ordinary skill in the art as to the scope of a "TCP-variant connection" as recited in claim 1.

139. For instance, even if RFC 793 provides a baseline for the "TCP-variant" terms (which is unclear from the '774 patent's disclosure), the intrinsic record still does not inform a person of ordinary skill in the art about the full scope of claim 1 with reasonable certainty. The claims, specification, and file history of the '774 patent do not indicate to a person of ordinary skill in the art as to which parts of RFC 793 a "TCP-variant" connection or packet must comply with or what aspects of a connection/packet causes such connection/packet to vary from the TCP enough to encompass the claimed "TCP-variant" connection/packet.

140. As one example, a person of ordinary skill in the art would not have been informed with reasonable certainty whether establishing a "TCP-variant connection" would require a three-way handshake, a feature that is required for

establishing a TCP connection defined by RFC 793. (*See, e.g.*, Ex. 1008, 31–32; Ex. 1001, 14:63–15:10.)

141. As another example, a person of ordinary skill in the art would not have been informed with reasonable certainty whether a "TCP-variant connection" would require acknowledgment of sent packets and error recovery mechanisms (and which ones), which are features required by the TCP implementation defined in RFC 793. (*See*, *e.g.*, Ex. 1008, 8 ("Reliability" section), 13–14 (section 2.6).)

142. Said differently, it my opinion that a person of ordinary skill in the art's understanding of whether a packet/connection is a "TCP-variant" packet/connection may differ depending on what features and characteristics of a baseline TCP implementation are considered important, e.g., based on application-specific needs and related design parameters. In sum, this is a subjective inquiry for which the '774 patent provides no guidance or contours.[8]

[8] It is my opinion that the ambiguity in ascertaining the meaning of the claimed "TCP-variant" terms also exists even when a person of ordinary skill in the art would know how to identify a TCP packet/connection or a "variant" of such in context of the '774 patent by using some software tool that presents a packet/connection's attributes/characteristics. This is because such a tool would not make the ultimate determination of whether a packet/connection is a "TCP-

143. It is my opinion that a person of ordinary skill in the art would have understood that the term "variant" in the "TCP-variant [packet]/[connection]" claim element to be a relative term rather than an objective term. And as I have described above, the claims, specification, and file history do not provide any details concerning the required variance from a TCP packet or TCP connection that would inform a person of ordinary skill in the art as to the scope of the "TCP-variant" features of claim 1 with reasonable certainty.

144. That is, a person of ordinary skill in the art would have understood that the term "variant" lacks a universal meaning, and that lack of clarity extends to its use in the "TCP-variant" limitations of claim 1. That is, the '774 patent's disclosure does not inform a person of ordinary skill in the art regarding the minimum or the maximum bounds of the word "variant" in the TCP-variant terms. For example, the term "variant," viewed in light of the claims, specification, and file history of the '774 patent, does not indicate to a person of ordinary skill in the art the scope of "variant" as it would have been used by a such skilled person to delineate between a TCP-variant packet/connection and a TCP packet/connection.

variant," but rather would be left to the subjective view of a person of ordinary skill in the art viewing such attributes/characteristics.

145. Accordingly, for at least this reason, it is my opinion that a person of ordinary skill in the art would not have been able to ascertain with reasonable certainty the scope of the "TCP-variant" features recited in claim 1.

c) The Other Claims Of The '774 Patent And Also The '774 Patent's Parent Applications Demonstrate The Lack Of Reasonable Certainty As To The Scope Of The Claimed "TCP-Variant" Features Of Claim 1

146. Based on my review of the related applications to the '774 patent, the "TCP-variant" features were first introduced into the '774 patent family through claims in the '802 application (now the '995 patent). (Ex. 1043, 63-64 (introducing claims with "TCP-variant connection" and "TCP-variant packet").) From there, the "TCP-variant" features were introduced in claims in the family of applications leading up to the '811 application that issued as the '774 patent. (*See e.g.*, *id.*, 75, 232; Ex. 1045, 225; Ex. 1004, 60–62, 136–37, 224.) As a result, the '774 patent now includes claims that introduces additional uncertainty as to how a person of ordinary skill in the art could ascertain the scope of the claimed "TCP-variant" features in claim 1 of the '774 patent.

147. Specifically, some of the claims pursued during prosecution and now issued in the '774 patent demonstrate the ambiguity regarding the scope of what encompasses the "TCP-variant" features found in claim 1, such as:

- "the TCP-variant connection is ***not a TCP connection***" (Ex. 1001, 25:58–60 (claim 22) (emphasis added));
- "the TCP-variant connection is ***not a standard TCP connection***" (*id*., 25:61–63 (claim 23) (emphasis added));
- "the TCP-variant ***includes another protocol, other than TCP, that includes a variation of TCP***" (*id*., 27:26–27 (claim 40) (emphasis added));
- "the TCP-variant ***includes another protocol, without TCP in its name, that includes a variation of TCP***" (*id*., 27:28–29 (claim 40) (emphasis added));
- "the TCP-variant includes ***one or more features of TCP, and one or more features not of TCP***" (*id*., 27:30–31 (claim 40) (emphasis added));
- "the TCP-variant packet includes ***one or more features of a TCP packet, and one or more features not of a TCP packet***" (*id*., 27:32–34 (claim 40) (emphasis added)); and
- "the TCP-variant connection includes ***one or more features of a TCP connection, and one or more features not of a TCP connection***" (*id*., 27:35–37 (claim 40) (emphasis added)).

(*See also* Ex. 1004, 60–61.)

148. First, the features recited in the claim above are not mentioned or otherwise described in the '774 patent specification. (*See* my discussions above at Sections VI, VII, X.1, X.2.a–b; *see generally* Ex. 1001.) Second, in my opinion, they raise more questions than answers as to assisting a person of ordinary skill in the art in ascertaining the scope of the claimed "TCP-variant" terms with reasonable certainty.

149. For example, the '774 patent claims, file history, and the specification do not provide details that would inform a person of ordinary skill in the art with reasonable certainty as to how a "TCP-variant connection" can be a variant to a TCP connection but also "not [be] a TCP connection" as set forth in claim 22. Moreover, the inter-play between claims 22 and 23 raises ambiguities as to the scope of claim 1's "TCP-variant" features as there is no support or guidance in the '774 patent that informs a person of ordinary skill in the art with reasonable certainty as to what is meant by, and what is the difference between, a "standard" TCP connection and a non-standard TCP connection.

150. Similarly, with respect to claim 40, the '774 patent does not provide any disclosure or guidance to inform a person of ordinary skill in the art with reasonable certainty as to what it means to "include[]" another protocol other than TCP that also includes a variation of TCP. Nor does the '774 patent provide any disclosure or guidance to inform a person of ordinary skill in the art with

reasonable certainty as to what it means to have (or not have) "TCP" in the "name" of a protocol while still being considered a "variation of TCP," as alternatively set forth in claim 40.

151. I have been asked to assume the notion that "TCP-variant" somehow encompasses protocols that are not based on TCP (e.g., non-TCP connections or packets like connections and packets based on the IPX/SPX networking protocol). Under that assumption, it is my opinion that this interpretation likewise has no support in the specification of the '774 patent or any of its parent applications.[9] Moreover, it is my opinion that such a position and interpretation would conflict

[9] I have been informed that Patent Owner maps a "TCP-variant" connection/packet to a UDP-based protocol connection/packet. In my opinion, a person of ordinary skill in the art would not have understood a UDP-based protocol connection/packet to be a TCP connection/packet at all. I have been informed that Patent Owner at the same time points to the same UDP-based protocol to show a "non-TCP" connection/packet. In my opinion, Patent Owner's mapping of the same UDP-based protocol connection/packet to both a "TCP-variant" connection/packet and a "non-TCP" connection/packet further demonstrates the unclear scope of the "TCP-variant" features recited in claim 1.

with the ’774 patent’s express recitation of “non-TCP” features in other claims. (Ex. 1001, 29:10–23 (claim 42).

152. Accordingly, for at least this reason, it is my opinion that a person of ordinary skill in the art would not have been able to ascertain with reasonable certainty the scope of the “TCP-variant” features recited in claim 1 based on the ’774 patent’s other claims (and file history pursuing such claims).

XI. THE PRIOR ART DISCLOSES ALL OF THE CLAIM ELEMENTS OF CLAIM 1 OF THE '774 PATENT

A. *Wookey* and *Eggert* Disclose or Suggest the Claim Elements of Claim 1

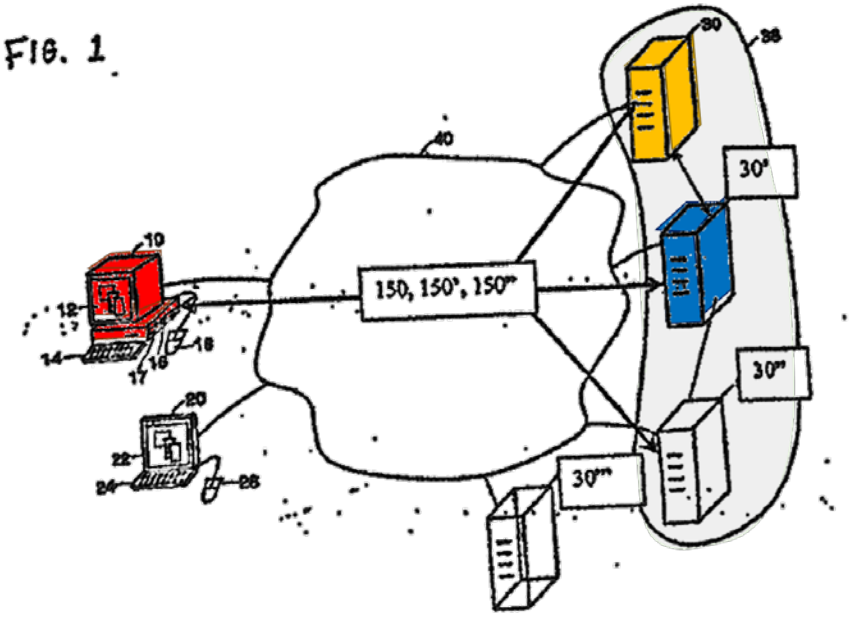
153. In my opinion, *Wookey* and *Eggert* disclose and/or suggest all of the claim elements of claim 1. Below, I address each claim element of the claims.

1. Claim 1

a) **An apparatus comprising:**

154. I have been asked to assume that the preamble of claim 1 is limiting. Under that assumption, it is my opinion that *Wookey* discloses an apparatus (and as I discuss further below, discloses and/or suggests such an apparatus that includes the features recited in claim 1 in view of *Eggert* and *RFC 793*). (*See also* my discussions above in Sections VIII.A (overview of *Wookey*).)

155. For example, *Wookey* discloses a server farm 38 ("apparatus"), which is a structure comprised of a collection of computer servers. (*See also* Ex. 1005 ¶[0002] (below), Abstract (below), FIG. 1.) Figure 1 below shows an exemplary environment in which client machines (e.g., **client machine 10**) access computer resources provided by a server farm 38 over a network connection 150. (Ex. 1005 ¶[0136] (excerpted below); *see also id.*, ¶¶[0016] (below), [0020] (below).)



(*Id.*, FIG. 1 (annotated).)

> FIG. 1 is a block diagram of one embodiment of an environment in which a client machine accesses a computing resource provided by a remote machine;

(*Id.*, ¶[0020].)

> A method for making a hypermedium page interactive, the hypermedium page displayed by a network browser, includes the step of selecting a hyperlink on the hypermedium page displayed on a client machine, the hyperlink identifying a desired computing resource. A hyperlink configuration file is retrieved, the hyperlink configuration file corresponding to the hyperlink and identifying a server machine. A client agent is started on the client machine. The client agent creates, via a terminal services session, a communication link to a

> virtual machine executing on the server identified by the hyperlink configuration file, the virtual machine executed by a hypervisor executing in the terminal services session provided by an operating system executing on the server. The client agent receives data from the virtual machine and displays, on the client machine, the received data without intervention by the network browser.

(*Id.*, Abstract.)

> The invention generally relates to providing access to computing environments. More particularly, t**he invention relates to** methods and **systems for making a hypermedium page interactive**.

(*Id.* ¶[0002] (emphasis added).)

156. *Wookey* discloses that the server farm 38 includes multiple remote machines, e.g., **remote machine 30** and **remote machine 30'**, amongst others. (*Id.* ¶[0136].) *Wookey* discloses that the remote machines are "typical computers." (*Id.* ¶[0021] ("FIGS. 1A and 1B are block diagrams depicting embodiments of typical computers useful in embodiments with remote machines or client machines"), FIGS. 1A–1B.) For example,

> A remote machine 30 such as remote machine 30, 30′, 30″, or 30‴ (hereafter referred to generally as remote machine 30) accepts connections from a user of a client machine 10. … For example, in one embodiment, the system may include multiple,

> logically-grouped remote machines 30, one or more of which is available to provide a client machine 10, 10′ access to computing resources. In these embodiments, **the logical group of remote machines may be referred to as a "server farm" or "machine farm," indicated in FIG. 1A as machine farm 38**. In some of these embodiments, the remote machines 30 may be geographically dispersed. … For example, **a machine farm 38 may include remote machines 30 physically located in geographically diverse locations around the world**, including different continents, regions of a continent, countries, regions of a country, states, regions of a state, cities, regions of a city, campuses, regions of a campus, or rooms. … **A machine farm 38 may be administered as a single entity.**

(*Id.*, ¶[0136] (excerpted with emphasis added).)

157. *Wookey* discloses, with respect to Figure 1, that the **client machine 10** may communicate with the **remote machine 30** to determine an enumeration of resources available to the **client machine 10**. (*Id.* ¶[0174] (reproduced below).) In some of the configurations contemplated for the environment in Figure 1 by *Wookey*, the **client machine 10** communicates with the **remote machine 30'** to access the identified resource from the communicated enumeration of resources. (*Id.*; *see also id.* ¶¶[0196], [0207].)

> In some embodiments, **a client machine 10 communicates with a remote machine 30 to determine an enumeration of**

> **resources available to the client machine 10 or to a user of the client machine 10**. Resources may include, without limitation, computing environments, applications, documents, and hardware resources. In another of these embodiments, the remote machine 30 provides the client machine 10 with address information associated with a remote machine 30′ hosting a resource identified by the enumeration of resources. In still **another of these embodiments, the client machine 10 communicates with the remote machine 30′ to access the identified resource**. In one embodiment, the client machine 10 executes a resource neighborhood application to communicate with the remote machines 30 and 30′. In some embodiments, each of the remote machines 30 provides the functionality required to identify and provide address information associated with a remote machine 30′ hosting a requested resource.

(*Id.* ¶[0174] (emphasis added).)

158. For example, Figures 2C and 3D below additionally show an exemplary environment in which **remote machine 30** and **remote machine 30'** are employed to permit a **client machine 10** access to a computer resource. (*E.g.*, *Id.*¶¶[0016], [0020] (reproduced above), [0196] (reproduced below).) For example,

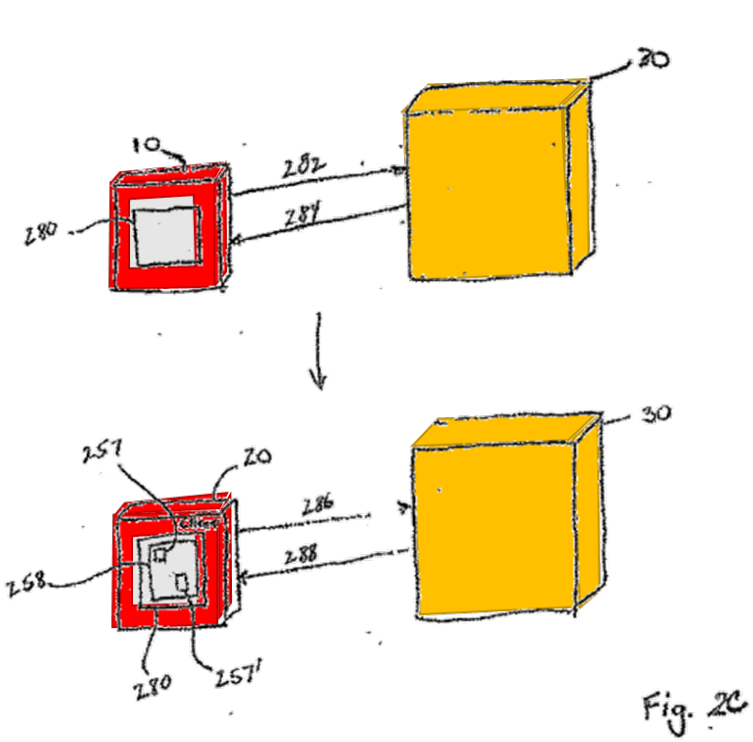


Fig. 2C

(*Id.*, FIG. 2C (annotated).)

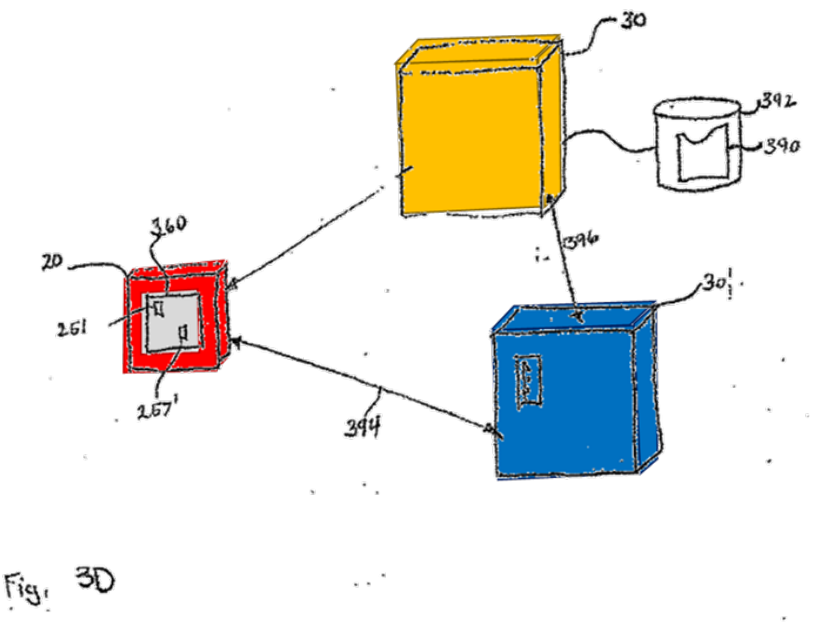


Fig. 3D

(*Id.*, FIG. 3D (each annotated).)

> In yet another aspect, **a system for making a hypermedium page interactive,** the hypermedium page displayed by a network browser **comprises a client machine, a network server**, and a client agent. The client machine executes a browser application, said browser application displaying a hypermedium page including a hyperlink identifying a desired computing resource. The network server transmits, in response to selection of said hyperlink, a network configuration file to said client machine, said network configuration file corresponding to said identified computing resource. **The client agent executes on the client machine**, **said client agent establishing, responsive**

> **to data in said configuration file, a communications link with a server machine**, the server machine executing an operating system providing a terminal services session launching a hypervisor executing the virtual machine providing the computing resource. The client agent receives data for display from the virtual machine without intervention by said browser application.

(*Id.*, ¶[0016] (emphasis added).)

> Still referring to FIG. 2C, on**ce the client machine 10 is authenticated by the remote machine 30, the remote machine prepares and transmits to the client machine 10 an HTML page 288** that includes a Resource Neighborhood window 258 in which appears graphical icons 257, 257′ representing resources to which the client machine 10 has access. A user of client machine 10 requests access to a resource represented by icon 257 by clicking that icon 257.

(*Id.*, ¶[0196] (emphasis added).)

159. Reviewing *Wookey* in context, a person of ordinary skill in the art would have understood that *Wookey*'s disclosures (including its figures) include a few typographical errors with respect to how the client machines are labeled and discussed. For example, Figure 1 of *Wookey* illustrates two client machines "10" and "20." (*Id.*, Fig. 1.) However, the corresponding disclosure for Figure 1 refers to client machines "10" and "10' instead of "20". (*Id.*, ¶¶[0135]-[0136], [0152]-

[0154].) Also, the bottom figure in Figure 2C below shows client machine "20," but the corresponding disclosure for both the top and bottom figures in Fig. 2C describes client machine "10." (*Id.*, ¶¶[0192] ("FIG. 2C shows another embodiment of a system in which a client machine 10 initiate execution of the Resource Neighborhood application 241, in this example via the World Wide Web"), [0193] (referring to the "client machine 10"), [0195] (discussing credentials associated with "client machine 10"),[0196] (describing with respect to the bottom portion of Figure 2C that "once the client machine 10 is authenticated by the remote machine 30, the remote machine prepares and transmits to the client machine 10 an HTML page 288 . . ").) Further, Figure 3D illustrates client machine "20," but the corresponding disclosure in the specification relating to that figure refers to client machine "10." (*Id.*, ¶¶[0207] ("FIG. 3D shows one embodiment of a system of communication between the client machine 10, a remote machine 30 … and a second remote machine 30'"), [0211] (referring to "client 10"), [0213] (referring to "client machine 10"), [0216] (referring to "client machine 10" in context of icon selections and creating "a connection to the remote machine hosting the resource" (e.g., remote machine 30')).) *Wookey* also describes client machines "10" and "20" (*see, e.g., id.*, ¶[0171]) and machines "10" and "10'" (*see, e.g., id.*, ¶¶ [0177]-[0179]) in a similar manner. Accordingly, in my opinion, a person of ordinary skill in the art would have understood in context that

Wookey's disclosures regarding client machine 10, client machine 10' and client machine 20 (even with the references to different client machine labels) apply equally to each other given that *Wookey* interchangeably refers to the client machine as "10," "10'"or "20." (*See, e.g.*, *id.*, ¶¶[0135], [0207] (referencing client machine as "10" in Figure 3D which shows client machine "20"), [0245]-[0247] (referencing client machine as "20" regarding Figure 4 which shows client machine "10"), FIGS. 1, 2C, 3D, 4.)

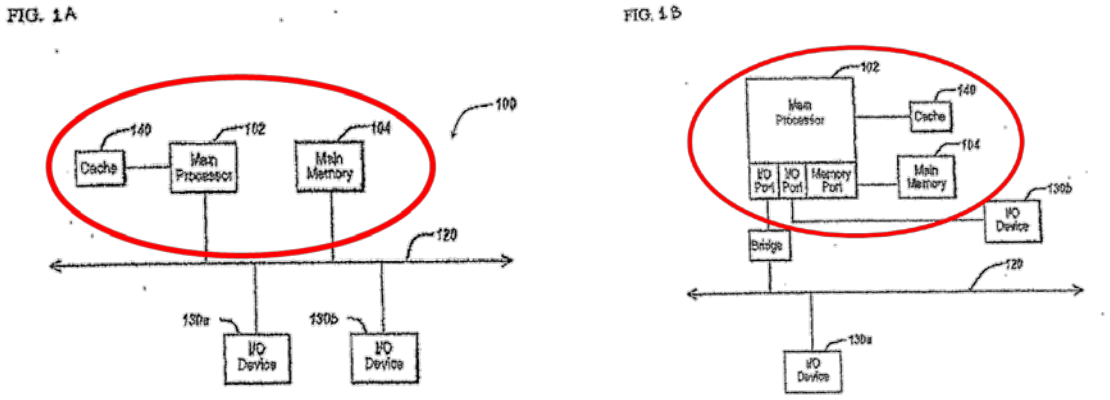
160. Thus, in my opinion, *Wookey*'s server farm, which is comprised of multiple remote machines (e.g., **remote machine 30** and **remote machine 30'**), is an apparatus like that claimed because, as I discuss further below, it includes a server computer having the features like those recited in claim 1. (*See* my discussions below at Sections XI.A.1.b–f.)

b) **a non-transitory memory storing instructions; and one or more processors in communication with the non-transitory memory, wherein the one or more processors execute the instructions for:**

161. In my opinion, *Wookey* discloses the features of this element.

162. For example, as I explained above, *Wookey* discloses that its system includes **client machine 10** which is a computer. (Ex. 1005 ¶[0160] ("[T]he client machines 10[] [is] provided as computers or computer servers."); *see also* my discussions above at Section XI.A.1.a).) As shown in Figures 1A and 1B below,

client machine 10 includes a main processor 102 ("one or more processors") configured to communicate via system bus 120 with a main memory unit 104 and a cache memory 140 (individually or collectively, the claimed "non-transitory main memory.") (Ex. 1005 ¶¶[0161]–[0165] (reproduced below).)



(*Id.*, FIGS. 1A-1B (annotated).) In particular, figures 1A and 1B (as I annotate above) show block diagrams of typical computer architectures for both client and remote machines:

> FIGS. 1A and 1B are block diagrams **depicting embodiments of typical computers useful in embodiments with remote machines or client machines**;

(*Id.*, ¶[0021] (excerpted with emphasis added).)

163. As shown in Figures 1A and 1B, each computer 100 includes a central processing unit (CPU) 102, main memory unit 104, and cache memory 140. (*Id.* ¶¶[0161] (excerpted and reproduced below), [0163] (excerpted and reproduced

below), [0164] (excerpted and reproduced below).)[10] The CPU 102 "responds to and processes ***instructions fetched from*** the main memory unit 104." (*Id.* ¶[0162] (excerpted and reproduced below); *id.* ¶¶[0554], [0556], [0802], [1121], FIG. 33.) The main memory unit 104 can include one or more memory chips (e.g., SRAM) that store data accessed by CPU 102. (*Id.* ¶[0163] (reproduced below).)

> **FIGS. 1A and 1B depict block diagrams of typical computer architectures useful in those embodiments as the remote machine 30, or the client machine 10.** As shown in FIGS. 1A and 1B, **each computer 100 includes a central processing unit 102, and a main memory unit 104**. Each computer 100 may also include other optional elements, such as one or more input/output devices 130 a-130 n (generally referred to using reference numeral 130), **and a cache memory 140 in communication with the central processing unit 102**.

(*Id.*, ¶[0161] (excerpted with emphasis added).)

> **The central processing unit 102 is any logic circuitry that responds to and processes instructions fetched from the**

[10] A person of ordinary skill in the art would have understood that *Wookey* uses the terms "Main Processor 102," "microprocessor 102," "processor 102," and "central processing unit 102" interchangeably. (Ex. 1005 ¶¶[0161], [0163]–[0164], FIGS. 1A–B.)

> **main memory unit 104**. In many embodiments, the central processing unit is provided by a microprocessor unit,…

(*Id.*, ¶[0162] (excerpted with emphasis added).)

> **Main memory unit 104 may be one or more memory chips capable of storing data and allowing any storage location to be directly accessed by the microprocessor 102**, such as Static random access memory (SRAM), …, Dynamic random access memory (DRAM), … , Extended Data Output RAM (EDO RAM), … Double Data Rate SDRAM (DDR SDRAM), … or Ferroelectric RAM (FRAM).

(*Id.*, ¶[0163] (excerpted with emphasis added).)

> In the embodiment shown in FIG. 1A, **the processor 102 communicates with main memory 104 via a system bus 120** (described in more detail below). FIG. 1B depicts an embodiment of a computer system 100 in which **the processor communicates directly with main memory 104 via a memory port.** For example, in FIG. 1B, the main memory 104 may be DRDRAM

(*Id.*, ¶[0164] (emphasis added).)

164. The cache memory 140 is a fast memory type that acts as a buffer between RAM and the CPU. (*Id*. ¶[0165] (excerpted and reproduced below).)

> **FIG. 1A and FIG. 1B depict embodiments in which the main processor 102 communicates directly with cache memory 140 via a secondary bus,** sometimes referred to as a

> "backside" bus. In other embodiments, **the main processor 102 communicates with cache memory 140 using the system bus 120**. Cache memory 140 typically has a faster response time than main memory 104 and is typically provided by SRAM, BSRAM, or EDRAM.

(*Id.*, ¶[0165] (emphasis added).)

165. A person of ordinary skill in the art would have known at the relevant time that cache memory stores requested data and instructions so that they are immediately available to a processing unit when needed. (Ex. 1050, 1:15–39 (reproduced below).)[11] Therefore, such a skilled person would have understood that in the context of *Wookey*'s disclosures, the cache memory 140 would necessarily have the same features (e.g., stores data and instructions that are made available to the CPU in *Wookey*'s system so that the system can perform the functionalities that I discuss above and below for this claim). For example, it was known in the art at the time that cache memory is used for storing both instructions and data that a processor may access:

> In a basic microprocessor-based system, a single microprocessor acts as the bus controller/system master.

[11] I refer to Exhibit 1050 here and elsewhere to demonstrate the knowledge of one of ordinary skill in the art at the time.

> Typically, this microprocessor **includes on-chip cache for storing both instructions and data.** In embedded chip controllers, as well as some microprocessor-base architectures, at least some of the data cache, instruction cache, or both can reside off-chip. In any event, the cache is a high-speed (shorter access time) memory, which makes up the higher levels in the memory hierarchy and is used to reduce the memory access time and supplement the processor register space.

(Ex. 1050, 1:15–25 (emphasis added).)

> **Generally, the processor first attempts to access cache to retrieve the instructions or data required for a given operation**. If these data or instructions have already been loaded into cache, then a "cache hit" occurs and the access is performed at the shorter cache access time. **If the necessary data or instructions are not encached, a "cache miss" occurs and processor must redirect the access to system memory or some other lower-speed memory resource**. The cache is then updated by replacing selected existing encached data with the data retrieved from the lower levels. Various caching techniques are used to reduce the miss penalty and execution errors in the processor pipelines when a cache miss does occur.

(*Id.*, 1:25–39 (emphasis added).)

166. In light of the disclosures by *Wookey* and the knowledge of computing systems at the time, a person of ordinary skill in the art would have understood that

cache memory 140 and main memory unit 104 of the **client machine 10** are each non-transitory memory devices. Further because *Wookey*'s cache memory 140 and main memory unit 104 store data and instructions that are used by processor 102 to perform operations, consistent with *Wookey*'s disclosure, this supports my opinion that the cache memory 140 and main memory unit 104 (individually or collectively) is a non-transitory memory, which I understand is different from a transitory type of medium, such as a carrier wave signal transmitted over a communication line.

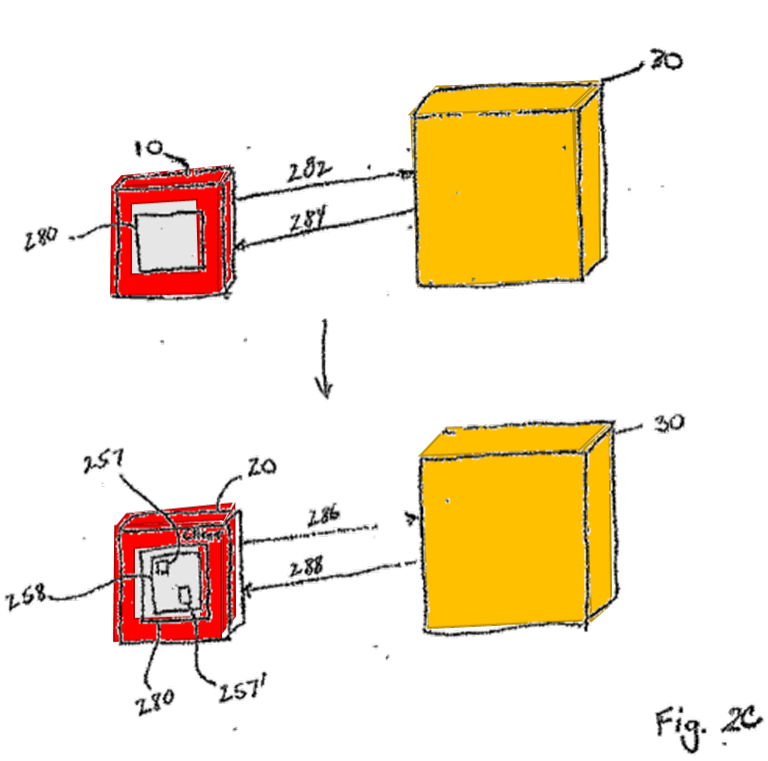
167. Therefore, it is my opinion that a person of ordinary skill in the art would have understood that **client machine 10** includes: cache memory 140 and main memory unit 104 (individually or collectively, the claimed "non-transitory memory"), each of which stores instructions; and a main processor 102 ("one or more processors") in communication with cache memory 140 and main memory unit 104 (individually or collectively, the claimed "non-transitory memory"), wherein the one or more processors execute the instructions for, recited in claim element 1.b. (*See* my discussions below at Sections XI.A.1.c–f.)

c) **receiving, by a second node from a first node, a transmission control protocol (TCP)-variant packet in advance of a TCP-variant connection being established;**

168. In my opinion, the combination of *Wookey* and *Eggert* discloses and/or suggests the features recited in claim element 1.c. For example, as I discuss below, the *Wookey-Eggert* combination discloses receiving, by **client machine 10** ("second node") from **remote machine 30'** ("first node"), a transmission control protocol (TCP)-variant packet in advance of a TCP-variant connection being established.

169. For example, **client machine 10**, as illustrated in Figure 2C below, executes a web browser application 280 (Ex. 1005 ¶[0192]) that "transmits a request 282 to access a Uniform Resource Locator (URL) address corresponding to an HTML page residing on remote machine [30]" (*id.* ¶[0193]).[12] The communication link between **client machine 10** and **remote machine 30** uses the transmission control protocol (TCP). (Ex. 1005 ¶¶[0154]–[0156], [0174], [0731]–[0732].)

[12] In my opinion, *Wookey*'s reference to "remote machine 10" in the first sentence of paragraph [0193] is a typographical error. In the same paragraph, *Wookey* identifies the same remote machine as "the remote machine 30," consistent with Figure 2C. (Ex. 1005 ¶¶[0193], [0195]–[0196] FIGS. 2C, 3D.)



(*Id*., FIG. 2C (annotated).)

> **FIG. 2C shows another embodiment of a system in which a client machine 10 initiates execution of the Resource Neighborhood application 241, in this example via the World Wide Web. A client machine 10 executes a web browser application 280**, such as NETSCAPE NAVIGATOR, manufactured by Netscape Communications, Inc. of Mountain View, Calif., INTERNET EXPLORER, manufactured by Microsoft Corporation of Redmond, Wash., or SAFARI, manufactured by Apple Computer of Cupertino, Calif.

(*Id*. ¶[0192] (excerpted with emphasis added).)

> **The client machine 10, via the web browser 280, transmits a request 282 to access a Uniform Resource Locator (URL) address corresponding to an HTML page residing on remote machine 10**. In some embodiments the first HTML page returned 284 to the client machine 10 by the remote machine 30 is an authentication page that seeks to identify the client machine 10 or the user of the client machine 10.

(*Id.* ¶[0193] (excerpted with emphasis added).)

170. **Client machine 10** is then authenticated by **remote machine 30**. (*Id.* ¶[0193]; *see also id.* ¶¶[0196] (excerpted and reproduced below), [0207]–[0214].) After authenticating **client machine 10**, **remote machine 30** may "transmit[] to the client machine 10 an HTML page 288 that includes a Resource Neighborhood window 258 in which appears graphical icons 257, 257' representing resources to which the client machine 10 has access." (Ex. 1005 ¶[0196] (excerpted and reproduced below).) The Resource Neighborhood window 258 displays graphical icons 257, 257' representing resources to which **client machine 10** has access. (*Id.* ¶[0196]; *see also id.* ¶¶[0215]–[0216].)

> [O]nce the client machine 10 is authenticated by the remote machine 30, **the remote machine prepares and transmits to the client machine 10 an HTML page 288** that includes a Resource Neighborhood window 258 in which appears graphical icons 257, 257′ representing resources to which the client machine 10 has access.

(Ex. 1005 ¶[0196] (excerpted with emphasis added).)

171. The HTML page 288 includes encoded URLs associated with icons (e.g., 257, 257') on the HTML page 288, and by clicking on one of these icons 257, 257', **client machine 10** can initiate a process for establishing a connection with **remote machine 30'** (e.g., to access a desired resource on **remote machine 30'**). (*Id*.) For example, *Wookey* explains that "each icon 257, 257′ is *associated with* ***an encoded URL that specifies***: ***the location of the resource*** (i.e., on which remote machines it is hosted or, alternatively, the address of a master remote machine, a gateway, or other remote machine 30); ***a launch command associated with the resource***; and ***a template identifying how the results of accessing the resource should be displayed*** (i.e., in a window "embedded" in the browser or in a separate window)." (Ex. 1005 ¶[0216].) *Wookey* also discloses that each encoded URL "***includes a file, or a reference to a file, that contains the information necessary for the client to create a connection to the remote machine hosting the resource***." (*Id*.) Thus, when a user *clicks* an icon 257, 257' corresponding to a resource from the displayed HTML page 288, **client machine 10** uses the encoded URL associated with the clicked icon to initiate the process for establishing a second connection (e.g., connection 394 in Figure 3D below) with a second **remote machine 30'**. (*Id*.; *see also id*. ¶[0026].)

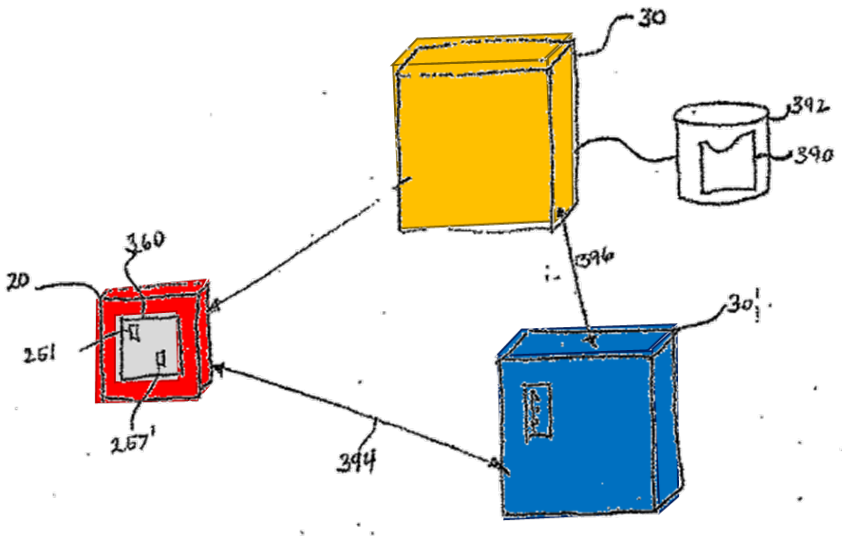


Fig. 3D

(Ex. 1005, FIG. 3D (annotated); *see also id.* ¶[0026].)

172. As I discuss below, the process for **client machine 10** establishing a second connection with **remote machine 30'** includes receiving, by **client machine 10** ("second node") from **remote machine 30'** ("first node"), a transmission control protocol (TCP)-variant packet in advance of a TCP-variant connection being established, as claimed, in the *Wookey-Eggert* combination. (*See also infra* Sections XI.A.1.d–f (discussing additional steps in the process for establishing the connection).)

173. For example, regarding the connection between **client machine 10** and **remote machine 30'**, *Wookey* provides that **client machine 10** includes an

"HTTP client agent," such as the web browser application 280, which "can use any type of protocol":

> The client agent can use any type of protocol, such as a remote display protocol, and it can be, for example, an HTTP client agent, an FTP client agent, an Oscar client agent, a Telnet client agent, an Independent Computing Architecture (ICA) client agent manufactured by Citrix Systems, Inc. of Fort Lauderdale, Fla., or a Remote Desktop Protocol (RDP) client agent manufactured by Microsoft Corporation of Redmond, Wash.

(*Id.* ¶[0159]; *see also id.* ¶[0155].)

174. For example, connection 394 can be made using various protocols, including the Virtual Network Computing (VNC) protocol. (*Id.* ¶[0216].) VNC was known by those of ordinary skill in the art at the time as a type of graphical desktop-sharing system that allows a computer to remotely access and control another computer:

> The client machine 10 establishes a connection (arrow 394) with the remote machine 30′ identified as hosting the requested resource and exchanges information regarding access to the desired resource. … In other embodiments, **the connection is made using:** the RDP protocol, manufactured by Microsoft Corp. of Redmond, Wash.; the X11 protocol; or **the Virtual Network Computing (VNC) protocol**, manufactured by AT&T Bell Labs. Thus, **the client machine 10 may display**

> **the results of accessing the resource in a window separate from the web browser 280**, or it may "embed" application output within the web browser.

(*Id.* ¶[0216] (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**. ….

(Ex. 1022 ¶[0019] (excerpted with emphasis added); *see also* my discussions above in Section V.C.)[13]

175. Regarding the type of transport protocols that VNC may use, *Wookey* explains that VNC can "run over industry standard transport protocols, such as TCP/IP, IPX/SPX, NetBEUI."

> The communications link 462 can be established by, for example, using the ICA protocol, the RDP protocol, the X11 protocol, the ***VNC protocol, or any other suitable presentation-***

[13] I refer to Exhibit 1022 here and elsewhere to demonstrate the knowledge of one of ordinary skill in the art at the time.

> ***level protocol designed to run over industry standard transport protocols, such as TCP/IP, IPX/SPX, NetBEUI***, using industry-standard network protocols, such as ISDN, frame relay, and asynchronous transfer mode (ATM)

(Ex. 1005 ¶[0225]; *see also id*. ¶¶[0155]–[0156], [0215]–[0216].)

176. *Wookey* further explains that **client machine 10** may use "a different type of protocol than the one used to send the request to the remote machine 30":

> [T]he datagram requesting an application address by a ***client machine 10 includes a request for a different type of protocol than the one used to send the request to the remote machine 30***. For example, ***the client machine 10 may make a request to the remote machine 30 using the IPX protocol and request the address of the remote machine 30' as a TCP/IP protocol address***.

(Ex. 1005 ¶[0732]; *see also id.* ¶¶[0738]–[0739], [0772].)

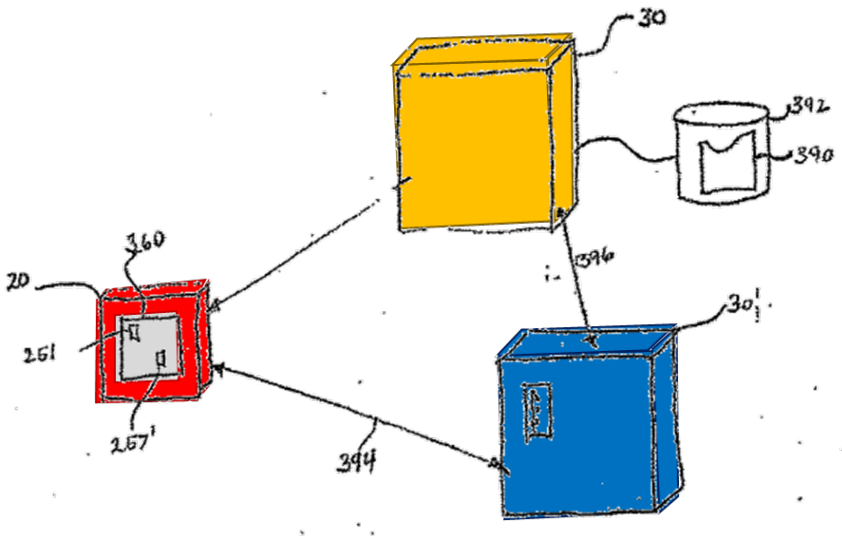


Fig. 3D

(*Id.*, FIG. 3D (annotated); *see also id.* ¶[0026].)

177. *Wookey* also identifies desired characteristics associated at least with the second connection that a person of ordinary skill in the art would have considered when choosing a protocol for that connection, such as:

- allow the "negotiate[ion] [of] the parameters under which communications will take place" (*id.*, ¶¶[0721], [0774]);
- reduce "unintentional termination of application sessions 7918 resulting from imperfect network connections" (*id.*, ¶[1134]);
- "detect a disconnection" (*id.*, ¶[1136]) and handle the disconnection (*id.*, ¶[0581]), such as when the mobile device "enters an elevator"

(*id*., ¶[1153]), the mobile device "traverse[s] network segments … that cause changes in the network address or host name … or causes the client machine 10 to disconnect" (*id*., ¶[0581]); and

- be able to specify a predetermined number of minutes of inactivity before terminating a connection. (*Id*., ¶[0751] ("if a user has not actively viewed an HTML page 2602 in a predetermined number of minutes, e.g. ten minutes, the parameter handler 2708 instantiation is instructed to terminate, the session with the hosting server is terminated").

178. Regarding establishing a connection, *Wookey* explains that the "remote machine 30' is 'listening' to the well-known communications port … for a connection request" from **client machine 10**. (Ex. 1005 ¶[0738], *see also id.* ¶[0772].) *Wookey* further explains that "[w]hen a connection request 2510 is received from the client machine 10, the connection manager" of **remote machine 30'** determines "which protocol is being used based on the notification." (*Id.* ¶[0738] (emphasis added); *see also id.* ¶[0739].)

179. *Wookey* further explains that **client machine 10** and **remote machine 30'** "exchange a set of messages which negotiate the parameters under which communications will occur." (Ex. 1005 ¶[0774].) And once negotiations are complete, *Wookey* discloses that **client machine 10** and **remote machine 30'**, via

the network applications on the machines, "are able to communicate as necessary." (*Id.*; *see also id.* ¶¶[0744], [0773].) Thus, in my opinion, **client machine 10** and **remote machine 30'** both send and receive packets when establishing a connection.

180. Moreover, a person of ordinary skill in the art would have understood that when **client machine 10** uses one of the encoded URLs (e.g., the one pointing to **remote machine 30'**) associated with one of the icons on the HTML page 288, it results in the **client machine 10** utilizing the web browser application to establish a connection (e.g., connection 394) with **remote machine 30'** in accordance with a transport protocol.

181. While *Wookey* discloses that various transport protocols (e.g., TCP/IP, IPX/SPX, NetBEUI) may be used to establish the second connection and desired characteristics of such a connection (Ex. 1005, ¶[0225]), *Wookey* does not expressly provide details regarding establishing and maintaining the second connection or the types of negotiable parameters used with the various protocols for the second connection. However, a person of ordinary skill in the art would have had reasons to consider and implement a protocol (which may be "a different type of protocol than the one used to send the request to the remote machine 30" (Ex. 1005 ¶[0732]) that would have met at least some of *Wookey*'s desired characteristics (e.g., such as those identified above) for the second connection

between **client machine 10** and **remote machine 30'** and would have resulted in the code, when used by **client machine 10**, resulting in **client machine 10** operating to identify "idle information for detecting an idle time period…" like that recited in claim element 1.c.

182. Such a skilled person at the time would have recognized that *Eggert* describes such a protocol (which is a TCP-variant protocol using TCP-variant packets) that allows nodes to negotiate or modify timeout timers on a per-connection basis to provide a reliable and efficient network connection. Thus, as I explain further below, based on the teachings of *Eggert* and the knowledge of a person of ordinary skill in the art and guidance from *Wookey*, a person of ordinary skill in the art would have been motivated to configure *Wookey*'s **client machine 10** such that when the user of the **client machine 10** clicks on one of the icons 257, 257' associated with one of the encoded URLs in the HTML page 288, **client machine 10**'s web browser application 280 operates in accordance with a protocol such as the one described by *Eggert* to initiate establishment of the connection between **client machine 10** and **remote machine 30'**uses the browser application 280.

(1) Eggert

183. *Eggert* discloses features relating to establishing a reliable connection between two nodes, where a host may be mobile (e.g., the host experiences

"changes [in] network attachment points based on [its] current location"), and thus discloses networking features in the same technical field as *Wookey*.

184. For example, *Eggert* explains:

> Some hosts are only intermittently connected to the Internet. One example is ***mobile hosts that change network attachment points based on current location***, for example using MobileIP [5] or HIP [6].

(Ex. 1006, 1–2 (emphasis added); *see also* my discussion on *Eggert* above at Section VIII.B and my discussion on the similarities between *Wookey* and *Eggert* below at Section XI.A.1.c.2.) *Wookey* similarly discloses that "the user and/or the client machine 10 may traverse network segments or network access points that cause changes in the network address or host name, e.g., internet protocol (IP) address, of the client machine 10 or causes the client machine 10 to disconnect." (Ex. 1005, ¶[0581] (*Wookey* similarly disclosing that "the user and/or the client machine 10 may traverse network segments or network access points that cause changes in the network address or host name, e.g., internet protocol (IP) address, of the client machine 10 or causes the client machine 10 to disconnect"); *see also id.*, ¶¶[1135]–[1136] (discussing unintentional disconnection between the client and remote machines), [1153] (discussing **client machine 10** losing connection because user enters an elevator).)

185. Therefore, a person of ordinary skill in the art would have had reason to consider *Eggert*'s teachings when contemplating and implementing *Wookey*'s method for establishing a connection between **remote machine 30'** and **client machine 10**.

186. Upon consideration, a person of ordinary skill in the art would have recognized that *Eggert* describes a modification to the TCP protocol to "allow established TCP connections to survive periods of disconnection." (Ex. 1006, Abstract.) In particular, *Eggert* introduces a "TCP Abort Timeout Option" (ATO) that "allows conforming TCP implementations to negotiate individual, per-connection abort timeouts": (*Id.*)

> The TCP Abort Timeout Option allows conforming TCP implementations to negotiate individual, per-connection abort timeouts. Lengthening abort timeouts allows established TCP connections to survive periods of disconnection.

(*Id.*) Thus, as I further explain below, *Eggert* discloses a TCP variant protocol that allows established TCP connections to survive periods of disconnection by negotiating abort timeouts. (*Id.*; *see also* my discussion on *Eggert* above at Section VIII.B.)

187. *Eggert* explains that its TCP-variant protocol solves the problem "where hosts are only intermittently connected to the Internet." (Ex. 1006, 1–3.) For example, a mobile host may "experience disconnected periods during which no

network service is available" when "mobile hosts … change network attachment points." (*Id.* 2–3.) These disconnected periods may lead to the "established TCP connections" being "abort[ed] during periods of disconnection" on the mobile host. (*Id.*) Specifically, *Eggert* states:

> In between connected periods, ***mobile hosts may experience disconnected periods during which no network service is available***. When such hosts use the Transmission Control Protocol (TCP) [1], ***their established TCP connections can abort during periods of disconnection***.

(*Id.*)

188. *Eggert* discloses features that aimed to overcome this problem in the art by "specif[ying] a new TCP option - the Abort Timeout Option [ATO] - that allows conforming hosts to negotiate per-connection abort timeouts." (Ex. 1006, 3) *Eggert*'s TCP-variant protocol "allow[s] mobile hosts to maintain TCP connections across disconnected periods that are longer than their system's default abort timeout":

> This document specifies a new TCP option - ***the Abort Timeout Option - that allows conforming hosts to negotiate per-connection abort timeouts***. This allows mobile hosts to ***maintain TCP connections across disconnected periods that are longer than their system's default abort timeout***.

(*Id.*)

189. *Eggert* further explains that if "[a] TCP implementation" on a device "does not support the TCP [ATO]," then the device "SHOULD silently ignore it":

> ***A TCP implementation that does not support the TCP Abort Timeout Option SHOULD silently ignore it [2].*** This causes the connection to use the default abort timeout, thus ensuring interoperability.

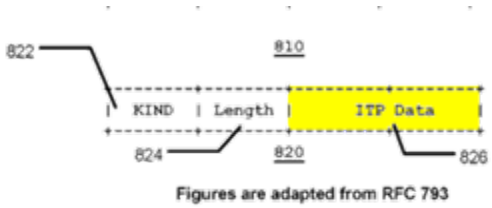
(Ex. 1006, 7.)

190. Thus, a person of ordinary skill in the art would have understood that *Eggert* disclosed and provided a new TCP option to the traditional TCP implementation at the time, where devices configured according to *Eggert* vary from the "traditional TCP implementation" in terms of the ATO negotiation.

191. And as I discuss below, just like the '774 patent, *Eggert*'s ATO is also incorporated in the TCP header and arranged in a similar format. (*Compare* Ex. 1001, 15:59–67, FIG. 8 *with* Ex. 1006, 3–5, FIG. 1.)

192. Thus, a connection using *Eggert*'s ATO would be a TCP-variant connection and at least the segments used for establishing such a TCP-variant connection (e.g., the segments using the TCP ATO) would be TCP-variant packets. (*See* my discussions above at Section X; s*ee also* my discussions above in Section V.A.4(a) (overview of the TCP/IP communication protocol).) As I show below, for example, both formats use a KIND sub-field to indicate the new option, a TCP

option length sub-field, and a data portion (YELLOW) of the TCP option containing the timeout value.



Figures are adapted from RFC 793

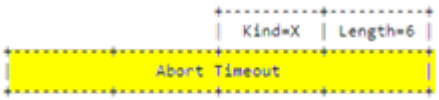


Figure 1: TCP Abort Timeout Option

(Ex. 1001, FIG. 8 (annotated) (left); Ex. 1006, FIG. 1 (annotated) (right).)

193. As a result, a connection using *Eggert*'s ATO would be a TCP-variant connection and at least the segments used for establishing such a TCP-variant connection (e.g., the segments using the TCP ATO) would be TCP-variant packets.

194. *Eggert*'s TCP-variant protocol uses the three-way handshake to establish a connection between two nodes on a network. (Ex. 1006, 5.) A person of ordinary skill in the art would have understood that a three-way handshake was known at the time to involve the exchange of synchronize (SYN) and acknowledge (ACK) messages between the nodes to create a connection—i.e., SYN, SYN-ACK, and ACK message in a standard TCP implementation. In fact, the TCP standard (RFC 793) informed those of ordinary skill at the time about such handshake processes. (*See* Ex. 1008, 31–32, 34–37.) For example, the RFC 793 explained that:

> The synchronization requires each side to send it's 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).)[14] The '774 patent also discloses the same. (*See* Ex. 1001, 14:63–15:10 (repeating the same from RFC 793).) Accordingly, the connection between the nodes is established at the completion of the three-way handshake.

195. As I discuss below (and also exemplified in annotated Figure 2 of *Eggert* below), *Eggert* discloses the features of this limitation in two different ways such that that a first node receives a TCP-variant packet from a second node during the exchange of three messages (3-way handshake) in *Eggert*'s TCP-variant

[14] I refer to Ex. 1008 to demonstrate the knowledge of one of ordinary skill in the art at the time.

protocol for negotiating a per-connection abort timeout, in advance of the TCP-variant connection being established.[15]

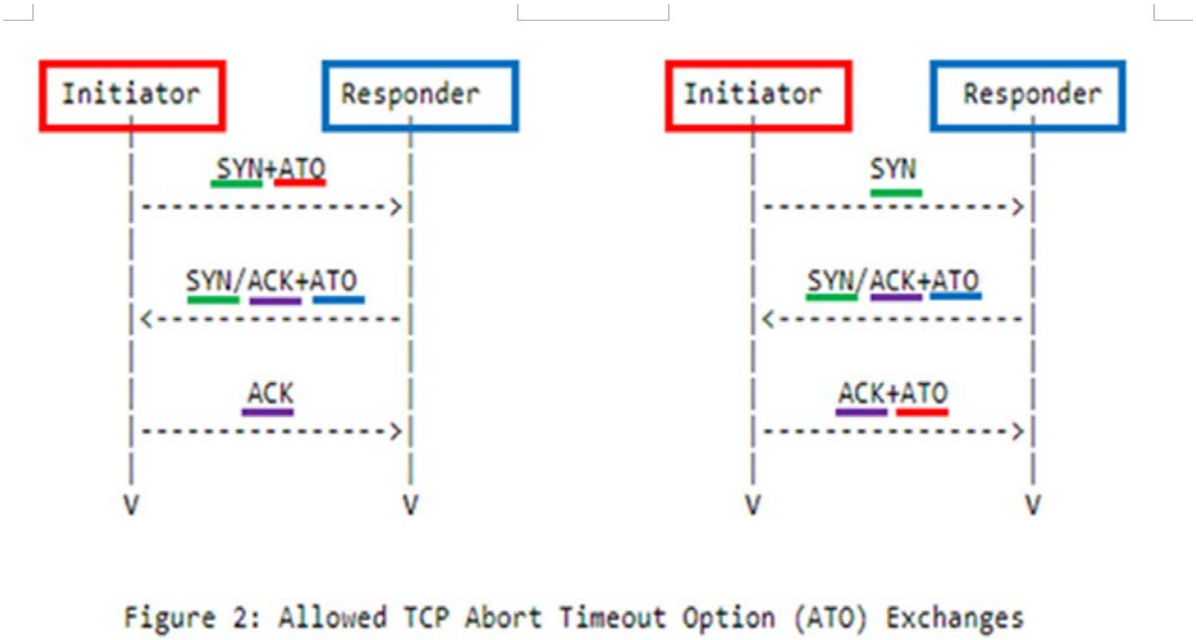


Figure 2: Allowed TCP Abort Timeout Option (ATO) Exchanges

(Ex. 1006, FIG. 2 (annotated).)

196. In the first way, *Eggert* discloses this feature because when an **initiator** initiates an abort timeout negotiation, as shown on the left-hand side of Figure 2 (when the **initiator**), the **initiator** (e.g., client node) sends the **responder**

[15] In the combined *Wookey-Eggert* process/system which I discuss below (*e.g.*, *see* my discussion on the similarities between *Wookey* and *Eggert* below at Section XI.A.1.c.2), it is my opinion that the combined process/system discloses the claim features (e.g., claim elements 1.d–g) under both embodiments shown in *Eggert* (i.e., left and right side of Figure 2).

(e.g., server node) a "**SYN**+**ATO**" segment.[16] (Ex. 1006, FIG. 2; *see also id.*, 5–7.) A person of ordinary skill in the art would have understood that the **ATO** abbreviation in the segment identifies that this segment contains an Abort Timeout Option. (*Id.*, 3.) *Eggert* explains that when the **responder** (e.g., server node) accepts the **initiator's** offered abort timeout, it must echo the offered timeout value in the **ATO** it sends. (Ex. 1006, 5–7.) That is, the **initiator** receives an "**SYN**/**ACK**+**ATO**" segment ("TCP-variant packet") from the **responder**. (*Id.*)

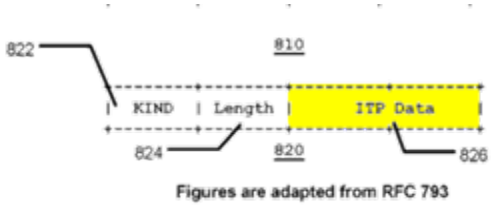
197. Second, as shown on the right-hand side of Figure 2 (when the **responder** initiates an abort timeout negotiation), *Eggert* teaches that the **initiator** (client node) receives an "**SYN**/**ACK**+**ATO**" segment ("TCP-variant packet")

[16] A person of ordinary skill in the art would have recognized and understood that the '774 patent describes a "packet" in a manner consistent with what the TCP standard (RFC 793) describes as "segment" at that time. (*Compare* Ex. 1001, 15:38–58, FIG. 8 *with* Ex. 1008 (RFC 793), 15, FIG. 3 (showing the same TCP header in a TCP "packet" (Ex 1001) and TCP "segment" (Ex. 1008)).) Thus, in my opinion, a person of ordinary skill in the art would have understood that *Eggert*'s TCP "segment" is the same as a TCP "packet" as described and claimed in the '774 patent. .

from a **responder** (server node). (Ex. 1006, 5–7.) As I explained above, the **ATO** identifies that this segment contains an Abort Timeout Option. (*Id.*, 3.)

198. A person of ordinary skill in the art would have understood that the received (**SYN**/**ACK**+**ATO**) segment in both scenarios (left and right side of Figure 2) is a TCP-variant packet because it contains an abort timeout option (ATO), which is an option not standard in the TCP protocol as defined by RFC 793.

199. In fact, as I discussed above, *Eggert*'s TCP ATO shown in Figure 1 uses the same TCP structure as the idle time period (ITP) field illustrated in Figure 8 of the '774 patent. For example, like the '774 patent, *Eggert*'s ATO is also incorporated in the TCP header and arranged in a similar format. (*Compare* Ex. 1001, 15:59–67, FIG. 8 *with* Ex. 1006, 3–5, FIG. 1.) As shown below, both formats use a KIND sub-field to indicate the new option, a TCP option length sub-field, and a data portion (YELLOW) of the TCP option containing the timeout value. (*Compare* Ex. 1001, FIG. 8 *with* Ex. 1006, FIG. 1.)



Figures are adapted from RFC 793

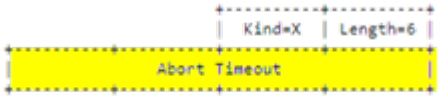


Figure 1: TCP Abort Timeout Option

(Ex. 1001, FIG. 8 (annotated) (left); Ex. 1006, FIG. 1 (annotated) (right).)

200. Thus, it is my opinion, *Eggert* uses the same format for the ATO as the embodiment disclosed in the '774 patent for a TCP-variant packet. Accordingly, as I explained above, a person of ordinary skill in the art would have understood that connection using *Eggert*'s ATO would be a TCP-variant connection and at least the segments used for establishing such a TCP-variant connection (e.g., the segments using the TCP ATO) would be TCP-variant packets. (*See* my discussions above at Section X.)

201. Therefore, *Eggert* discloses the features of "receiving, by a second node from a first node, a *transmission control protocol (TCP)-variant* packet in advance of a *TCP-variant* connection being established," as recited in this claim element. And as I explain below, a person of ordinary skill in the art would have been motivated to include such features into *Wookey*'s process/system. (*See* my discussion below in Section XI.A.1.c.(2).)

(2) Reasons to Combine

202. Based on *Eggert*'s disclosures and the knowledge of a person of ordinary skill at the time in context of the disclosures and guidance of *Wookey*, it is my opinion that a person of ordinary skill in the art at the time would have been motivated to configure and implement *Wookey*'s **client machine 10** to use a TCP-variant protocol (with a TCP-variant packet) like the one disclosed by *Eggert* when establishing (and using) the second communication between **client machine 10**

and **remote machine 30'** in relation to claim elements 1.c–1.f. A person of ordinary skill in the art would have recognized that such an implementation would have had many advantages and would have resulted in a predictable combined system/process by combining well-known technologies using known methods at the time.

203. A person of ordinary skill in the art would have recognized that *Eggert* and *Wookey* disclose features in a similar technological field. (*See* my discussions above in Section V.A (overview of computer networks and communication protocols).) For example, both *Wookey* and *Eggert* are concerned with nodes such as mobile nodes maintaining connections in situations where at least one node in the connection may lose connectivity due to its movement, such as where a node is transitioning from one network access point to another access point.

204. *Wookey* discloses that "the user and/or the client machine 10 may traverse network segments or network access points that cause changes in the network address or host name, e.g., internet protocol (IP) address, of the client machine 10 or causes the client machine 10 to disconnect." (Ex. 1005 ¶¶[0581]; *see also id.*, ¶¶[1135]-[1136] (discussing unintentional disconnection between the client and remote machines), [1153] (discussing **client machine 10** losing connection because user enters an elevator).)

205. Similarly, *Eggert* discloses that "[l]engthening abort timeouts allows established TCP connections to survive periods of disconnection" as "[s]ome hosts are only intermittently connected to the Internet." (Ex. 1006, 1–2.) Similar to *Wookey*, *Eggert* discloses that "[o]ne example [of such unintentional disconnections] is mobile hosts that change network attachment points based on current location" and "[i]n between connected periods, mobile hosts may experience disconnected periods during which no network service is available." (*Id.*, 2-3.)

206. Thus, in my opinion, because both *Wookey* and *Eggert* are directed to establishing a reliable connection between two nodes, where the two nodes may be mobile nodes that experience an unintentional disconnection, a person of ordinary skill in the art would have had reason to consider *Eggert* when contemplating and implementing the teachings of *Wookey*. Consequently, as I explain further below, in context of the knowledge of a person of ordinary skill in the art at the time and in light of *Wookey*'s disclosures and guidance, upon considering the disclosures and suggestions in *Eggert*, one of ordinary skill in the art would have been motivated to modify *Wookey*'s system/process in the manner that I discuss in this Declaration.

207. Moreover, as I discuss above, *Wookey* describes providing access to icons 257, 257' on an HTML page 288 that, when used by **client machine 10**,

causes **client machine 10** to establish a connection between **client machine 10** and **remote machine 30'** using a protocol that can be different from the one used to connect **client machine 10** and **remote machine 30**.

208. I also explained above how *Wookey* further discloses the types of characteristics a person of ordinary skill in the art would have considered concerning the protocol that can be used for the second connection, such as negotiating parameters related to the connection, reducing unintentional termination of sessions due to an imperfect connection, detecting and handling disconnections like when a mobile device enters an elevator, and ability to specify inactive time prior to connection termination. (Ex. 1005 ¶¶[0581], [0721], [0751], [1134]–[1136], [1153]; *see* my discussion above at Section XI.A.1.c.) Therefore, a person of ordinary skill in the art would have been motivated by *Wookey*'s disclosures and suggestions to consider protocols that would provide the desired features to *Wookey* for the second connection with **remote machine 30'**, including the features associated with the TCP-variant connection described by *Eggert*. Such a person of ordinary skill in the art would have been motivated to incorporate a protocol like *Eggert*'s TCP-variant protocol for use by **client machine 10** for establishing a connection with **remote machine 30'**.

209. A person of ordinary skill in the art would have recognized that by using the TCP variant protocol with an ATO option as described in *Eggert* would

have provided **client machine 10** with benefits including an improved, reliable connection to **remote machine 30'** with the versatility of negotiated timeout parameters consistent with *Wookey*'s desired characteristics for the second connection. For example, a person of ordinary skill in the art would have recognized that by using a TCP variant protocol with an ATO option (or similar feature as described by *Eggert*), *Wookey*'s method would enable "negotiat[ion] [of] individual, per-connection abort timeouts … to survive periods of disconnection" over a reliable connection. (Ex. 1006, 1.) A person of ordinary skill in the art would have thus understood that *Wookey*'s method for "providing a client with a reliable connection to a host service" (Ex. 1005 ¶[0077]) would have been improved by allowing "[l]ong abort timeout values" so that the "hosts … tolerate extended periods of [temporary] disconnection," as provided by *Eggert* (Ex. 1006, 5; *see also* Ex. 1005 ¶[0903].)

210. Moreover, *Wookey* describes using various reliable communication protocols to establish a second connection. (Ex. 1005 ¶[0216].) As such, a person of ordinary skill in the art would have recognized that configuring *Wookey* to utilize a TCP-variant protocol based on the teachings of *Eggert* would have been both a predictable and straightforward implementation. A person of ordinary skill in the art could have achieved this implementation by combining well-known

technologies using known methods, such as known network design concepts and technologies as described by *Wookey* and *Eggert* and known in the art at the time.

211. Additionally, *Eggert*'s protocol aims to avoid or mitigate interruptions caused by intermittent disconnections between the nodes, a problem recognized by *Wookey*. (Ex. 1006, 1–3; Ex. 1005 ¶[0903].) Thus, a person of ordinary skill in the art would have been motivated to consider *Eggert*'s features when considering how to implement *Wookey*'s process/system and modify *Wookey* as discussed. For example, a person of ordinary skill in the art would have understood that *Eggert*'s TCP variant protocol, which allows the nodes to negotiate an abort timeout, would have improved *Wookey*'s process just as described in *Eggert*. Accordingly, a person of ordinary skill in the art would have understood that such a combination as I explained above would have resulted in a predictable modified *Wookey* process/system that would have provided a network communication exchange that minimizes disruptions caused by timeout issues.

212. A person of ordinary skill in the art would have further recognized that the *Wookey-Eggert* combination would have involved the use of known technologies (e.g., aspects of similar protocols) and design concepts and processes to obtain the foreseeable result of a reliable connection between **client machine 10** and **remote machine 30'**. For example, a person of ordinary skill in the art would have appreciated that the above-modification would have involved the substitution

of features from one reliable protocol amongst a finite number of available alternative reliable communication protocols, such as that described by *Eggert*. Given such disclosures and guidance by *Wookey* and *Eggert*, coupled with the skills and knowledge of an ordinarily skilled person at the time, it is my opinion that a person of ordinary skill in the art would have reasonably expected the above-discussed combination to successfully operate as intended to ensure communication disruptions were handled in a manner consistent with the way *Eggert* and *Wookey* contemplated. Thus, in my opinion, a person of ordinary skill in the art would have had a reasonable expectation of success in the above-modification.

213. Given such disclosures and guidance by *Eggert* and *Wookey*, a person of ordinary skill in the art would thus have found configuring *Wookey*'s system/process such that the encoded URLs in HTML page 288 or page 288 including the encoded URLs (associated with the icons 257, 257'), when used by **client machine 10**, would cause **client machine 10** to utilize a TCP-variant protocol like *Eggert*'s TCP-variant protocol a foreseeable and straightforward implementation. In my opinion, a person of ordinary skill in the art would have been motivated to configure *Wookey*'s encoded URL such that it includes information causing **client machine 10** to properly establish a connection with **remote machine 30'** using the TCP-variant protocol (like that described

by *Eggert*), based on what I have discussed above including the type of characteristics *Wookey* desired for the second connection. As I have explained, *Wookey* discloses that "***the URL includes a file, or a reference to a file, that contains the information necessary for the client to create a connection to the remote machine hosting the resource***." (Ex. 1005 ¶[0216].) Accordingly, as one non-limiting example, such a skilled person would have been motivated to configure the file "containing the information necessary for the client to create a connection" (Ex. 1005 ¶[0216]) such that it included information causing the client to establish a connection using the TCP-variant protocol, like that disclosed by *Eggert*. In addition, the encoded URL can also include "the location of the resource (i.e., on which remote machines it is hosted or, alternatively, the address of a master remote machine, a gateway, or other remote machine 30); [and] a launch command associated with the resource." Thus, in the combined *Wookey-Eggert* system/process, use of the encoded URL by **client machine 10** (e.g., as a result of its selection (e.g., clicking on it)) would cause **client machine 10** to initiate a process of establishing the second connection with **remote machine 30'** using the TCP-variant protocol. In my opinion, a person of ordinary skill in the art would have had the capabilities and knowledge to successfully achieve this implementation by combining well-known technologies using known methods,

such as known network design concepts and technologies as described by *Wookey* and *Eggert* and also known in the art at the time.

214. In fact, as I explained, a person of ordinary skill in the art would have been skilled and would have had the knowledge to configure *Wookey*'s system and method to implement a TCP-variant packet and protocol in various ways, while taking into account any known programming, design, and other related concepts, limitations, benefits, and the like to ensure the resulting combination operated properly and as intended.

215. For example, a person of ordinary skill in the art would have been motivated based on such disclosures to configure *Wookey*'s system and process consistent with the exemplary annotated figures from *Wookey* and *Eggert* that I provide below. The configurations below show one example of how a person of ordinary skill in the art could and would configure *Wookey*'s system in view of the teachings of *Eggert*.[17]

[17] The configurations that I discuss below are exemplary and not limiting.

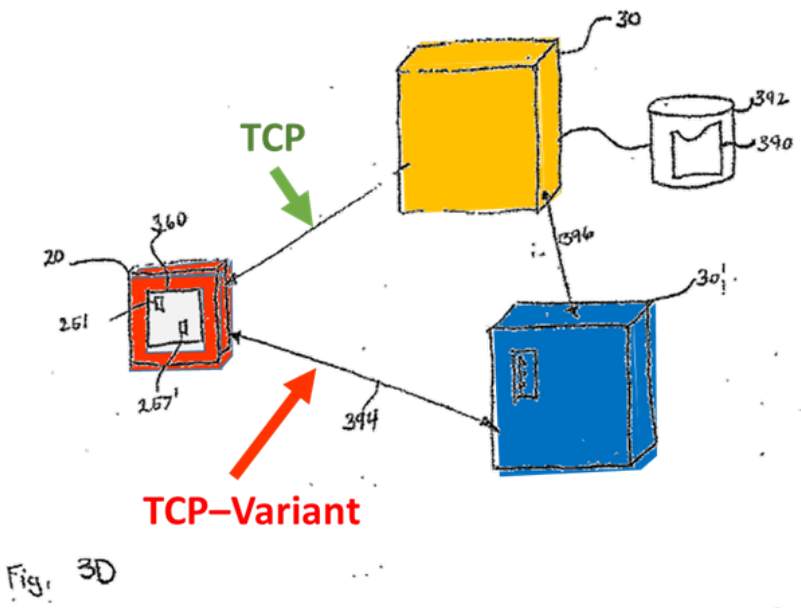


Fig. 3D

(Ex. 1005, FIG. 3D (annotated).)

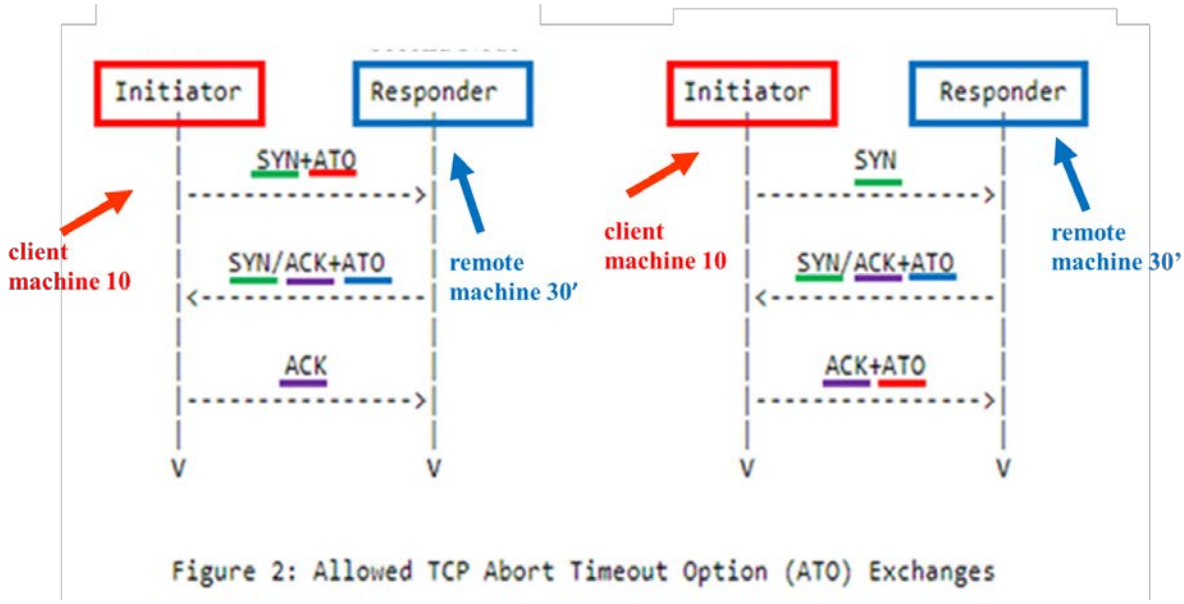


Figure 2: Allowed TCP Abort Timeout Option (ATO) Exchanges

(Ex. 1006, FIG. 2 (annotated).)

216. Accordingly, the *Wookey-Eggert* combination as I have discussed above discloses and/or suggests that when during the process of establishing a connection between **client machine 10** and **remote machine 30'**, **client machine 10** ("second node") would have performed a three-way handshake with **remote machine 30'** ("first node"), which would have included **client machine 10** receiving, from **remote machine 30'**, a TCP-variant packet in advance of a TCP-variant connection being established (which would be established at the completion of the three-way handshake). (*See* my discussion above at Section XI.A.1.c.2 (discussing the *Wookey-Eggert* combination).) Therefore, the *Wookey-Eggert* combination discloses and/or suggests limitation 1.c.

d) **detecting a time period parameter field in the TCP-variant packet;**

217. In my opinion, the combination of *Wookey* and *Eggert* discloses and/or suggests the features of this claim element.

218. As I have discussed above for claim limitation 1.c, the combined *Wookey-Eggert* system/process would have involved negotiating the abort timeout option included in the received "**SYN/ACK+ATO**" segment ("TCP-variant packet") at **client machine 10** in advance of establishing a TCP-variant connection with **remote machine 30'**. (*See* my discussion above at Section XI.A.1.c.)

219. *Eggert* explains that this negotiation includes the **initiator** detecting the **ATO** field ("time period parameter field") incorporated in the received "**SYN/ACK+ATO**" segment ("TCP-variant packet") from **remote machine 30'**. (Ex. 1006, 3–5.) As shown in Figure 1 below the **ATO** field comprises three fields: the Kind, Length, and Abort Timeout Option fields. (Ex. 1006, 3–5, FIG. 1.)

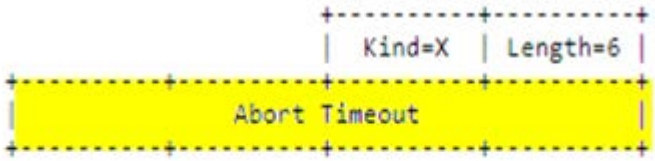


Figure 1: TCP Abort Timeout Option

(Ex. 1006, FIG. 1 (annotated).)

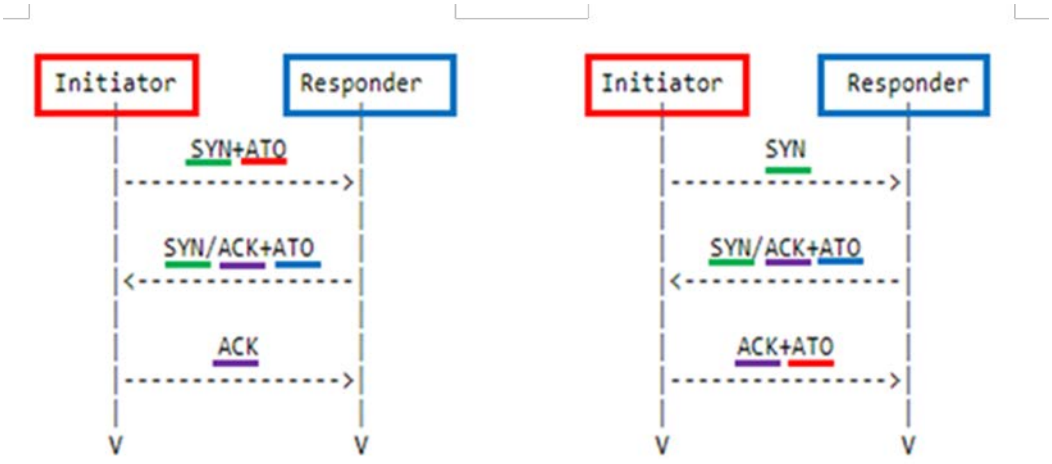


Figure 2: Allowed TCP Abort Timeout Option (ATO) Exchanges

(Ex. 1006, FIG. 2 (annotated).)

220. A person of ordinary skill in the art would have understood that *Eggert* discloses at least two ways that the **initiator** detects the ATO field ("time period parameter field") in the "**SYN**/**ACK**+**ATO**" segment ("TCP-variant packet").

221. First, regarding the left-hand side of Figure 2, *Eggert* provides that the **initiator** must accept the **responder**'s returned abort timeout value in the received "**SYN**/**ACK**+**ATO**" segment because the **initiator** proposed the abort time in the initial "**SYN**+**ATO**" segment it sent to the **responder**:

> ***The host that initially proposed the Abort Timeout Option analyzes the next segment it receives from its peer***. If the next reply segment does not contain an Abort Timeout Option, the connection MUST use the default abort timeout. ***If it does, the connection MUST use the abort timeout contained inside the***

> ***Abort Timeout Option.*** This can be a different abort timeout than initially proposed, if the peer decided to shorten it.

(Ex. 1006, 5–7 (excerpted with emphasis added).) Also *Eggert* explains that "[t]he host that initially proposed the Abort Timeout Option analyzes the next segment it receives from its peer" and if the next segment (e.g., the **SYN/ACK+ATO**" segment) contains the ATO, "the connection MUST use the abort timeout contained inside the Abort Timeout Option." (*Id.*, 7.)

222. Second, regarding the right-hand side of Figure 2, the **initiator** may either accept or shorten the offered abort timeout from the **responder**. (*Id.*, 5–7)

223. Thus, a person of ordinary skill in the art would have understood that for either side of Figure 2, the **initiator** (e.g., client node) must first detect the ATO parameter field containing the ATO value before it can accept or shorten the value. (*Id.*, 5.) In this way, it is my opinion that *Eggert* discloses detecting an ATO field in the TCP-variant packet.

224. The ATO field represents a time period parameter field used to convey an abort timeout value between two hosts. (Ex. 1006, 1–7, FIG. 1.) The ATO field contains the abort timeout value, which defines a "time period," as that claimed. (Ex. 1006, 2–5, FIG. 1.) For example, *Eggert* explains that the abort timeout value in the ATO field "is the desired abort timeout of the connection, specified in seconds":

> Figure 1 shows the format of the TCP Abort Timeout Option. In Figure 1, "X" is a TCP option number to be assigned by IANA upon publication of this document (see Section 5.) ***"Abort Timeout" is the desired abort timeout of the connection, specified in seconds.***

(Ex. 1006, 3 (excerpted with emphasis added).)

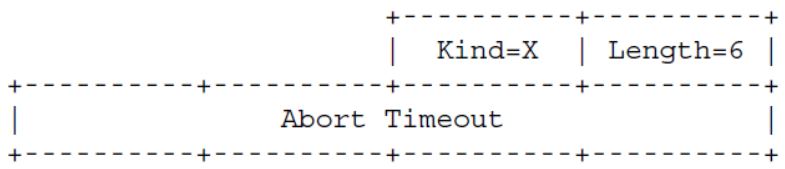


Figure 1: TCP Abort Timeout Option

(*Id.*, FIG. 1; *see also id.*, Abstract.)

225. *Eggert* further explains that by "[l]engthening [the] abort timeout" value that the "established TCP connections" can "survive periods of disconnection." (*Id.*, Abstract ("Lengthening abort timeouts allows established TCP connections to survive periods of disconnection.").)

226. A person of ordinary skill in the art would have understood that the abort timeout value defines an idle time period because it is associated with a period where no packet is communicated in the connection to keep the connection active. (*See, e.g.*, *id.*, Abstract; *see* my discussion below in Section XI.A.1.e.)

227. Such features would have been implemented in the combined *Wookey-Eggert* process for the same reasons I explain above (*see, e.g.*, my

discussions above for claim element 1.c in Section XI.A.1.c regarding implementing the discussed features from *Eggert* with *Wookey*'s process). Accordingly, for similar reasons, and based on the disclosures I discuss above, a person of ordinary skill in the art would have understood that in the combined *Wookey-Eggert* process, which would have implemented the process of establishing the connection between the **client machine 10** and **remote machine 30'** using the TCP-variant protocol, to include the process of detecting, by **client machine 10**, an ATO field ("time period parameter field") in the "**SYN/ACK+ATO**" segment ("TCP-variant packet") received from **remote machine 30'**. (*See* my discussion above at Section XI.A.1.c.2 (discussing the *Wookey-Eggert* combination).)

228. Thus, in my opinion, the *Wookey-Eggert* combination discloses and/or suggests claim element 1.d.

e) **identifying metadata in the time period parameter field for a time period, during which no packet is received from the first node in a data stream of the TCP-variant connection and processed by the second node to keep the TCP-variant connection active; and**

229. In my opinion, the *Wookey-Eggert* combination I discuss above discloses and/or suggests the features recited in claim element 1.e.

230. As I discussed above for claim element 1.d, the *Wookey-Eggert* combination involves the **client machine 10** detecting an ATO field ("time period

parameter field") in the received **SYN/ACK+ATO**" segment. (*See* my discussion above in Section XI.A.1.d.) As I discuss below, the *Wookey-Eggert* combination further discloses identifying an abort timeout value ("metadata") in the abort timeout field ("time period parameter field") for an abort timeout ("time period"), during which no packet is received from **remote machine 30'** ("first node") in a data stream of the TCP-variant connection and processed by **client machine 10** ("second node") to keep the TCP-variant connection active. (*See* my discussion above at Section XI.A.1.d.)

231. For example, *Eggert* discloses that the **initiator** (client node) "identif[ies] metadata in the time period parameter field for an time period," as claimed. Like as I discussed above for claim limitation 1.d, *Eggert* explains that the **initiator** accepts or shortens the offered abort timeout value ("metadata") specifying an abort timeout ("time period") contained within the "**SYN/ACK+ATO**" segment ("TCP-variant packet"). (*See* my discussions above at Section XI.A.1.d; Ex. 1006, 5–7.) Thus, the **initiator** must first identify the abort timeout value before it can either accept or shorten the offered abort timeout. (Ex. 1006, 5–7.) Consequently, *Eggert* discloses that the **initiator** processes the received "**SYN/ACK+ATO**" segment from the **responder** according to an established format (Ex. 1006, FIG. 1) to identify which bits of the option represents the proposed abort timeout duration value. (*Id.*, 3–5.)

232. *Eggert* further explains that the "Abort Timeout" is "specified in seconds" (Ex. 1006, 3) and is represented by a "32-bit value" (*id*., 9), and thus it can "express abort timeouts from zero seconds to over 136 years":

> The TCP specification [1] does not define a range for permitted abort timeouts. Similarly, this document does not restrict the range of timeout values used with the TCP Abort Timeout Option. ***The 32-bit value in the option can express abort timeouts from zero seconds to over 136 years.***

(*Id*., 9 (excerpted with emphasis added.) The bits that identify the ATO field as associated with an abort timeout (KIND, LENGTH) and the bits representing the abort timeout value each constitute "metadata" for the proposed or responsive abort timeout ("time period"). Further, as I have discussed above for claim limitation 1.c, the format for representing the TCP abort timeout in Figure 1 of *Eggert* is similar to the ITP data field 826 in Figure 8 of the '774 patent. (*See* my discussions above at Section XI.A.1.c (comparing the '774 patent packet format in Figure 8 with *Eggert*'s similar packet format in Figure 1).)

233. The manner as to how *Eggert* discloses identifying such values is consistent with the '774 patent's discussions. For example, the '774 patent explains that the metadata can be a "duration of time." (Ex. 1001, 16:8–16 ("an ITP data field in a packet may include and/or otherwise identify one or more of a duration of time for detecting an idle time period"), 22:5–10 ("The first idle time

period header may identify metadata including and/or identifying for detection of the first idle time period by first node 601 a duration of time, a generator for determining a duration of time, and/or an input for determining a duration of time."). A duration of time is what the ATO value in *Eggert* contains. (Ex. 1006, 3–5, FIG. 1.)

234. In my opinion, *Eggert* further discloses that the abort timeout value corresponds to a "time period, during which no packet is received from the first node in a data stream of the TCP-variant connection … to keep the TCP-variant connection active," as recited in claim element 1.e.

235. For example, *Eggert* explains that the nodes use the abort timeout value to track periods during which no packet is received in a data stream of the TCP-variant connection to keep the connection active. (Ex. 1006, 1–7; *see also id.*, 9 ("Long abort timeout values allow hosts to tolerate extended periods of disconnection").) A person of ordinary skill in the art would have recognized that the packet that must be "received … to keep the … connection active" in *Eggert* is an ACK that a node receives from the other node in response to a previous sent segment (where the ACK is received in the data stream through which the segment was sent). (*See* also my discussion above regarding user timeout in Section V.A.4(a)(c).) To be sure, *Eggert* explains that its ATO relies upon:

> The TCP specification [1] [which] includes a 'user timeout' that defines the maximum amount of time that segments may remain ***unacknowledged*** before TCP will abort the connection. If a disconnection lasts longer than the user timeout, the TCP connection will abort.

(Ex. 1006, 3 (excerpted with emphasis added).)

236. My opinions above are further supported because the '774 patent, like *Eggert*, also describes modifying a TCP user timeout:

> 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.

(Ex. 1001, 13:62–65.)

> In an aspect, ITP option handler component 562 may one or more attribute option handler components 564 to modify one or more corresponding attributes of a keep-alive option, a TCP user timeout, a retransmission timeout, an acknowledgment timeout, and another timeout associated with the TCP connection, in response to identifying the ITP header. The modifying may be based on the content of the ITP header.

(*Id.*, 22:28–36.)

237. Furthermore, a person of ordinary skill in the art would have understood that the TCP-variant connection as described in *Eggert* corresponds to

a “data stream” as recited in claim element 1.e. For example, as I discussed above, *Eggert*’s implementation is based on the TCP specification set forth in RFC 793. RFC 793 provides that a connection corresponds to a data stream:

> Connections:
>
> **The reliability and flow control mechanisms described above require that TCPs initialize and maintain certain status information for each data stream. The combination of this information, including sockets, sequence numbers, and window sizes, is called a connection**. Each connection is uniquely specified by a pair of sockets identifying its two sides.
>
> When two processes wish to communicate, their TCP’s must first establish a connection (initialize the status information on each side). When their communication is complete, the connection is terminated or closed to free the resources for other uses.
>
> Since connections must be established between unreliable hosts and over the unreliable internet communication system, a handshake mechanism with clock-based sequence numbers is used to avoid erroneous initialization of connections.

(Ex. 1008, 5 (excerpted and emphasis added).)

> To identify the separate data streams that a TCP may handle, the TCP provides a port identifier. Since port identifiers are selected independently by each TCP they might not be unique. To provide for unique addresses within each TCP, we

> concatenate an internet address identifying the TCP with a port identifier to create a socket which will be unique throughout all networks connected together.
>
> A connection is fully specified by the pair of sockets at the ends. A local socket may participate in many connections to different foreign sockets. A connection can be used to carry data in both directions, that is, it is "full duplex".

(Ex. 1008, 10 (excerpted).)

238. Moreover, it is my opinion that a person of ordinary skill in the art would have understood that *Eggert* teaches that the abort timeout value corresponds to a "time period, during which no packet is received from the first node in a data stream of the TCP-variant connection *and processed by the second node* to keep the TCP-variant connection active," as claimed.

239. For example, as I discussed above, *Eggert*'s implementation is based on the TCP specification set forth in RFC 793 (Ex. 1008), with the added TCP Abort Timeout Option (ATO) which "allows conforming TCP implementations to negotiate individual, per-connection abort timeouts." (Ex. 1006, 1; *see also id.*, 3 (referencing RFC 793, which is reference "[1]," when discussing communication between two nodes), 5 (describing that in its operation, during certain other states, normal timeouts are used in accordance with RFC 793), 9 (discussing the length of abort timeouts as specified in RFC 793), 15 (identifying reference [1] as RFC

793).). Thus, a person of ordinary skill in the art would have understood that in *Eggert*, a received packet such as an ACK packet would have been processed in accordance with the disclosures of RFC 793.

240. In light of this understanding, a person of ordinary skill in the art would have been motivated that in *Eggert*, if an ACK packet is received during the user timeout, the receiving node would have performed processing on the received ACK packet to link the acknowledgment in the ACK packet with a previously sent data packet to keep the TCP-variant connection active. For example, a person of ordinary skill in the art would have understood that the sequence number in the received ACK packet would be compared with the sequence number in the sent data segment to ensure the acknowledgment is related to the previously sent packet, to keep the TCP-variant connection active:

> The typical kinds of sequence number comparisons which the TCP must perform include:
>
> (a) Determining that an acknowledgment refers to some sequence number sent but not yet acknowledged.
>
> (b) Determining that all sequence numbers occupied by a segment have been acknowledged (e.g., to remove the segment from a retransmission queue).

> (c) Determining that an incoming segment contains sequence numbers which are expected (i.e., that the segment "overlaps" the receive window).

(Ex. 1008, 28 (section 3.3) (excerpted).) Thus, *Eggert* suggests that a packet needs to be received and processed by the client node to keep the TCP-variant connection active.

241. Accordingly, in my opinion, a person of ordinary skill in the art would have understood that *Eggert*'s description of the abort timeout as the duration of time during which a node can wait to receive and process an ACK is a period "during which no packet is received from the first node in a data stream of the TCP-variant connection and processed by the second node to keep the TCP-variant connection active."

Disconnection

242. It is my opinion that the combined *Wookey-Eggert* process/system discloses or suggests that the abort timeout value in the ATO field corresponds to a "time period, during which no packet is received from the first node in a data stream of the TCP-variant connection and processed by the second node to keep the TCP-variant connection active," as claimed, in an **additional** way where a disconnection (e.g., which may be linked to the physical disconnection of a network medium or simply a dead connection) occurs between **client machine 10**

and **remote machine 30'**.[18] (Ex. 1005 ¶¶[0581], [1135]-[1136], [1153]; Ex. 1006, 1–3; *see* my discussions above at Section XI.A.1.c.2 (noting that both *Wookey* and *Eggert* aim to maintain connection between nodes through unintentional disconnection periods).)

243. For example, when an interruption in communication occurs such as a disconnection in the network service (as contemplated by *Wookey* and *Eggert*) or in a physical network medium (as described in the '774 patent) (Ex. 1001, 2:16–19, 2:33–36, 13:5–14), packets from one node do not reach the other node in either direction. In this situation (as contemplated by the '774 patent), no packets are received from the **responder** in the data stream of the TCP-variant connection and processed by the **initiator** to keep the connection active because no packet can get through the interruption or the physical medium's disconnection. (Ex. 1001, 13:5–14, 2:33–36, FIG. 7 (annotated below in yellow showing no communication between the nodes during a disconnection).)

[18] A person of ordinary skill in the art would have understood that this disconnection scenario is consistent with how the '774 patent describes such features. (Ex. 1001, 13:5–14, 2:33–36 (describing that the idle time period may be linked to the physical disconnection of a network medium or simply a dead connection).)

244. Certainly, during such a disconnection period, neither node would have ever received any packets from the other node during such a disconnection period. Therefore, neither node would have attempted to send a responsive acknowledgment packet during the period. This is the type of situation where, as discussed above, *Eggert* aims to maintain the connection, e.g., maintain the connection even when no packet is received by the **initiator** from the **responder** in the data stream of the TCP-variant connection during the negotiated abort timeout period. And in this disconnection scenario, since no packet would have been received by the **initiator**, no packet would have been processed by the **initiator** (e.g., there would be no received ACK packet to process).

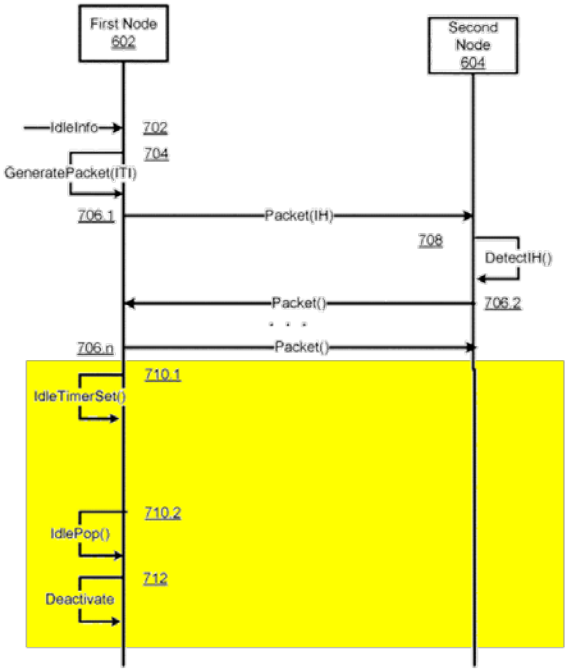


Fig. 7

(Ex. 1001, FIG. 7 (annotated).)

245. Accordingly, for reasons similar to those I have explained above, the *Wookey-Eggert* combined process would have been further configured to include the process of **client machine 10** identifying an abort timeout value ("metadata") in the abort timeout field ("time period parameter field") for a time period, during which no packet is received from **remote machine 30'** ("first node") in a data stream of the TCP-variant connection and processed by **client machine 10** ("second node") to keep the TCP-variant connection active. (*See* my discussions above at Section XI.A.1.c.2.)

246. Thus, for the reasons discussed above, it is my opinion that the *Wookey-Eggert* combination discloses and/or suggests limitation 1.e.

f) **calculating, by the second node and based on the metadata, a timeout attribute associated with the TCP-variant connection for being used to at least partially close the TCP-variant connection.**

247. In my opinion, the *Wookey-Eggert* combination discloses and/or suggests the features of claim 1.f.

248. As I discussed above for claim elements 1.c–d, the *Wookey-Eggert* combination involves establishing a TCP-variant connection between the **client machine 10** and the **remote machine 30'** using the TCP-variant protocol like or as taught in *Eggert*. (*See* my discussion above in Section IX.A.1.c–d.) The analysis above for claim limitation 1.d shows how the *Wookey-Eggert* combination involves identifying an abort timeout value ("metadata") for an abort timeout ("time period"). (*See* my discussion above in Section IX.A.1.e.) As I further discuss below, the *Wookey-Eggert* combination discloses and/or suggests calculating, by **client machine 10** ("second node") and based on the abort timeout value ("metadata"), a timeout attribute associated with the TCP-variant connection in at least two ways. Moreover, a person of ordinary skill in the art would have understood that this timeout attribute is used to at least partially close the TCP-

variant connection in the *Wookey-Eggert* combination. (*See* my discussions above at Section XI.A.1.c–e.)

249. For example, *Eggert* discloses that while "[m]any TCP implementations default to user timeout values of a few minutes" (Ex. 1006, 3), after successful negotiation, hosts use the abort timeout for the connection:

> If the next reply segment does not contain an Abort Timeout Option, the connection MUST use the default abort timeout. If it does, the connection MUST use the abort timeout contained inside the Abort Timeout Option.

(Ex. 1006, 7 (excerpted).) Figure 2 reveals the two scenarios for ATO negotiation, both of which disclose the features recited in claim limitation 1.f.

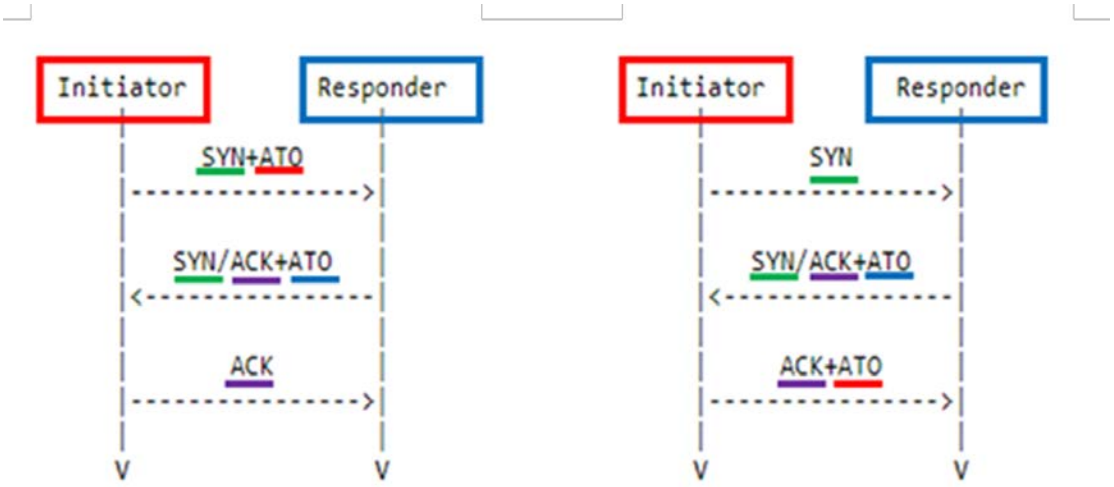


Figure 2: Allowed TCP Abort Timeout Option (ATO) Exchanges

(*Id.*, FIG. 2 (annotated).)

250. First, *Eggert* explains that the **initiator** in the left-hand side scenario of Figure 2 has only a single option because it proposed the abort timeout value. (Ex. 1006, 5–7.) Namely, the **initiator** "MUST be prepared to accept a shorter timeout value [from the **responder**] than proposed after the negotiation." (*Id.*, 5.) A person of ordinary skill in the art would have understood that *Eggert*'s disclosure of the **initiator** setting the user timeout value for the TCP-variant connection as the abort timeout value chosen by the **responder** describes a "calculate[ing]" process like that recited in limitation 1.g.

251. Second, as shown on the right-hand side of Figure 2, the **initiator**, because the **responder** specified the initial ATO in the "**SYN/ACK+ATO**" segment, may decide "whether to accept, shorten, or reject its peer's proposed abort timeout." (Ex. 1006, 5–7; *see also id.*, 9.) A person of ordinary skill in the art would have understood that the **initiator** calculates the user timeout value for the TCP-variant connection by setting it based on accepting, shortening, or rejecting the value ("metadata") in the ATO field in the "**SYN/ACK+ATO**" segment.

252. In my opinion, the '774 patent describes timeout attributes broadly and is consistent with *Eggert*'s abort timeout value:

> ITP option handler component 562 may [*sic*] one or more attribute option handler components 564 to modify one or more

> corresponding attributes of a keep-alive option, a TCP user timeout, a retransmission timeout, an acknowledgment timeout, and another timeout associated with the TCP connection, in response to identifying the ITP header.

(Ex. 1001, 22:28–35.)

253. In addition, in the *Wookey-Eggert* combination, the user timeout value ("timeout attribute") set for the TCP-variant connection is used to at least partially close the TCP-variant connection. For example, as I have discussed above, the user timeout in *Eggert* is used to abort ("at least partially close") the TCP-variant connection. (*See* my discussions above at Sections XI.A.1.c.1, XI.A.1.e.) In particular, a person of ordinary skill in the art would have understood that the abort timeout value that is negotiated in *Eggert* via the ATO is used to set the user timeout value for the TCP-variant connection established between the two nodes. (Ex. 1006, 1 ("The TCP Abort Timeout Option allows conforming TCP implementations to negotiate individual, per-connection abort timeouts.").)

254. Accordingly, in the scenario when there is no ACK packet received by the **initiator**, with respect to a previously sent segment, within the duration of the set user timeout, the TCP-variant connection at the **initiator** will abort ("at least partially close"). (Ex. 1006, 3 ("The TCP specification [1] includes a 'user timeout' that defines the maximum amount of time that segments may remain unacknowledged before TCP will abort the connection. *If a disconnection lasts*

longer than the user timeout, the TCP connection will abort."), 5 ("The timeout value included in the option specifies *the proposed abort timeout for the connection*.").) Therefore, the user timeout value ("timeout attribute") set for the TCP-variant connection is used to abort ("at least partially close") the TCP-variant connection. That is, a person of ordinary skill in the art would have understood, for example, that the **initiator**'s timer would be configured based on the negotiated abort timeout value, which as I have explained above is used to set the user timeout value for the connection, and in a scenario where the **initiator** detects the expiration of this timer, the TCP-variant connection is aborted. (*See, e.g.*, Ex. 1006, 3, 5.)

255. Such features would have been implemented in the combined *Wookey-Eggert* process for the same reasons I explain above (*see, e.g.*, my discussions above for claim elements 1.c–1.e in Sections XI.A.1.c–e regarding implementing the discussed features from *Eggert* with *Wookey*'s process). Accordingly, for similar reasons, and based on the disclosures I discuss above, a person of ordinary skill in the art would have understood that in the combined *Wookey-Eggert* system/process, which would have implemented the process of establishing the connection between the **client machine 10** and **remote machine 30'** using the TCP-variant protocol, the would have been further configured to include the process of calculating, by **client machine 10** and based on the abort

timeout value ("metadata"), a user timeout value ("timeout attribute") associated with the TCP-variant connection for being used to abort ("at least partially close") the TCP-variant connection. (*See* my discussions above at Section XI.A.1.c.2.)

256. Thus, it is my opinion that the *Wookey-Eggert* combination discloses and/or suggests the features as recited in claim element 1.f.

XII. CONCLUSION

257. 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 10, 2021 By: _____
Bill Lin, Ph.D.