

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

META PLATFORMS, INC.; INSTAGRAM, INC.; WHATSAPP LLC; META
PLATFORMS TECHNOLOGIES, LLC; AND GIPHY, INC.

Petitioners

v.

VL COLLECTIVE IP LLC

Patent Owner

Case IPR2023-00923
U.S. Patent 7,266,682

**DECLARATION OF STUART G. STUBBLEBINE, PH.D. IN SUPPORT OF
PATENT OWNER'S RESPONSE**

TABLE OF CONTENTS

I.	SCOPE AND SUMMARY OF OPINIONS.....	7
II.	MATERIALS CONSIDERED	9
III.	QUALIFICATIONS AND PROFESSIONAL BACKGROUND	9
IV.	UNDERSTANDING OF LEGAL PRINCIPLES	13
	A. Obviousness.....	14
	B. Claim Construction	16
	C. Priority Date of the '682 patent.....	16
V.	LEVEL OF ORDINARY SKILL	17
VI.	TECHNOLOGY BACKGROUND.....	17
	A. OSI Reference Model and Layered Processing	17
	B. Real-Time Transport Protocol (“RTP”) (RFC 1889).....	22
	1. RTP provides transport functions for applications transmitting real-time data.	22
	2. The RTP specification provides for encryption at or above the RTP level but expressly does not provide authentication or integrity at the RTP level.	24
	3. The RTP packet includes a fixed header and variable- sized payload.	26
	C. Transmission Control Protocol (“TCP”) (RFC 793).....	31
	D. User Datagram Protocol (“UDP”) (RFC 768)	35
	E. Ethernet (IEEE 802.3)	36
VII.	SUMMARY OF THE '682 PATENT	38
	A. Overview of the '682 Patent.....	38

VIII.	CLAIM CONSTRUCTION	42
A.	“application layer” and “application protocol”	43
IX.	THE ASSERTED REFERENECS	47
A.	Handley (EX1005)	47
1.	Handley’s architecture relates to Internet multicasting.	47
2.	Handley segregates real-time traffic over UDP from non- real-time traffic over TCP	50
3.	Handly carries RTP traffic over UDP for “real-time” delivery.....	53
4.	Handley relies on RTP for packet encryption.....	56
B.	Basturk (EX1006).....	57
1.	Basturk teaches a transport layer implementation of a transport protocol that extends standard transport level reliability features to adversarial settings.	57
2.	Basturk is a transport level protocol, not an application level protocol.....	61
3.	The cryptographic functions of Basturk’s protocol are performed as a transport level protocol on a transport layer.....	64
4.	Basturk’s “on-the-fly” signature computation is not enabled by the use of a trailer.	72
C.	PacketCable (EX1007)	81
1.	PacketCable’s MTA provides transport services.....	81
X.	DR. HOUH DOES NOT SHOW THAT ANY CHALLENGED CLAIM WOULD HAVE BEEN OBVIOUS	91

A.	Dr. Houh does not demonstrate that Handley in combination with Basturk (Ground I) discloses or renders obvious any challenged claim.....	91
1.	Dr. Houh does not demonstrate that a POSITA would have been motivated to combine Handley and Basturk to arrive at the claimed invention.....	91
2.	Dr. Houh does not demonstrate that a POSITA would have found claim 1 obvious over Handley and Basturk.	100
3.	Independent Claims 25-28	110
4.	All dependent claims are patentable over Handley in combination with Basturk.	110
B.	Dr. Houh does not demonstrate that PacketCable in combination with Handley (Ground II) discloses or renders obvious any challenged claim.	110
1.	Dr. Houh fails to show that a POSITA would have found claim 1 obvious over PacketCable and Handley.	110
2.	Independent Claims 25-28	125
3.	Dr. Houh does not demonstrate that a POSITA would have found claims 22 and 24 obvious over PacketCable and Handley.	126
4.	All dependent claims are patentable over PacketCable in combination with Handley.....	127
C.	Dr. Houh does not demonstrate that PacketCable in combination with Basturk (Ground III) discloses or renders obvious any challenged claim.	127
1.	Dr. Houh does not demonstrate that a POSITA would have been motivated to combine PacketCable with Basturk to arrive at the claimed invention.	127

2.	Dr. Houh fails to show that a POSITA would have found claim 1 obvious over PacketCable and Basturk.	132
3.	Independent Claims 25-28	133
4.	Dr. Houh does not demonstrate that a POSITA would have found claims 22 and 24 obvious over PacketCable and Basturk.	133
5.	All dependent claims are patentable over PacketCable in combination with Basturk.	134
D.	Dr. Houh does not demonstrate that that Handley in combination with Basturk and PacketCable (Ground IV) discloses or renders obvious claims 22 and 24.	134
1.	[22] The method as claimed in claim 21, wherein the receiver and transmitter are each disposed in at least one of a switching node and a switching installation.	134
2.	[24] The method as claimed in claim 23, wherein the receiver and transmitter are each disposed in at least one of a switching node and a switching installation.	134

EXHIBIT LIST

Exhibit	Description
2001	Declaration of Jaime F. Cardenas-Navia in Support of Motion for Admission Pro Hac Vice
2002	The Secure Real-time Transport Protocol (SRTP)
2003	Compare Results for EX1005
2004	Declaration of Stuart Stubblebine in Support of Patent Owner's Response
2005	Curriculum Vitae of Stuart G. Stubblebine, Ph.D.
2006	Transmission Control Protocol: RFC 793 (https://www.rfc-editor.org/rfc/pdf/rfc793.txt.pdf)
2007	User Datagram Protocol: RFC 768 (https://www.rfc-editor.org/rfc/pdf/rfc768.txt.pdf)
2008	Shai Halevi and Hugo Krawczyk, <i>MMH: Software Message Authentication in the Gbit/Second Rates</i>

I. SCOPE AND SUMMARY OF OPINIONS

1. I, Stuart G. Stubblebine, Ph.D., am making this declaration in support of VL Collective IP LLC's Patent Owner's Response in the matter of IPR2023-00923 before the Patent Trial and Appeal Board (PTAB) of the United States Patent and Trademark Office, which involves Meta Platform, Inc. et al.'s petition for *inter partes* review of U.S. Patent No. 7,266,682 ("the '682 patent"). I am being compensated for my work in this matter at my normal hourly rate. My compensation does not depend in any way on the outcome of this proceeding or the opinions that I provide.

2. I have been asked to analyze the Petition (Paper 3) and Dr. Houh's declaration (EX1003) in support thereof, as well as the '682 patent and its file history. I have been asked to provide my opinions about certain obviousness theories and opinions in Dr. Houh's declaration. I have also been asked to provide opinions about how the person of ordinary skill in the art ("POSITA") would have understood and interpreted the '682 patent, including its claims; how a POSITA would have understood and interpreted the Petition's alleged prior art references; and a POSITA's knowledge of the state of the art as it relates to addressing these obviousness theories and opinions. My opinions and conclusions are set forth in this

Declaration. I understand that this Declaration is being submitted in connection with Patent Owner's Response to the Petition.

3. In preparing this Declaration, I have reviewed the declaration of Dr. Houh (EX1003) and certain materials cited therein; and Patent Owner's Preliminary Response and materials cited therein.

4. For purposes of this Declaration, I was asked to consider claims 1-28 of the '682 patent, which I understand are the challenged claims. *See* EX1003, ¶7 footnote 1. I understand that claims 1 and 25-28 are the independent claims.

5. This Declaration and its exhibits, in addition to my education, training, and experience in technology fields pertinent to the '682 patent, form the basis of my opinion.

6. I am not offering any conclusions as to the ultimate determinations that I understand the Patent Trial and Appeal Board ("Board") will make in this proceeding. I am solely providing my opinions on the technical interpretations, characterizations, theories, and opinions that I understand are involved in this proceeding.

7. I may offer additional opinions based on my further review of the materials in this case, including opinions and/or testimony of other expert declarants.

II. MATERIALS CONSIDERED

8. In forming the opinions in this Declaration, I reviewed Dr. Houh's declaration (EX1003), the '682 patent, and the '682 patent's file history. A list of the materials I considered is included as Appendix A.

III. QUALIFICATIONS AND PROFESSIONAL BACKGROUND

9. I have over 30 years of experience studying, researching, and working with computers, computer programming, and networks. I received a Bachelor of Science degree in Computer Science and Mathematics in 1983 from Vanderbilt University, a Master of Science degree in Electrical Engineering in 1988 from the University of Arizona, and a Ph.D. degree in Electrical Engineering in 1992 from the University of Maryland.

10. I served in the U.S. Army from 1984 to 1987, focusing on telecommunications. From 1985 to 1987, I was an instructor at the City Colleges of Chicago, teaching undergraduate computer science courses relating to programming and system analysis and design. Then, in 1988, as a Research Assistant in the Electrical and Computer Engineering Department at the University of Arizona, I worked on a video, telecommunication, and distributed computer architecture for conferencing. Among other things, I optimized the network design and communication protocols for the system.

11. From 1989 to 1990, I was the Director of Secure Systems Engineering at Commcrypt, where I led research and development in several security-related areas, including network and file server architectures, automated cryptographic key management, and secure e-mail. At Commcrypt, I also worked with the National Institute of Standards and Technology to establish national standards for secure programming.

12. From 1990 to 1991, while conducting my doctoral research at the University of Maryland, I also taught a laboratory component of an upper division computer engineering course. From 1991 to 1992, while continuing my doctoral research, I worked as a Computer Scientist and consultant in the Federal Systems Division of IBM. In that capacity, I analyzed the security of certain network architectures and distributed computing systems and identified significant vulnerabilities in Privacy-Enhanced Electronic Mail and the Kerberos network authentication service.

13. After receiving my Ph.D. in 1992, I had a joint appointment at the University of Southern California as a Research Assistant Professor in the Computer Science Department and as a Computer Scientist with the Information Sciences Institute, from 1992 to 1994. I continued on at USC for the next four years as an adjunct faculty member in the Computer Science department. During my time at

USC, among several other responsibilities, I advised students on research in computer networks and security. I also conducted research relating to minimizing delay and bandwidth for protecting traffic flow confidentiality in networks and contributed to the design of the Real-Time Transport Protocol (“RTP”). In fact, the RTP specification cites one of my papers, *see* EX1008 at 72, 75, and I am credited in the Acknowledgements of an earlier Internet Draft of the RTP specification for contributions to the security aspects of RTP, *see* [https:// datatracker. ietf.org/doc/ html/draft-ietf-avt-rtp-04.txt](https://datatracker.ietf.org/doc/html/draft-ietf-avt-rtp-04.txt).

14. During part of my time at USC, from 1994 to 1998, I was also a researcher at AT&T Bell Labs (later AT&T Labs – Research), performing research in computer and network security technology. While at Bell Labs, I worked on numerous projects, including projects involving research into secure Internet telephony, attacks on the IPSEC protocol and security for e-commerce services.

15. From 1998 to 2001, I was a Vice President and Cryptographer at CertCo, Inc., conducting research, design, and analysis of public key infrastructure protocols and related risk management services. Beginning in 2001, I formed Stubblebine Consulting and Stubblebine Research Labs, and began my affiliation as a professional researcher with the Computer Science department at the University of California, Davis.

16. I was an Associate Editor of Association for Computing Machinery (ACM) Transactions on Information and System Security, the premier academic journal in the area of network and computer security, and a member of its editorial board from January 2000 to April 2007. I was an invited editor for the Special Issue on Software Engineering and Security for ACM Transactions on Software Engineering and Methodology in 2000.

17. I was a member of the Program Committee, ACM Conference on Computer and Communications Security in 1996, 1997, 2002, and 2003. I was also a member of the Program Committee, Formal Methods in Security Engineering (FMSE) in 2003 and 2004. I served on the Program Committee for Financial Cryptography in 2001 and 2006. I was the Tutorial Chair, ACM Conference on Computer and Communications Security in 2000, and a Session Chair for that conference in 2000 and 2003. I was on the Program Committee for the Institute of Electrical and Electronics Engineers (IEEE) Computer Security Symposium on Research in Security and Privacy in 1994, 1996, 1997, and 1998, and a Session Chair for that Symposium in 1994 and 1998. I was on the Program Committee, European Symposium on Research in Computer Security in 1998, and the Publications Chair, ACM Conference on Computer and Communications Security, also in 1998. I served as a Session Chair, 1997 DIMACS Workshop on Design and Formal Verification of

Security Protocols, and also served on the Program Committee for the National Computer Security Conference in 1993 and 1994.

18. As a research scientist with Stubblebine Research Labs, I performed research sponsored by the National Science Foundation into security and privacy technology. Currently, as a principal of SSNOLOGY LLC, I provide consultation services in the fields of computer and network security. Furthermore, I am an inventor on 12 United States patents, all of which relate to computer and network security.

19. A copy of my curriculum vitae is included as EX2005.

IV. UNDERSTANDING OF LEGAL PRINCIPLES

20. At the outset, I want to clarify that all of my analysis and opinions in this declaration are from the perspective of a person of ordinary skill in the art at the time of invention. To the extent that portions of this declaration do not expressly include language indicating that the analysis is from the perspective of a person of ordinary skill in the art at the time of invention, it should be understood that my analysis and opinions are from the perspective of a person of ordinary skill in the art at the time of invention. For example, my opinion that a reference does not disclose or teach a claim limitation should be understood to mean that, in my opinion, a person of ordinary skill in the art at the time of invention would have interpreted the

reference as not disclosing the limitation. Likewise, my opinions regarding obviousness are from the perspective of a person of ordinary skill in the art at the time of the invention.

21. I do not offer opinions of law as I am not an attorney. The opinions I am expressing in this Declaration involve the application of my knowledge and experience to the evaluation of the '682 patent and certain art with respect to the '682 patent. The following paragraphs express my understanding of legal principles related to my analysis.

A. Obviousness

22. I understand that a claim is unpatentable under 35 U.S.C. § 103 if the differences between the claimed subject matter and the prior art are such that the claimed subject matter as a whole would have been obvious to a person of ordinary skill in the art at the time of the invention. I also understand that an obviousness analysis is based on the scope and content of the prior art, the differences between the claimed subject matter and the prior art, the level of ordinary skill in the pertinent art at the time of the invention, and objective indicia of nonobviousness, such as commercial success, industry praise, and unexpected results.

23. It is my understanding that, to find a claimed invention obvious in light of a combination of prior-art references, a person of ordinary skill in the art at the

time of the invention must have had reason to combine the teachings of the prior-art references to achieve the claimed invention, and that the person of ordinary skill must have had a reasonable expectation of succeeding in combining the teachings to arrive at the claimed invention. I further understand that a reason to combine or expectation of success is undermined if the combination or modification of the technology disclosed in the prior art would interfere with an objective of the technology disclosed in the prior art or if the prior art teaches away from making the combination or modification. I understand that a prior art reference is said to teach away from the claimed invention if a person of ordinary skill in the art at the time of the invention, upon reading the reference, would be discouraged from following the path set out in the reference, or would be led in a direction divergent from the path that was taken in the claim.

24. I understand that an assessment of what a reference discloses or teaches, for purposes of an obviousness analysis, must be conducted from the perspective of a person of ordinary skill in the art at the time of the invention. In other words, a reference discloses or teaches a claim limitation only if a person of ordinary skill in the art would at the time of the invention would have interpreted the reference as disclosing the claim limitation. I understand that reliance on common sense,

common wisdom, and common knowledge to inform an obviousness analysis must be explained with sufficient reasoning.

B. Claim Construction

25. I understand that claims and their terms are construed, that is, interpreted, in accordance with their ordinary and customary meaning as understood by a person of ordinary skill in the art at the time of the invention when read in the context of the patent's specification and prosecution history. I further understand that if the patent's specification reveals a special definition given to a claim term by the inventor, then the inventor's special definition governs, even if it differs from the term's ordinary meaning.

26. I understand that Petitioners presented no terms for construction. I further understand that Dr. Houh offered no express claim construction opinions.

C. Priority Date of the '682 patent

27. I understand that Dr. Houh does not dispute that the '682 patent is entitled to a priority date of January 18, 2000. EX1003, ¶44. For the purposes of my opinions in this Declaration, I have been asked to assume the '682 patent is entitled to the same priority date of January 18, 2000. Therefore, when I refer to a person of ordinary skill in the art at the time of the invention in my Declaration, I am referring to a person of ordinary skill in the art as of January 18, 2000.

V. LEVEL OF ORDINARY SKILL

28. Dr. Houh set forth a level of ordinary skill as follows:

A POSITA at the time of the '682 Patent would have had a Bachelor's degree in computer science, electrical engineering, or a related field, and two years of experience; or by obtaining a Master's degree in electrical engineering or a related field, but having only one to two years of experience; or by having no formal education but experience in network data transmission and computer networking of at least eight years. Additional education might compensate for less experience, and vice-versa.

EX1003, ¶45.

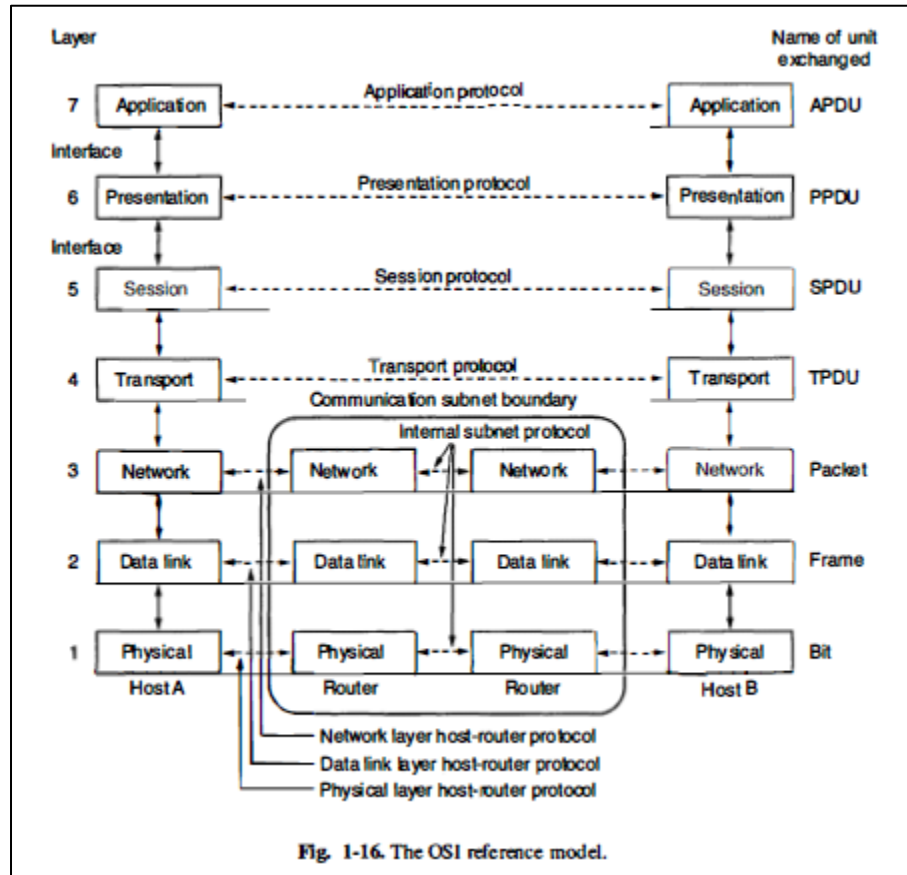
29. I met Dr. Houh's definition of a POSITA before January 18, 2000 based on my academic achievement and employment experience. *See* my qualifications and professional background in § III above.

VI. TECHNOLOGY BACKGROUND

A. OSI Reference Model and Layered Processing

30. The OSI Reference Model "is based on a proposal developed by the International Standards Organization (ISO) as a first step toward international standardization of the protocols used in the various layers." EX1009 at 49. "The OSI model has seven layers," (*i.e.*, levels of abstraction). EX1009 at 49-50. The seven OSI Reference Model layers are (starting at the "bottom") (1) Physical, (2) Data Link (or simply Link), (3) Network, (4) Transport, (5) Session, (6) Presentation, and (7) Application. EX1009 at 50-55. "[T]he OSI model itself is not

a network architecture because it does not specify the exact services and protocols to be used in each layer. It just tells what each layer should do.” EX1009 at 50.



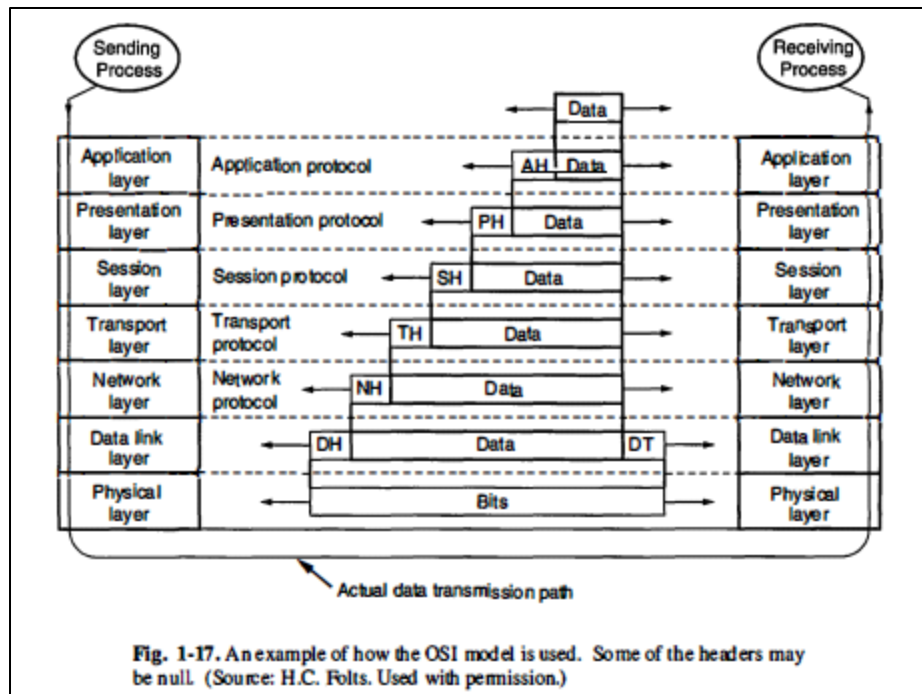
EX1009 at 50

31. To transmit data using the OSI model, a layer entity communicates with its peer entity at each layer according to one or more defined protocol for the layer. Layer 4 and above entities communicate end to end. Layer 3 and below protocols are not necessarily end to end.

32. Encapsulation and decapsulation are processes that wrap and unwrap data with protocol-specific information as it moves through the layers of a network protocol stack, such as the OSI model. This allows for modular and interoperable network communication.

33. Data is first passed from an application sending process at the highest layer (*i.e.*, the application layer). The application layer adds a header and passes the resulting data unit (“protocol data unit” or “PDU,” sometimes more informally termed a “packet” although “packet” is the name for the PDU at the network layer or ISO layer 3) to the next lower layer, which receives the higher-layer’s PDU as its payload, adds a header, and sends the resulting PDU to the next lower layer. The process is repeated until the message is eventually sent down to the lowest layer, called the physical layer, and across the physical network. As shown in the figures above and below, the unit of exchange (or protocol data unit) at the physical layer is bits because communication is bit by bit to another machine at the physical layer (in conjunction with layer 2 link layer).

34. Upon receipt at another machine the process is performed in reverse by stripping off the header at each layer (starting at the lowest layer) and passing up the resulting PDU to a higher layer until it reaches the receiving process.



EX1009 at 55

35. The active elements that perform services at each layer may be referred to as “entities” (*e.g.*, software or hardware), where entities in the same layer on the sending and receiving end are known as “peer” entities. EX1009 at 43. The function of each layer ordinarily is to provide services to the layer above it, which are provided at the layers’ interface. EX1009 at 43. As explained by Tanenbaum, “[t]he entities in layer n implement a service used by layer $n + 1$,” where “layer n is called the service provider and layer $n + 1$ is called the service user,” and “[l]ayer n may use the services of layer $n - 1$ in order to provide its service.” EX1009 at 43. In order for messages to be sent in this way, a completed “packet” (*i.e.*, PDU) is formed

by adding a header to the information destined for the peer entity and passing the completed packet to the lower layer. EX1009 at 43-44.

1.3.3. Interfaces and Services

The function of each layer is to provide services to the layer above it. In this section we will look at precisely what a service is in more detail, but first we will give some terminology.

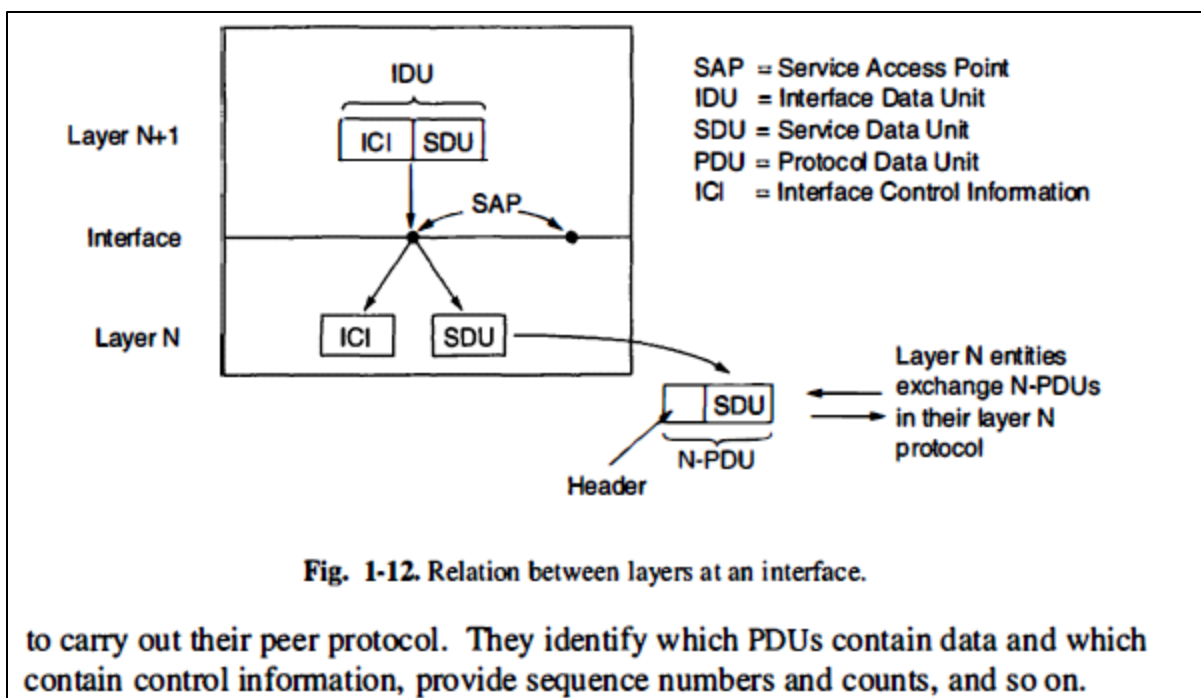
The active elements in each layer are often called **entities**. An entity can be a software entity (such as a process), or a hardware entity (such as an intelligent I/O chip). Entities in the same layer on different machines are called **peer entities**. The entities in layer n implement a service used by layer $n + 1$. In this case layer n is called the **service provider** and layer $n + 1$ is called the **service user**. Layer n may use the services of layer $n - 1$ in order to provide its service. It may offer several classes of service, for example, fast, expensive communication and slow, cheap communication.

Services are available at **SAPs (Service Access Points)**. The layer n SAPs are the places where layer $n + 1$ can access the services offered. Each SAP has an address that uniquely identifies it. To make this point clearer, the SAPs in the telephone system are the sockets into which modular telephones can be plugged, and the SAP addresses are the telephone numbers of these sockets. To call someone, you must know the callee's SAP address. Similarly, in the postal system, the SAP addresses are street addresses and post office box numbers. To send a letter, you must know the addressee's SAP address.

In order for two layers to exchange information, there has to be an agreed upon set of rules about the interface. At a typical interface, the layer $n + 1$ entity passes an **IDU (Interface Data Unit)** to the layer n entity through the SAP as shown in Fig. 1-12. The IDU consists of an **SDU (Service Data Unit)** and some control information. The SDU is the information passed across the network to the peer entity and then up to layer $n + 1$. The control information is needed to help the lower layer do its job (e.g., the number of bytes in the SDU) but is not part of the data itself.

EX1009 at 43

In order to transfer the SDU, the layer n entity may have to fragment it into several pieces, each of which is given a header and sent as a separate **PDU (Protocol Data Unit)** such as a packet. The PDU headers are used by the peer entities



EX1009 at 43-44

B. Real-Time Transport Protocol (“RTP”) (RFC 1889)

1. RTP provides transport functions for applications transmitting real-time data.

36. The RTP specification states that “RTP provides end-to-end network transport functions suitable for applications transmitting real-time data, such as audio, video or simulation data, over multicast or unicast network services.”

EX1008 at 1.

Abstract

This memorandum describes RTP, the real-time transport protocol. RTP provides end-to-end network transport functions suitable for applications transmitting real-time data, such as audio, video or simulation data, over multicast or unicast network services. RTP does not address resource reservation and does not guarantee quality-of-service for real-time services. The data transport is augmented by a control protocol (RTCP) to allow monitoring of the data delivery in a manner scalable to large multicast networks, and to provide minimal control and identification functionality. RTP and RTCP are designed to be independent of the underlying transport and network layers. The protocol supports the use of RTP-level translators and mixers.

EX1008 at 1

37. According to the RTP specification, RTP “provides end-to-end delivery services for data with real-time characteristics, such as interactive audio and video.”

EX1008 at 3. The RTP protocol indicates that “[a]pplications typically run RTP on top of UDP to make use of its multiplexing and checksum services....” EX1008 at 3.

This memorandum specifies the real-time transport protocol (RTP), which provides end-to-end delivery services for data with real-time characteristics, such as interactive audio and video. Those services include payload type identification, sequence numbering, timestamping and delivery monitoring. Applications typically run RTP on top of UDP to make use of its multiplexing and checksum services; both protocols contribute parts of the transport protocol functionality. However, RTP may be used with other suitable underlying network or transport protocols (see Section 10). RTP supports data transfer to multiple destinations using multicast distribution if provided by the underlying network.

EX1008 at 3

2. The RTP specification provides for encryption at or above the RTP level but expressly does not provide authentication or integrity at the RTP level.

38. The RTP specification provides for RTP level encryption by encrypting the whole RTP packet. EX1008 at 49. The default encryption algorithm for RTP level encryption is the Data Encryption Standard (DES) algorithm in cipher block chaining (CBC) mode.

9.1 Confidentiality

Confidentiality means that only the intended receiver(s) can decode the received packets; for others, the packet contains no useful information. Confidentiality of the content is achieved by encryption.

When encryption of RTP or RTCP is desired, all the octets that will be encapsulated for transmission in a single lower-layer packet are encrypted as a unit. For RTCP, a 32-bit random number is prepended to the unit before encryption to deter known plaintext attacks. For RTP, no prefix is required because the sequence number and timestamp fields are initialized with random offsets.

EX1008 at 49

The presence of encryption and the use of the correct key are confirmed by the receiver through header or payload validity checks. Examples of such validity checks for RTP and RTCP headers are given in Appendices A.1 and A.2.

The default encryption algorithm is the Data Encryption Standard (DES) algorithm in cipher block chaining (CBC) mode, as described in Section 1.1 of RFC 1423 [21], except that padding to a multiple of 8 octets is indicated as described for the P bit in Section 5.1. The initialization vector is zero because random values are supplied in the RTP header or by the random prefix for compound RTCP packets. For details on the use of CBC initialization vectors, see [22]. Implementations that support encryption should always support the DES algorithm in CBC mode as the default to maximize interoperability. This method is chosen because it has been demonstrated to be easy and practical to use in experimental audio and video tools in operation on the Internet. Other encryption algorithms may be specified dynamically for a session by non-RTP means.

EX1008 at 50

39. The RTP specification also provides an alternative encryption option whereby only the RTP payload data is encrypted (and not the header).

As an alternative to encryption at the RTP level as described above, profiles may define additional payload types for encrypted encodings. Those encodings must specify how padding and other aspects of the encryption should be handled. This method allows encrypting only the data while leaving the headers in the clear for applications where that is desired. It may be particularly useful for hardware devices that will handle both decryption and decoding.

EX1008 at 50

40. RTP explicitly states that “[a]uthentication and message integrity are not defined in the current specification of RTP.” EX1008 at 50. The RTP specification expresses an expectation that “authentication and integrity services will be provided by lower layer protocols in the future.” EX1008 at 50.

9.2 Authentication and Message Integrity

Authentication and message integrity are not defined in the current specification of RTP since these services would not be directly feasible without a key management infrastructure. It is expected that authentication and integrity services will be provided by lower layer protocols in the future.

EX1008 at 50

9. Security

Lower layer protocols may eventually provide all the security services that may be desired for applications of RTP, including authentication, integrity, and confidentiality. These services have recently been specified for IP. Since the need for a confidentiality service is well established in the initial audio and video applications that are expected to use RTP, a confidentiality service is defined in the next section for use with RTP and RTCP until lower layer services are available. The overhead on the protocol for this service is low, so the penalty will be minimal if this service is obsoleted by lower layer services in the future.

EX1008 at 49

3. The RTP packet includes a fixed header and variable-sized payload.

41. According to the RTP specification, an “RTP payload” is “[t]he data transported by RTP in a packet, for example audio samples or compressed video data.” EX1008 at 7. The RTP specification defines an “RTP packet” as a header and a payload: “RTP packet: A data packet consisting of the fixed RTP header, a possibly empty list of contributing sources (see below), and the payload data.” EX1008 at 7. The list of contributing sources is part of the header and includes 0-

15 fixed-length items, the number of which is identified in the header. EX1008 at 10-13.

3. Definitions

RTP payload: The data transported by RTP in a packet, for example audio samples or compressed video data. The payload format and interpretation are beyond the scope of this document.

RTP packet: A data packet consisting of the fixed RTP header, a possibly empty list of contributing sources (see below), and the payload data. Some underlying protocols may require an encapsulation of the RTP packet to be defined. Typically one packet of the underlying protocol contains a single RTP packet, but several RTP packets may be contained if permitted by the encapsulation method (see Section 10).

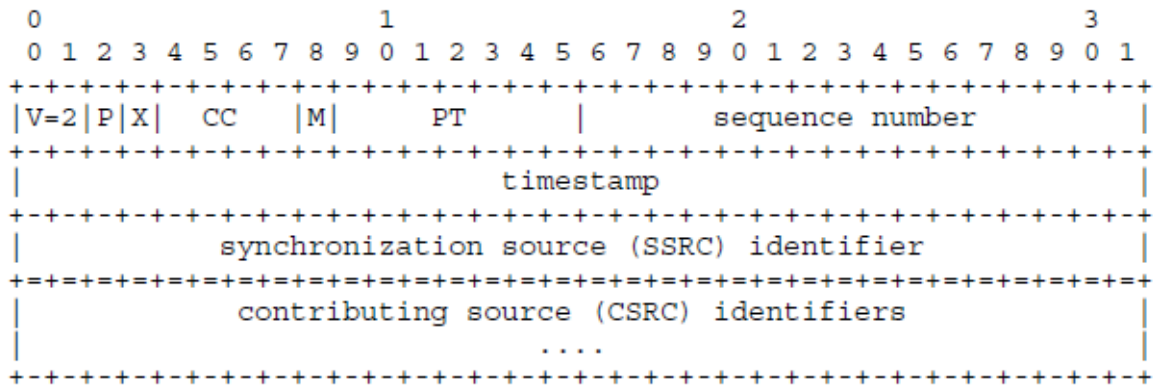
EX1008 at 7

Contributing source (CSRC): A source of a stream of RTP packets that has contributed to the combined stream produced by an RTP mixer (see below). The mixer inserts a list of the SSRC identifiers of the sources that contributed to the generation of a particular packet into the RTP header of that packet. This list is called the CSRC list. An example application is audio conferencing where a mixer indicates all the talkers whose speech was combined to produce the outgoing packet, allowing the receiver to indicate the current talker, even though all the audio packets contain the same SSRC identifier (that of the mixer).

EX1008 at 8

5.1 RTP Fixed Header Fields

The RTP header has the following format:



The first twelve octets are present in every RTP packet, while the list of CSRC identifiers is present only when inserted by a mixer. The fields have the following meaning:

EX1008 at 10

CSRC count (CC): 4 bits

The CSRC count contains the number of CSRC identifiers that follow the fixed header.

EX1008 at 11

CSRC list: 0 to 15 items, 32 bits each

The CSRC list identifies the contributing sources for the payload contained in this packet. The number of identifiers is given by the CC field. If there are more than 15 contributing sources, only 15 may be identified. CSRC identifiers are

inserted by mixers, using the SSRC identifiers of contributing sources. For example, for audio packets the SSRC identifiers of all sources that were mixed together to create a packet are listed, allowing correct talker indication at the receiver.

EX1008 at 11-12

42. RTP payload length is not defined by RTP and is limited only by lower level protocols.

10. RTP over Network and Transport Protocols

This section describes issues specific to carrying RTP packets within particular network and transport protocols. The following rules apply unless superseded by protocol-specific definitions outside this specification.

RTP relies on the underlying protocol(s) to provide demultiplexing of RTP data and RTCP control streams. For UDP and similar protocols, RTP uses an even port number and the corresponding RTCP stream uses the next higher (odd) port number. If an application is supplied with an odd number for use as the RTP port, it should replace this number with the next lower (even) number.

RTP data packets contain no length field or other delineation, therefore RTP relies on the underlying protocol(s) to provide a length indication. The maximum length of RTP packets is limited only by the underlying protocols.

EX1008 at 51

43. The RTP header includes an extension bit “X” that, when set, means that “the fixed header is followed by exactly one header extension” to support additional information. EX1008 at 14-15.

extension (X): 1 bit

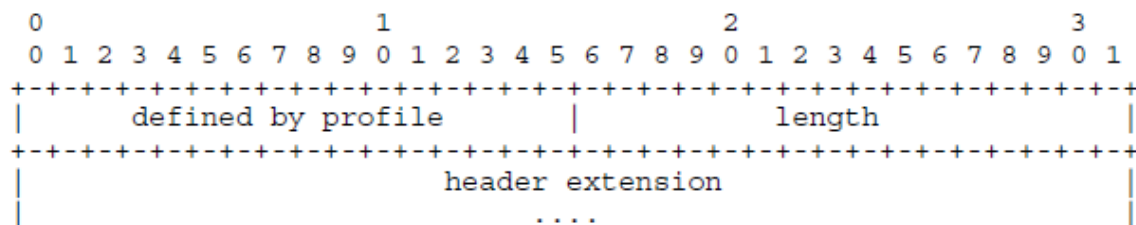
If the extension bit is set, the fixed header is followed by exactly one header extension, with a format defined in Section 5.3.1.

EX1008 at 11

5.3.1 RTP Header Extension

An extension mechanism is provided to allow individual implementations to experiment with new payload-format-independent functions that require additional information to be carried in the RTP data packet header. This mechanism is designed so that the header extension may be ignored by other interoperating implementations that have not been extended.

Note that this header extension is intended only for limited use. Most potential uses of this mechanism would be better done another way, using the methods described in the previous section. For example, a profile-specific extension to the fixed header is less expensive to process because it is not conditional nor in a variable location. Additional information required for a particular payload format should not use this header extension, but should be carried in the payload section of the packet.



If the X bit in the RTP header is one, a variable-length header extension is appended to the RTP header, following the CSRC list if present. The header extension contains a 16-bit length field that counts the number of 32-bit words in the extension, excluding the four-octet extension header (therefore zero is a valid length). Only a single extension may be appended to the RTP data header. To allow multiple interoperating implementations to each experiment independently with different header extensions, or to allow a particular implementation to experiment with more than one type of header extension, the first 16 bits of the header extension are left open for distinguishing identifiers or parameters. The format of these 16 bits is to be defined by the profile specification under which the implementations are operating. This RTP specification does not define any header extensions itself.

EX1008 at 14-15

44. RTP looks for information in the header to check for validity issues, including “to verify that incoming packets have been correctly decrypted.” EX1008 at 59.

A.1 RTP Data Header Validity Checks

An RTP receiver should check the validity of the RTP header on incoming packets since they might be encrypted or might be from a different application that happens to be misaddressed. Similarly, if encryption is enabled, the header validity check is needed to verify that incoming packets have been correctly decrypted, although a failure of the header validity check (e.g., unknown payload type) may not necessarily indicate decryption failure.

EX1008 at 59

C. Transmission Control Protocol (“TCP”) (RFC 793)

45. The “Transmission Control Protocol,” or “TCP” as it is commonly known, is a reliable networking protocol that “is intended for use as a highly reliable host-to-host protocol between hosts in packet-switched computer communication networks, and in interconnected systems of such networks.” EX2006 at 7. As the TCP specification states, “TCP is a connection-oriented, end-to-end reliable protocol designed to fit into a layered hierarchy of protocols which support multi-network applications.” EX2006 at 7. The specification further states that “[t]he TCP is intended to provide a reliable process-to-process communication service in a multinetwork environment.” EX2006 at 8. TCP is a reliability protocol, meaning that it provides a “reliability” feature. EX2006 at 10. Here, “reliability” is the ability

to “recover from data that is damaged, lost, duplicated, or delivered out of order by the internet communication system. This is achieved by assigning a sequence number to each octet transmitted, and requiring a positive acknowledgment (ACK) from the receiving TCP. If the ACK is not received within a timeout interval, the data is retransmitted.” EX2006 at 10. Accordingly, as stated in the TCP specification, “[t]ransmission is made reliable via the use of sequence numbers and acknowledgments.” EX2006 at 16.

The Transmission Control Protocol (TCP) is intended for use as a highly reliable host-to-host protocol between hosts in packet-switched computer communication networks, and in interconnected systems of such networks.

EX2006 at 7

TCP is a connection-oriented, end-to-end reliable protocol designed to fit into a layered hierarchy of protocols which support multi-network applications. The TCP provides for reliable inter-process communication between pairs of processes in host computers attached to distinct but interconnected computer communication networks. Very few assumptions are made as to the reliability of the communication protocols below the TCP layer. TCP assumes it can obtain a simple, potentially unreliable datagram service from the lower level protocols. In principle, the TCP should be able to operate above a wide spectrum of communication systems ranging from hard-wired connections to packet-switched or circuit-switched networks.

EX2006 at 7

The TCP is intended to provide a reliable process-to-process communication service in a multinet environment. The TCP is intended to be a host-to-host protocol in common use in multiple networks.

EX2006 at 8

As noted above, the primary purpose of the TCP is to provide reliable, securable logical circuit or connection service between pairs of processes. To provide this service on top of a less reliable internet communication system requires facilities in the following areas:

- Basic Data Transfer
- Reliability
- Flow Control
- Multiplexing
- Connections
- Precedence and Security

The basic operation of the TCP in each of these areas is described in the following paragraphs.

EX2006 at 9

Reliability:

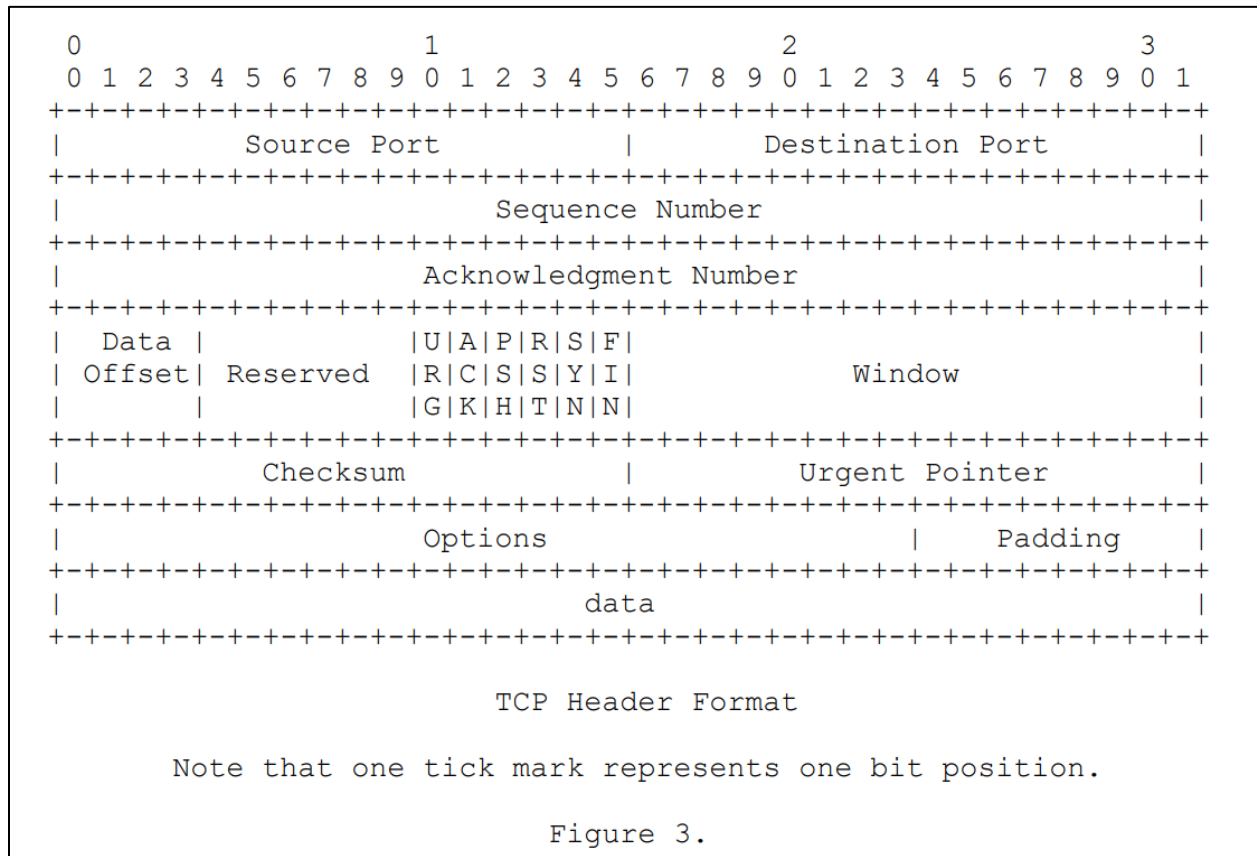
The TCP must recover from data that is damaged, lost, duplicated, or delivered out of order by the internet communication system. This is achieved by assigning a sequence number to each octet transmitted, and requiring a positive acknowledgment (ACK) from the receiving TCP. If the ACK is not received within a timeout interval, the data is retransmitted. At the receiver, the sequence numbers are used to correctly order segments that may be received out of order and to eliminate duplicates. Damage is handled by adding a checksum to each segment transmitted, checking it at the receiver, and discarding damaged segments.

EX2006 at 10

Transmission is made reliable via the use of sequence numbers and acknowledgments. Conceptually, each octet of data is assigned a sequence number. The sequence number of the first octet of data in a segment is transmitted with that segment and is called the segment sequence number. Segments also carry an acknowledgment number which is the sequence number of the next expected data octet of transmissions in the reverse direction. When the TCP transmits a segment containing data, it puts a copy on a retransmission queue and starts a timer; when the acknowledgment for that data is received, the segment is deleted from the queue. If the acknowledgment is not received before the timer runs out, the segment is retransmitted.

EX2006 at 16

46. The TCP header includes a checksum:



EX2006 at 21

47. The TCP checksum is used to check for errors:

Checksum: 16 bits

The checksum field is the 16 bit one's complement of the one's complement sum of all 16 bit words in the header and text. If a segment contains an odd number of header and text octets to be checksummed, the last octet is padded on the right with zeros to form a 16 bit word for checksum purposes. The pad is not transmitted as part of the segment. While computing the checksum, the checksum field itself is replaced with zeros.

EX2006 at 22

D. User Datagram Protocol (“UDP”) (RFC 768)

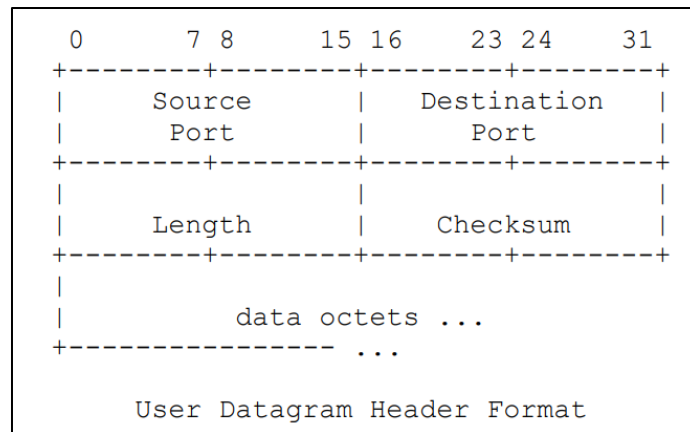
48. The “User Datagram Protocol,” or “UDP” as it is commonly known, is a non-reliable networking protocol that “is defined to make available a datagram mode of packet-switched computer communication in the environment of an interconnected set of computer networks.” EX2007 at 1. The UDP specification indicates that “[UDP] is transaction oriented, and delivery and duplicate protection are not guaranteed. Applications requiring ordered reliable delivery of streams of data should use the Transmission Control Protocol (TCP).” EX2007 at 1.

This User Datagram Protocol (UDP) is defined to make available a datagram mode of packet-switched computer communication in the environment of an interconnected set of computer networks. This protocol assumes that the Internet Protocol (IP) [1] is used as the underlying protocol.

This protocol provides a procedure for application programs to send messages to other programs with a minimum of protocol mechanism. The protocol is transaction oriented, and delivery and duplicate protection are not guaranteed. Applications requiring ordered reliable delivery of streams of data should use the Transmission Control Protocol (TCP) [2].

EX2007 at 1

49. UDP checksums are placed in the Header not at the end of the PDU. The UDP protocol is a transport protocol (ISO layer 4). The UDP header is placed in the packet payload portion of the lower layer protocol such as IP or Internet Protocol at the networking layer (ISO layer 3). The UDP header may contain the following fields shown in the following figure.



EX2007 at 1

50. The header may contain a checksum. The checksum is computed over the IP header, UDP header, UDP payload data, and padding if necessary.

Checksum is the 16-bit one's complement of the one's complement sum of a pseudo header of information from the IP header, the UDP header, and the data, padded with zero octets at the end (if necessary) to make a multiple of two octets.

The pseudo header conceptually prefixed to the UDP header contains the source address, the destination address, the protocol, and the UDP length. This information gives protection against misrouted datagrams. This checksum procedure is the same as is used in TCP.

EX2007 at 2

51. This is an example of where a message integrity code is placed in the header in a higher layer protocol.

E. Ethernet (IEEE 802.3)

52. Ethernet, a protocol operating at the physical layer and link layer (ISO Layers 1 & 2), is designed to process incoming frames bit by bit as they are received as the unit of exchange at the physical layer is the bit. See EX1014 at 32, 142-143.

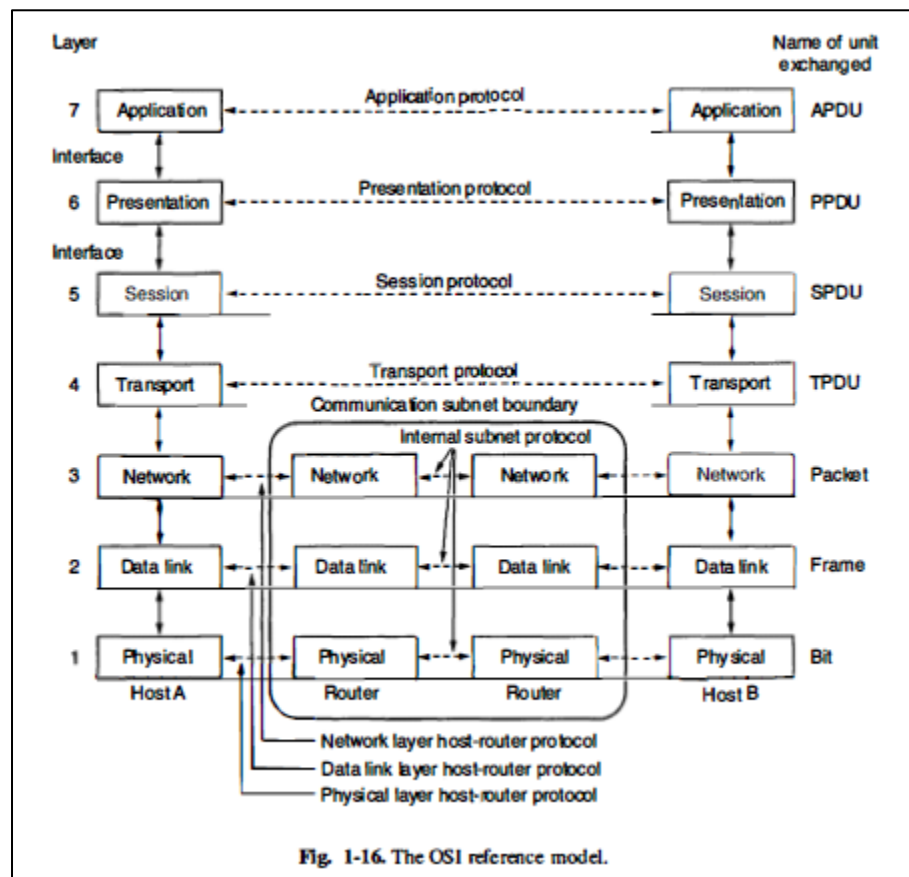
VL Collective Exhibit 2004

IPR2023-00923

Meta Platforms, Inc. v. VL Collective IP LLC

Page 36 of 136

The physical circuitry in Ethernet devices decodes the waveform bit by bit in real-time, extracting the bits and passing them up to higher layers for processing. *Id.* As discussed with respect to the OSI Reference Model, the protocol data unit of the physical layer is different from higher layer protocols. At the physical layer, bit-by-bit transmission occurs (in conjunction with the link layer).



EX1009 at 50

4.1.2.1 Normal Operation

4.1.2.1.1 Transmission Without Contention. When a LLC sublayer requests the transmission of a frame, the Transmit Data Encapsulation component of the CSMA/CD MAC sublayer constructs the frame from the LLC-supplied data. It appends a preamble and a Start Frame Delimiter to the beginning of the frame. Using information passed by the LLC sublayer, the CSMA/CD MAC sublayer also appends a PAD at the end of the MAC information field of sufficient length to ensure that the transmitted frame length satisfies a minimum frame size requirement (see 4.2.3.3). It also appends destination and source addresses, a length count field and a frame check sequence to provide for error detection. The frame is then handed to the Transmit Media Access Management component in the MAC sublayer for transmission.

Transmit Media Access Management then attempts to avoid contention with other traffic on the medium by monitoring the carrier sense signal provided by the PLS component and deferring to passing traffic. When the medium is clear, frame transmission is initiated (after a brief interframe delay to provide recovery time for other CSMA/CD MAC sublayers and for the physical medium). The MAC sublayer then provides a serial stream of bits to the PLS interface for transmission.

EX1014 at 32

C2.3 Transmit Component Action Descriptions

Construct Frame. This action encapsulates the data field with the Preamble, SFD, DA, SA, Length, PAD and FCS fields.

Start Frame Transmission. This action initiates bit transmission of the frame.

EX1014 at 142

Transmit. The transmit component is actively transmitting bits onto the medium.

EX1014 at 143.

VII. SUMMARY OF THE '682 PATENT

A. Overview of the '682 Patent

53. The '682 patent addresses a longstanding problem in the transmission of real-time audio and visual data over the Internet. In the mid-1990s, the Real Time Transport Protocol, or RTP, was developed by the Internet Engineering Task Force

VL Collective Exhibit 2004

IPR2023-00923

Meta Platforms, Inc. v. VL Collective IP LLC

Page 38 of 136

(IETF), which was designed for end-to-end real-time transfer of streaming media. *See* EX1008 1, 72-73.

54. As the RTP specification itself acknowledges, RTP had security vulnerabilities. EX1008 at 72. Transmission networks utilizing RTP were susceptible to security breaches at the transmitter, during transmission, and at the receiver. *See* EX1008 at 72; EX1001 at 2:44–3:10. Attackers could overload the network by sending mass data to receivers, which can harm system availability and performance. EX1001 at 2:47–64. In that regard, although it was possible for RTP traffic to be sent behind firewalls, EX1008 at 72, firewalls did not solve such problems because it was a “simple matter for an attacker to send [its] meaningless (mass) data to addresses within the private network as well, that is to say, behind the firewall.” EX1001 at 2:65–3:10.

55. In one approach, the IPSEC protocol, “data packets of the Internet protocol (IP data packets) [could] be encapsulated and [could] be protected in terms of confidentiality and/or integrity.” EX1001 at 3:16–41. It included a cookie-based mechanism, linking fast one-way hash functions, random numbers, and IP addresses to one another. *Id.* But that solution had a security gap, in that “first ‘cookie’ transmitted from the transmitter to the receiver” was unprotected. *Id.* 3:30–34. And the cookie method could not ensure protection against unwanted mass data on the

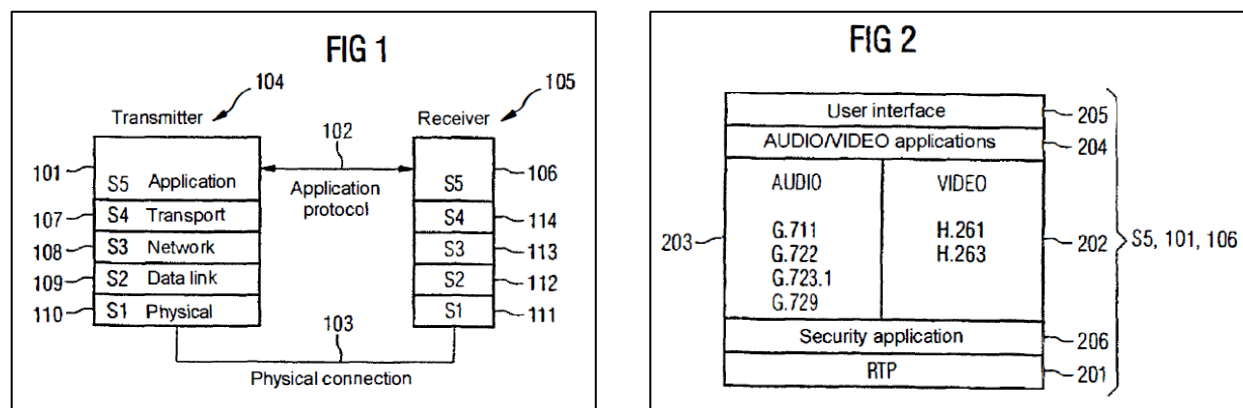
application layer (for the application protocol) under real-time conditions. *Id.* 3:34–38. Structurally, it was not possible to link IPSEC to the security functions of an application layer. *Id.* 3:38–41.

56. The RTP specification expressly did not define authentication or message integrity services (or the necessary key management infrastructure) for RTP, expecting instead that authentication and integrity services would be provided by lower layer protocols (*i.e.*, protocols for different, lower layers in the protocol stack) in the future. EX1008 at 50.

57. However, the '682 patent's inventors recognized and solved RTP's security problems with a solution of their own. The '682 patent provides a solution to RTP's security flaws by ensuring, at the receiver, that data being transmitted is not insecure or unwanted. *See* EX1001 3:45-48. Specifically, the transmitter is used to insert authentication data into the RTP data packets before transmission. *See id.* 3:49-55, 7:6-23. The authentication data is then transmitted together with the data packet and analyzed by the receiver to ensure that the transmitter and the receiver know each other. *See id.* Fig. 3, 3:49-55, 5:30-38. 7:6-23. If the receiver knows the transmitter, the data is processed; if not, the data is rejected. *See id.* 7:14-21.

58. The '682 patent teaches that authenticating RTP packets on the application layer (as an application protocol), as claimed in the '682 patent, is

extremely advantageous from both a user and network perspective. *See id.* 3:49-4:30, 7:24-48. The “application layer” for purposes of the ’682 patent refers to “the layers above the transport layer (layers above 4)” combined, and the “application protocol” refers to “the protocol for communication between a transmitter application layer situated above the transport layer and a receiver application layer situated above the transport layer.” *Id.* 2:6–11. Thus, with reference to Figures 1 and 2, the patent teaches an example where “directly above the layer 4 is the real time transport protocol (RTP)” in “the application layer (layer 5).” *Id.* 6:33–51, Figs. 1 and 2.



EX1001, Figs. 1 and 2

59. Authenticating RTP packets in the application layer permits reduced size of authentication data, because the authentication is of smaller data PDUs than at other layers, which improves speed and real-time availability of the entire communication system. *See id.* at 3:60-4:10, 7:24-36. Another benefit is that it

increases the transmission system's performance and reduces the implementation complexity of the system. *See id.* at 4:11-22. Once the receiver determines that the transmitter is known, the message is immediately accepted. *See id.* at 4:3-7. If the receiver determines that the transmitter is not known, the message is immediately rejected, which means that no buffering or loading are performed, and no further action is needed. *See id.* at 4:7-9.

VIII. CLAIM CONSTRUCTION

60. In my opinion, a person of ordinary skill in the art would have interpreted the following claim terms, when read in the context of '682 patent's specification and prosecution history, to have the meanings identified in the table below:

Term	Meaning to a POSITA
"application layer"	"the layers above the transport layer (layers above 4)"
"application protocol"	"the protocol for communication between a transmitter application layer situated above the transport layer and a receiver application layer situated above the transport layer"

A. “application layer” and “application protocol”

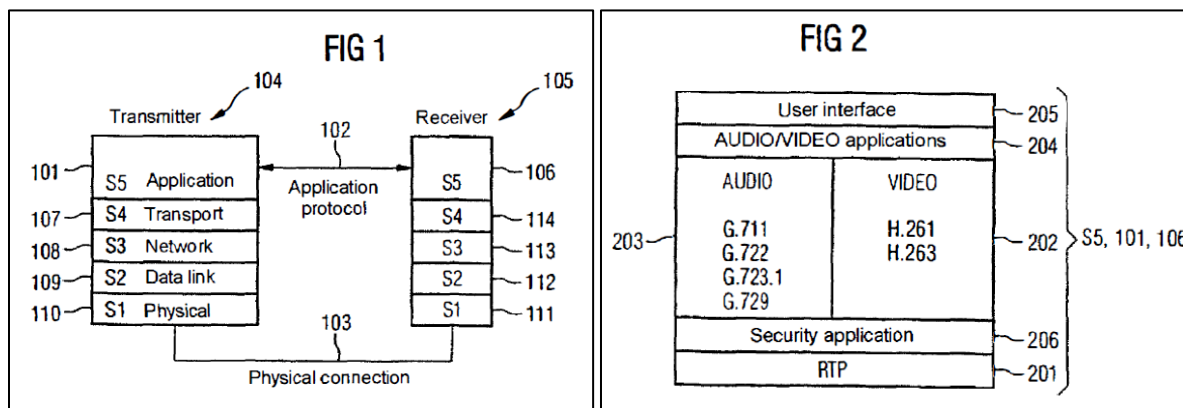
61. Independent claim 1 requires “providing transmitter-to-receiver authentication at a Real Time Transport Protocol (RTP) packet level *as an application protocol on an application layer.*” EX1001, claims 1, 25-28. In my opinion, a person of ordinary skill in the art would have understood the specification of the ’682 patent expresses a clear intent to define “application layer” and “application protocol” as stated above.

62. The context for a person of ordinary skill in the art to interpret the ’682 patent’s “application layer” and “application protocol” terms includes the ’682 patent’s explanation that “[t]he OSI reference model (OSI model for short) from the International Standards Organization (ISO) has seven layers, each of which has a different functionality (in terms of abstraction).” EX1001 at 1:19–45. (describing layers 1 through 7). The ’682 patent explains that “[i]n the OSI model ... layer 4, a transport layer, uses a transport protocol ... layer 5 corresponds to a session layer using a session protocol, layer 6 corresponds to a presentation layer using a presentation protocol, and layer 7 corresponds to an application layer using an application protocol.” EX1001, 1:22-37. These explanations about the 7-layer OSI model are consistent with how a person of ordinary skill in the art would have understood the OSI model generally. *See* analysis in § V.A. above.

63. The '682 patent further explains, however, that “[i]n practice, many applications do not always permit exact nomination of strict boundaries specifically between the upper protocol layers, particularly layers 5 to 7,” such that “the three layers above the transport layer 4 combine to form an ‘application layer’ for which . . . an ‘application protocol’ is used.” EX1001 at 1:19–45 (emphasis added). Furthermore, the specification states that “[w]hen referring to application layer below, this *denotes* the layers above the transport layer (layers above 4)” and “[t]he application protocol *also denotes* the protocol for communication between a transmitter application layer situated above the transport layer and a receiver application layer situated above the transport layer.” EX1001 2:6-11 (emphasis added). A person of ordinary skill in the art would have understood these statements, which include the signals of “when referring to . . . below,” “denotes” and “also denotes,” to express an intent by the author to set forth definitions for understanding those terms according to the definitions provided.

64. A person of ordinary skill in the art would have also understood those definitions to be consistent with the '682 patent's description of an exemplary protocol stack including an application layer and possible protocols in Figures 1 and 2, further confirming the patent's intent to provide definitions for understanding its technical disclosures. EX1001 at 5:65-6:2, Figure 1 (“protocol architecture

(protocol stack)” showing layers above “S4 Transport” as a combined “S5 Application” layer), Figure 2 (showing a multi-part “application layer with possible protocols”). The application layer with possible protocols of Figure 2 is consistent with the description of Figure 1.



EX1001, Figs. 1 and 2

65. It is therefore my opinion that a person of ordinary skill in the art would have understood the specification of the '682 patent to clearly define “application layer” to mean “the layers above the transport layer (layers above 4),” and “application protocol” to mean “the protocol for communication between a transmitter application layer situated above the transport layer and a receiver application layer situated above the transport layer.” EX1001 2:6-11.

66. A person of ordinary skill in the art would have understood that by setting forth these definitions in a preliminary section (“Background of the Invention”), the '682 patent intends to define how these terms should be interpreted

when referenced “below” in the other parts of the patent. *See* EX1001, 2:6-11. As a person of ordinary skill in the art would have known, it was (and is) extremely common for authors of technical documents to include definitional statements of terms that readers might be familiar with from usage in other contexts, but that should be understood in a particular way for purposes of that document. This is commonly done, for example, to provide clarity, reduce ambiguity, or to provide context for other technical matters to be understood. As an example, the RTP specification (RFC 1889) includes a “Definitions” section for such purposes. EX1008 at 7-9. As another example, Handley provides and qualifies a definition of “Real-time internet traffic” to provide context for its discussions. EX1005 at 4-5.

67. Furthermore, the patent’s use of the word “denotes” would have been understood by a person of ordinary skill in the art to express an intent to define “application layer” and “application protocol.” Here, the patent explicitly states that the term “application layer” “denotes the layers above the transport layer (layers above 4)” and that “application protocol” “denotes the protocol for communication between a transmitter application layer situated above the transport layer and a receiver application layer situated above the transport layer.” EX1001 2:6–11. A person of ordinary skill in the art would have understood the use of “denotes” to indicate that “application layer” and “application protocol” is defined by what

follows. The word “denotes” is a normal English word that is used in technical and academic writing in this way, and thus would have been understood by a person of ordinary skill in the art. For example, Basturk uses “denote” and “denoted” to define symbols used throughout its paper. EX1006 at 3, 8, 11.

IX. THE ASSERTED REFERNECS

A. Handley (EX1005)

1. Handley’s architecture relates to Internet multicasting.

68. In section 2 (titled “The Technology”), Handley states that its paper is about an architecture for “so-called ‘real-time’ conferencing, including audio, video, shared whiteboards and other applications” over the Internet. EX1005 at 2.

2 The Technology

In conjunction with computers, the term “conferencing” is often used in two different ways: firstly, to refer to bulletin boards and mail list style *asynchronous* exchanges of messages between multiple users; secondly, to refer to *synchronous* or so-called “real-time” conferencing, including audio, video, shared whiteboards and other applications. This paper is about the architecture for this latter application in an Internet environment.

There are other infrastructures for conferencing in the world: POTS (Plain Old Telephone System) networks often provide voice conferencing and phone-bridges, while the ISDN provides H.320[11] for small, strictly organised video-telephony conferencing.

The architecture that has evolved in the Internet is far more general as well as being scalable to very large groups, and permits the open introduction of new media and new applications as they are devised.

The determining factors of a conferencing architecture are communication between (possibly large) groups of humans and real-time delivery of information. In the Internet, this is supported at a number of levels. The remainder of this section provides an overview of this support, and the rest of the document describes each aspect in more detail.

In a conference, information must be distributed to all the conference participants. Early conferencing systems used a fan-out of data streams, e.g., one connection between each pair of participants, which means that the same information must cross some networks more than once. The Internet architecture uses the more efficient approach of *multicasting* the information to all participants (see section 3).

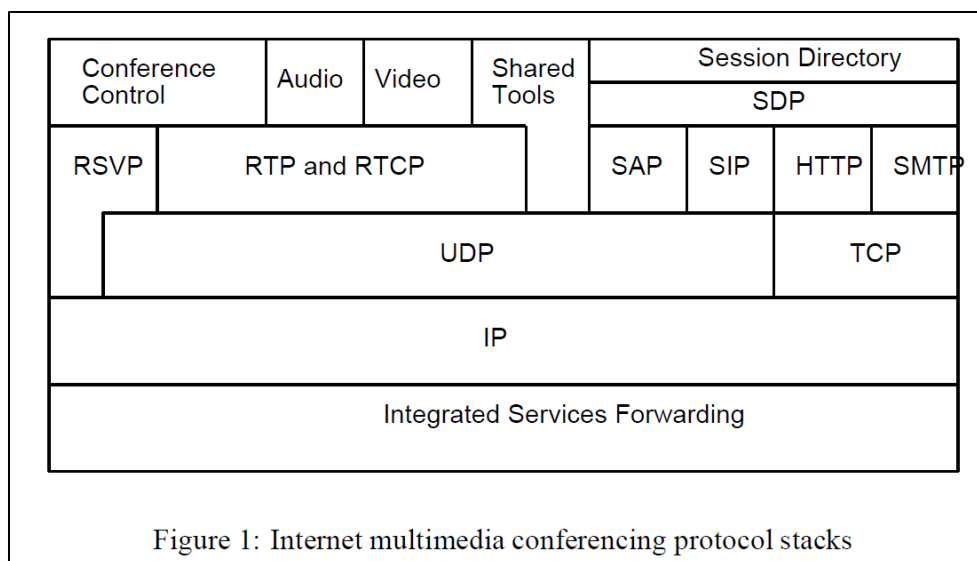
EX1005 at 1-2.

69. Handly describes “a different way of thinking about application design” for multicast distribution to a group of receivers, in which senders send messages “to the *group*,” receivers express interest in receiving data “sent to *the group*,” and components called “routers” are the elements that “conspire to deliver data from senders to receivers.” EX1005 at 2-3. Accordingly, as Handley explains, “[r]eceivers do not need to know who or where the senders are to receive traffic from them. Senders never *need* to know who the receivers are. Neither senders or receivers need care about the network topology as the network optimises delivery.” EX1005 at 3. Under Handley’s approach, a sender can send messages to “members of a group without needing to discovery who they are.” EX1005 at 3.

3 Multicast Traffic Distribution

IP multicast provides efficient many-to-many data distribution in an internet environment. It is easy to view IP multicast as simply an optimisation for data distribution, and indeed this is the case, but IP multicast can also result in a different way of thinking about application design. To see why this might be the case, examine the IP multicast service model, as described by Van Jacobson[7]:

EX1005 at 2



EX1005 at 3

- Senders just send to *the group*
- Receivers express an interest in receiving data sent to *the group*
- Routers conspire to deliver data from senders to receivers

With IP multicast, the group is indirectly identified by a single IP class-D multicast address.

Several things are important about this service model from an architectural point of view. Receivers do not need to know who or where the senders are to receive traffic from them. Senders never *need* to know who the receivers are. Neither senders or receivers need care about the network topology as the network optimises delivery.

An IP multicast group is scalable because information about group membership and group changes at the IP level are kept local to routers near the relevant members. How this is performed depends on the particular multicast routing scheme in use local to the member, and although it is not a trivial task, several solutions do exist and therefore multicast routing will not be discussed in detail here. For more detailed information on multicast routing, see [6][5][3][1][17]. Typically, as a group with s senders and r receivers increases in size, state in routers scales $O(s)$ or $O(1)$ depending on the routing scheme in use. This state may be in on-tree routers for newer so called sparse-mode algorithms such as PIM, or in off-tree routers for older so-called dense-mode algorithms such as DVMRP. Thus the most scalable current multicast routing algorithms require $O(1)$ state in on-tree routers, and hence the total routing state scales $O(g)$ in a router that is on-tree for g groups. We can also envisage multicast routing schemes which require less than $O(g)$ state¹, but the requirement is not currently urgent, so none of these are currently implemented.

The level of indirection introduced by the IP class D address denominating the group solves the distributed systems binding problem, by pushing this task down into routing; given a multicast address (and UDP port), a host can send a message to the members of a group without needing to discover who they are. Similarly receivers can “tune in” to multicast data sources without needing to bother the data source itself with any form of request.

EX1005 at 3

IP multicast is a natural solution for multi-party conferencing because of the efficiency of the data distribution trees, with data being replicated in the network at appropriate points rather than in end-systems. It also avoids the need to configure special-purpose servers to support the session, which require support,

and which cause traffic concentration and can be a bottleneck. For larger broadcast-style sessions, it is essential that data-replication be carried out in a way that only requires per-receiver network-state to be local to each receiver, and that data-replication occurs within the network. Attempting to configure a tree of application-specific replication servers for such broadcasts rapidly becomes a “multicast routing” problem, and thus native multicast support is a more appropriate solution.

EX1005 at 3-4

2. Handley segregates real-time traffic over UDP from non-real-time traffic over TCP.

70. Handley’s architecture seeks to provide a “better-than-best-effort” approach to real-time delivery of audio and video data using aspects of a traditional “best-effort” datagram delivery model (*i.e.*, using the UDP protocol), rather than a traditional “non-best effort” model (*i.e.*, using the TCP protocol) that is not suitable for real-time traffic. EX1005 at 2-5, 8. Handley defines “real-time” traffic as “datagrams that are delay sensitive,” where “TCP traffic is normally not considered to be real-time traffic.” EX1005 at 4-5. Handley suggests that while best-effort services can provide real-time traffic, link congestion can “seriously degrade the service provided.” EX1005 at 4-5. Handley suggests a modified routing model where “real-time” non-TCP traffic (as defined) is segregated from and prioritized over non-real-time TCP traffic. *See* EX1005 at 5. According to Handley, its

approach of “classifying real-time traffic separately from non-real-time traffic and giving real-time traffic priority treatment ensures that real-time traffic sees minimum delays.” EX1005 at 5.

4 Internet Service Models

Traditionally the internet has provided so-called *best-effort* delivery of datagram traffic from senders to receivers. No guarantees are made regarding when or if a datagram will be delivered to a receiver, however datagrams are normally only dropped when a router exceeds a queue size limit due to congestion. The best-effort internet service model does not assume FIFO queuing, although many routers have implemented this.

With best-effort service, if a link is not congested, queues will not build at routers, datagrams will not be discarded in routers, and delays will consist of serialisation delays at each hop plus propagation delays. With sufficiently fast link speeds, serialisation delays are insignificant compared to propagation delays.

If a link is congested, with best-effort service, queuing delays will start to influence end-to-end delays, and packets will start to be lost as queue size limits are exceeded. Real-time traffic does not cope terribly well with packet loss levels of more than a few percent, although it is possible to add redundancy[10] to increase the levels at which loss becomes a problem. In the last few years a significant amount of work has also gone into providing non-best-effort services that would provide a better assurance that an acceptable quality conference will be possible.

EX1005 at 4

4.1 Non-best effort service

Real-time internet traffic is defined as datagrams that are delay sensitive. It could be argued that all datagrams are delay sensitive to some extent, but for these purposes we refer only to datagrams where exceeding

an end-to-end delay bound of a few hundred milliseconds renders the datagrams useless for the purpose they were intended. For the purposes of this definition, TCP traffic is normally not considered to be real-time traffic, although there may be exceptions to this rule.

On congested links, best-effort service queuing delays will adversely affect real-time traffic. This does not mean that best-effort service cannot support real-time traffic — merely that congested best-effort links seriously degrade the service provided. For such congested links, a better-than-best-effort service is desirable.

To achieve this, the service model of the routers can be modified. FIFO queuing can be replaced by packet forwarding strategies that discriminate different “flows” of traffic. The idea of a flow is very general. A flow might consist of “all marketing site web traffic”, or “all fileserver traffic to and from teller machines”. On the other hand, a flow might consist of a particular sequence of packets from an application in a particular machine to a peer application in another particular machine set up on request, or it might consist of all packets marked with a particular Type-of-Service bit.

EX1005 at 4-5

There is really a spectrum of possibilities for non-best-effort service something like that shown in figure 2. This spectrum is intended to illustrate that between best-effort, and hard per-flow guarantees lie many

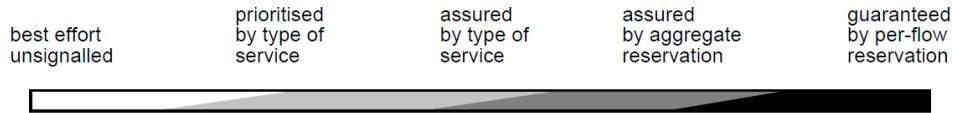


Figure 2: Spectrum of internet service types

possibilities for non-best-effort service, including having hard guarantees based on an aggregate reservation, assurances that traffic marked with a particular type-of-service bit will not be dropped so long as it remains in profile, and simpler prioritisation-based services.

Towards the right hand side of the spectrum, flows are typically identifiable in the Internet by the tuple: {source machine, destination machine, source port, destination port, protocol} any of which could be “ANY” (wildcarded).

In the multicast case, the destination is the group, and can be used to provide efficient aggregation.

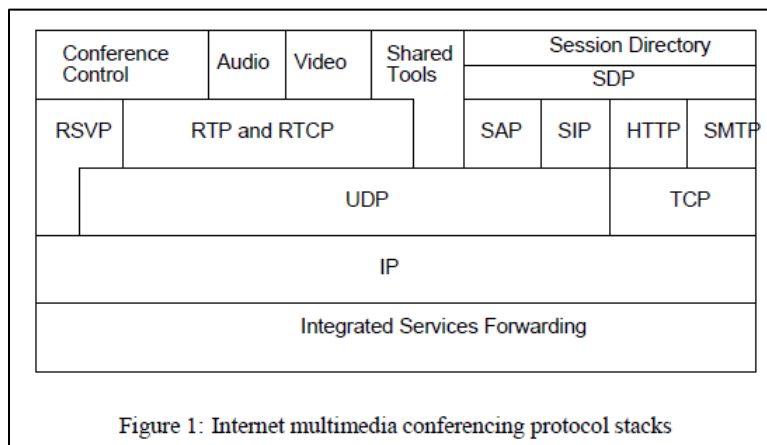
Flow identification is called classification and a class (which can contain one or more flows) has an associated service model applied. This can default to best effort.

Through network management, we can imagine establishing classes of long lived flows — enterprise networks (“Intranets”) often enforce traffic policies that distinguish priorities which can be used to discriminate in favor of more important traffic in the event of overload (though in an underloaded network, the effect of such policies will be invisible, and may incur no load/work in routers).

The router service model to provide such classes with different treatment can be as simple as a priority queuing system, or it can be more elaborate.

Although best-effort services can support real-time traffic, classifying real-time traffic separately from non-real-time traffic and giving real-time traffic priority treatment ensures that real-time traffic sees minimum delays. Non-real-time TCP traffic tends to be elastic in its bandwidth requirements, and will then tend to fill any remaining bandwidth.

EX1005 at 5



EX1005 at 3

The level of indirection introduced by the IP class D address denominating the group solves the distributed systems binding problem, by pushing this task down into routing; given a multicast address (and UDP port), a host can send a message to the members of a group without needing to discover who they are. Similarly receivers can “tune in” to multicast data sources without needing to bother the data source itself with any form of request.

EX1005 at 3

5.2 RTP

The transport protocol for real-time flows is RTP[22]. This provides a standard format packet header which gives media specific timestamp data, as well as payload format information and sequence numbering amongst other things. RTP is normally carried using UDP. It does not provide or require any connection setup, nor does it provide any enhanced reliability over UDP. For RTP to provide a useful media flow, there must be sufficient capacity in the relevant traffic class to accommodate the traffic. How this capacity is ensured is independent of RTP.

EX1005 at 8

3. Handly carries RTP traffic over UDP for “real-time” delivery.

71. Handley states that “[i]n a datagram network, multimedia information must be transmitted in packets, some of which may be delayed more than others,” and that “[i]n order that audio and video streams be played out at the recipient in the correct timing, information must be transmitted that allows the recipient to reconstitute the timing,” for which “[a] *transport protocol* with the specific functions needed for this has been defined.” EX1005 at 2.

In a datagram network, multimedia information must be transmitted in packets, some of which may be delayed more than others. In order that audio and video streams be played out at the recipient in the correct timing, information must be transmitted that allows the recipient to reconstitute the timing. A *transport protocol* with the specific functions needed for this has been defined (see section 5). The nature of the Internet reflects that of the world in that it is very heterogeneous. Techniques exist to exploit this, and to deliver appropriate quality to different participants in the same conference according to their capabilities.

EX1005 at 2

72. Handley states that “[s]o-called real-time delivery of traffic requires little in the way of transport protocol,” and that “[t]he transport protocol for real-time flows is RTP,” citing RFC 1889. EX1005 at 8. Handly states that “RTP is normally carried using UDP” and does not “provide any enhanced reliability over UDP.” EX1005 at 8. A person of ordinary skill in the art would have understood Handley as teaching the use of standard RTP for providing “real-time” delivery over UDP.

5 Transport Protocols

So-called real-time delivery of traffic requires little in the way of transport protocol. In particular, real-time traffic that is sent over more than trivial distances is not retransmittable.

With packet multimedia data there is no need for the different media comprising a conference to be carried in the same packets. In fact it simplifies receivers if different media streams are carried in separate flows (i.e., separate transport ports and/or separate multicast groups). This also allows the different media to be given different quality of service. For example, under congestion, a router might preferentially drop video packets over audio packets. In addition, some sites may not wish to receive all the media flows. For example, a site with a slow access link may be able to participate in a conference using only audio and a whiteboard whereas other sites in the same conference with more capacity may also send and receive video. This can be done because the video can be sent to a different multicast group than the audio and whiteboard. This is first step towards coping with heterogeneity by allowing the receivers to decide how much traffic to receive, and hence allowing a conference to scale more gracefully.

EX1005 at 7

5.2 RTP

The transport protocol for real-time flows is RTP[22]. This provides a standard format packet header which gives media specific timestamp data, as well as payload format information and sequence numbering amongst other things. RTP is normally carried using UDP. It does not provide or require any connection setup, nor does it provide any enhanced reliability over UDP. For RTP to provide a useful media flow, there must be sufficient capacity in the relevant traffic class to accommodate the traffic. How this capacity is ensured is independent of RTP.

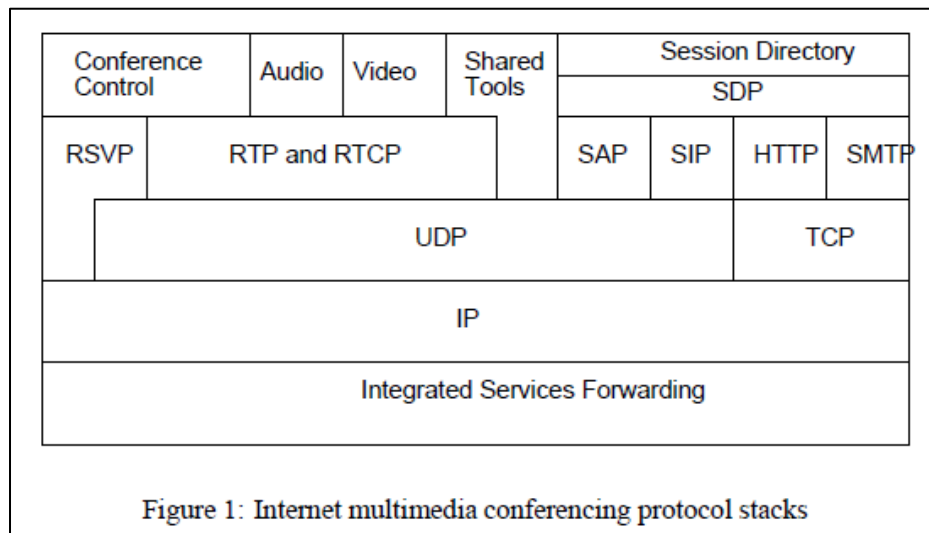
Every original RTP source is identified by a source identifier, and this source id is carried in every packet. RTP allows flows from several sources to be mixed in gateways to provide a single resulting flow. When this happens, each mixed packet contains the source ids of all the contributing sources.

RTP media timestamp units are flow specific — they are in units that are appropriate to the media flow. For example, 8kHz sampled PCM encoded audio has a timestamp clock rate of 8kHz. This means that

inter-flow synchronisation is not possible from the RTP timestamps alone.

Each RTP flow is supplemented by Real-Time Control Protocol (RTCP) packets. There are a number of different RTCP packet types. RTCP packets provide the relationship between the realtime clock at a sender and the RTP media timestamps so that inter-flow synchronisation can be performed, and they provide textual information to identify a sender in a conference from the source id.

EX1005 at 8-9



EX1005 at 3

4. Handley relies on RTP for packet encryption.

73. Handley states that “[i]f the application requirement (conference policy) is to communicate between some defined set of users, then strict privacy can only be enforced in any case through adequate end-to-end encryption.” EX1005 at 12. Handley states that “RTP specifies a standard way to encrypt RTP and RTCP packets using private key encryption schemes such as DES,” and “specifies a standard mechanism to manipulate plain text keys using MD5 so that the resulting bit string can be used as a DES key.” EX1005 at 12.

8 Security

There is a temptation to believe that multicast is inherently less private than unicast communication since the traffic visits so many more places in the network. In fact, this is not the case except with broadcast and prune type multicast routing protocols[3]. However, IP multicast does make it simple for a host to anonymously join a multicast group and receive traffic destined to that group without the other senders' and receivers'

knowledge. If the application requirement (conference policy) is to communicate between some defined set of users, then strict privacy can only be enforced in any case through adequate end-to-end encryption.

RTP specifies a standard way to encrypt RTP and RTCP packets using private key encryption schemes such as DES[18]. It also specifies a standard mechanism to manipulate plain text keys using MD5[19] so that the resulting bit string can be used as a DES key. This allows simple out-of-band mechanisms such as privacy-enhanced mail to be used for encryption key exchange.

EX1005 at 12

B. Basturk (EX1006)

1. Basturk teaches a transport layer implementation of a transport protocol that extends standard transport level reliability features to adversarial settings.

74. Basturk’s focus is on solving a problem specifically of *transport level* security—in explicit contrast to application level security—and suggests solving the transport level security problem with a secure *transport level* protocol. For example, the title of Basturk is “Secure Transport Protocols for High-Speed Networks.” EX1006 at 1. Basturk’s abstract also clearly states the security concern of the paper is “secure delivery,” which Basturk suggests is “best addressed by the design of secure transport protocols.” EX1006 at 1. The abstract goes on to say, “Our work reflects the experience gained in designing and implementing a particular secure transport protocol” (which is repeated in the “Introduction” section). Basturk states that “secure delivery” is “most naturally identified with the transport level.” EX1006 at 3. Basturk further states that it “present[s] a general method to transform a given transport protocol into a secure one.” EX1006 at 3. Basturk’s statements clearly indicate to a POSITA that the authors were aware of the different protocol layers—especially the difference between the application and transport layers—and that the “transport protocol” in Basturk is a protocol of the transport layer, not the application layer. EX1006 at 3.

Abstract

We argue that the increasingly widespread use of networks is leading to new security concerns such as *secure delivery* which are most naturally associated with the transport level and best addressed by the design of *secure transport protocols*. We present general methods for such a design which are directed at optimizing performance on a *high speed* network. Our work reflects the experience gained in designing and implementing a particular secure transport protocol.

EX1006 at 1

1 Introduction

As we entrust more and more of our communication to networks, the concern with security increases. This paper begins by arguing that the *nature* of the security guarantees which users will need and expect is changing. In particular, new requirements, like *secure delivery*, are emerging. In contrast to traditional security features which tend to be purely application level, the new ones are most naturally identified with the transport level. We suggest the design of *secure transport protocols* as the most effective way of filling all needs, and present a general method to transform a given transport protocol into a secure one. The design puts particular emphasis on maintaining the level of performance of the original transport protocol, particularly in a high speed environment. Our work reflects the experience gained in the design and implementation of a specific secure high-speed transport protocol.

EX1006 at 3

75. Basturk makes clear that the security problem it addresses and the solution that it proposes are specifically directed to the transport layer rather than the application layer. Citing “the need to add security functionality in a way which is consistent with, and best matches, the existing communication structure,” Basturk expressly “distinguish[es] security features according to the layer of the communication structure to which they are most naturally associated,” stating that “the traditional features are mostly application level, while the new ones we deem necessary are transport level.” EX1006 at 3. Basturk also clearly states that “[w]e

VL Collective Exhibit 2004

IPR2023-00923

Meta Platforms, Inc. v. VL Collective IP LLC

Page 58 of 136

found that the traditional [security] features are mostly application level, while the new ones we deem necessary are transport level.” EX1006 at 3. These statements further indicate to a POSITA that Basturk’s proposed protocol is a transport level protocol, not an application level protocol.

1.1 Security requirements and their placement

Two parties, denoted A and B , communicate over a network. We will naturally view the parties as applications who use a common transport protocol. An adversary E monitors the parties’ communications, and is even capable of injecting or removing messages from the lines. Nonetheless, the parties wish to be able to communicate “securely.” Their advantage over E is to possess a random, shared secret key a (eg. a DES key) which E doesn’t know.^{1,2}

Our discussion of security features which follows takes the perspective of a designer and architect of a secure communications medium. It reflects the need to add security functionality in a way which is consistent with, and best matches, the existing communication structure. To this end, we distinguish security features according to the layer of the communication structure to which they are most naturally associated. We found that the traditional features are mostly application level, while the new ones we deem necessary are transport level.

EX1006 at 3

76. Basturk clearly distinguishes its transport level protocol from an application level protocol. Basturk first addresses “Application level security features,” which it says are “provided by *application level security*,” where “the sender applies appropriate cryptographic functions to his data *before* passing it to the transport for conveyance.” EX1005 at 3. Basturk gives examples, stating that “Integrity is usually guaranteed by *signing* the data before passing it to the transport level.” EX1005 at 3-4. A POSITA would have understood this to be consistent with the traditional layer-respecting approach of the OSI model.

APPLICATION LEVEL SECURITY FEATURES: INTEGRITY AND CONFIDENTIALITY. *Integrity* and *confidentiality* are the two traditional, and important, security features. The first means that it isn't possible for *E* to modify the contents of transmitted data. The second means that *E* cannot figure out the content of the transmission but the receiver can recover it. Note they are requirements on the *data* transmitted by the application. This is the reason we view them as application level security features, a view which is reflected in the traditional method of providing these features, which we discuss next.

EX1006 at 3

APPLICATION LEVEL SECURITY. Application level security features are easily (and usually) provided by *application level security*. Here the sender applies appropriate cryptographic functions to his data *before* passing it to the transport for conveyance. This is the approach of Kerberos [5] and KryptoKnight [3]. Integrity is usually guaranteed by *signing* the data before passing it to the

transport level. Confidentiality is usually guaranteed by having *A encrypt* the data before passing it to the transport level, and having *B decrypt* the ciphertext which is received by his transport module and passed up to him.

EX1006 at 3-4

77. Basturk then goes on to state that “transport level security” concerns “secure delivery,” which “roughly speaking, is a guarantee to the sender *A* that his message was received by *B* and not intercepted and destroyed by the adversary.” EX1006 at 4. A POSITA would have understood secure delivery to be a transport level security issue. Basturk further states that secure delivery “*is essentially an extension of the standard transport level reliability feature to the more adversarial setting we are discussing, which is why it seems most natural to view it as a transport level feature.*” EX1006 at 4 (emphasis added). Not only does that indicate to a POSITA that Basturk’s protocol is clearly a transport level protocol, but also that

Basturk relies on “standard transport level reliability features, which indicates to a POSITA that Basturk’s protocol is for use with standard reliability protocols such as TCP as opposed to unreliable protocols like UDP.

TRANSPORT LEVEL SECURITY FEATURES: SECURE DELIVERY. Secure delivery, roughly speaking, is a guarantee to the sender *A* that his message was received by *B* and not intercepted and destroyed by the adversary. It is essentially an extension of the standard transport level *reliability* feature to the more adversarial setting we are discussing, which is why it seems most natural to view it as a transport level feature.

EX1006 at 4

2. Basturk is a transport level protocol, not an application level protocol.

78. Basturk’s description of its protocol clearly demonstrates to a POSITA that it is a transport level protocol, not an application level protocol. According to Basturk’s design, the application interacts with the transport layer through an “interface,” which indicates to a POSITA a demarcation between the transport level and application level. EX1006 at 8.

79. Consistent with that demarcation, Basturk states that “[i]n this design, the application passes its key *a* to the transport module, and leaves to the latter the task of applying any cryptographic functions or taking other security steps.” EX1006 at 4-5 (emphasis added). That indicates to a POSITA that **the transport layer is responsible for all cryptographic computations and exchange of cryptographic information.** Clearly, the application layer does none of this.

Basturk even reiterates that the task of “*applying any cryptographic functions or taking other security steps is performed by the transport module,*” which is part of the transport layer performing a secure transport level protocol. EX1006 at 4-5. Moreover, a POSITA would have understood Basturk’s protocol is a transport layer protocol because the transport module, in addition to having a clear demarcation with the application, is also used to “sign the acknowledgements messages which exists only at the transport level.” EX1006 at 4-5. At the transport layer, a reliable delivery requires recipients to send PDU acknowledgements so the sender can determine if retransmission is necessary.

2.2 The user interface

STARTUP INFORMATION. Each potential user (application) A of the secure transport protocol has a table T_A whose entries are pairs: $T_A = (A_1, a_1), \dots, (A_N, a_N)$ for some N . Each A_i is the identity of another user, and a_i is the key shared by A and A_i (but not known to any other user or the adversary). Henceforth this key will be referred to as the *long lived* key (LL-key).

CONNECTION INITIATOR INTERFACE. Now suppose A wishes to initiate a connection with B , with whom it shares LL-key a . It interfaces with a local piece of software to which it passes the identity B and the key a . It can also control the nature and strength of the security to be provided by specifying some options which will be discussed below. This software creates for her an entity we call her *secure transport machine* and denote by M_A . Once initialized, A can send (and receive) data via the secure transport machine just as it would via the original (insecure) version. That is, when it wants to send a message m , it passes it to the secure transport machine, and the latter is responsible for relaying the message, applying any necessary cryptographic functions to it, taking care of retransmission, and in general providing all transport facilities.

RESPONDER INTERFACE. At the receiving end, party B has also initialized a secure transport machine M_B . This machine has access to the table T_B of identities and associated keys. An incoming packet identifies the identity of the (claimed) sender, and the receiving machine then uses the table to get the associated key. With this key it can now verify signatures or perform any other necessary cryptography.

SUMMARY. In summary, we view A as having a secure machine M_A with which it interfaces. Similarly, B interfaces with its secure machine M_B . The cross network communication is performed by the two machines, who send each other packets across the network and are responsible for the secure transport mechanism.

EX1006 at 8

Actually secure delivery is only one example of a transport level security feature that is desirable. In Appendix A we will discuss things more formally and point to a mathematical notion of security which we believe captures the actual requirement, and includes secure delivery, integrity (and, optionally, confidentiality) as special cases. Roughly, we will ask that either the flows between the parties maintain, in the presence of E , the “relative ordering” as well as content that they would have if E were not present, or the parties detect the presence of E .

TRANSPORT LEVEL SECURITY. In this design, the application passes its key a to the transport module, and leaves to the latter the task of applying any cryptographic functions or taking other security steps. The advantage is that the transport module can compute a signature which covers

not only the data but also (relevant parts of) the transport header which it creates, particularly the byte sequence numbers. It can also sign the acknowledgments messages which exist only at the transport level. As we will see, these possibilities are key to the provision of secure delivery.

1.2 The design of efficient and secure transport protocols

We specify a method to transform any transport protocol into a “secure” version, equipped not only with the new security features discussed above, but providing, at least as options, the standard application level security features. Below we discuss the elements of the solution followed by some of its features.

3. The cryptographic functions of Basturk’s protocol are performed as a transport level protocol on a transport layer.

80. Basturk teaches that the elements of its solution are included in a secure transport protocol having two phases: “initialization” and “steady state.” EX1006 at 5, 9-10. During initialization, “an entity authentication and key exchange protocol is executed,” which “authenticates the parties and yields them a session key α .” EX1006 at 5, 9. In the steady state, “both data integrity and secure delivery are provided by attaching to each packet a signature, computed under key α ,” “acknowledgments ... are also signed,” and “the data is encrypted, also under α .” EX1006 at 5, 9-10.

ELEMENTS OF THE SOLUTION. The design will be specified in full in the body of the paper, but here is a brief sketch. The secure transport protocol has two phases, the *initialization* phase and the *steady state*. In the former, an *entity authentication and key exchange* protocol is executed. This authenticates the parties and yields them a *session key* α . In the steady state, if the integrity option is selected, both data integrity and secure delivery are provided by attaching to each packet a signature, computed under key α , and taken over some portions of the header and the data payload. Note that acknowledgments, which have no data, are also signed. If confidentiality is desired, the data is encrypted, also under α . See the rest of the paper for more details on which parts of the header must be signed and which left in the clear; why the design guarantees secure delivery; and also for explanations on why a session key is needed when the parties already share the long lived key a .

EX1006 at 5

2.3 Overview of design

We divide the design into the following components:

- (1) *Initialization: Authentication and session key exchange.* When A initializes M_A (by passing it B 's identity and the shared key a) the machine initiates and executes an authentication and key exchange protocol with M_B . Specifically, the protocol used is the one chosen by the initiator in the options specified above. This protocol results in the distribution of a new (fresh) *session key* α while making sure that this key is authentic (ie. known only to M_A and M_B).
- (2) *Steady state:* Initialization completed, M_A is ready to receive data from the application A and send it to B ; similarly, M_B is ready to receive data from M_A and pass it to B .
 - (2.1) *Integrity via signing.* If the integrity option is selected then data is *signed* by M_A under the session key α (a signature is also called a MAC, or message authentication code). The signature covers not only the data payload but also the transport header (as explained later). On receiving a signed packet, M_B uses α to verify the signature.

- (2.2) *Confidentiality via encryption.* If the confidentiality option is selected then the data payload is encrypted by M_A under the session key α . On receiving a packet, M_B uses α to decrypt the payload and then passes the plaintext to B .

EX1006 at 9-10

81. A person of ordinary skill in the art would have understood that the cryptographic functions associated with Basturk's secure transport protocol

(including for providing secure delivery, authentication, and encryption) are performed at the transport layer (by transport machines) as part of the transport protocol—not by the application as part of an application protocol or at the application layer. Basturk teaches that two applications, denoted A and B , that wish to communicate with each other will share a “long lived key” (or “LL-key”) denoted “ a .” EX1006 at 3, 8. Each application will also have software that generates for it a “secure transport machine” denoted “ M_A ” (for application A) and M_B (for application B). EX1006 at 8. According to Basturk’s protocol, “the application passes its key a to the transport module, and leaves to the latter the task of applying any cryptographic functions or taking other security steps.” EX1006 at 4. That is, “when [an application] wants to send a message m , it passes it to the secure transport machine, and the latter is responsible for relaying the message, applying any necessary cryptographic functions to it, taking care of retransmission, and in general providing all transport facilities.” EX1006 at 8. Then, “[t]he cross network communication is performed by the two machines, who send each other packets across the network and are responsible for the secure transport mechanism.” EX1008 at 8.

1.1 Security requirements and their placement

Two parties, denoted A and B , communicate over a network. We will naturally view the parties as applications who use a common transport protocol. An adversary E monitors the parties' communications, and is even capable of injecting or removing messages from the lines. Nonetheless, the parties wish to be able to communicate "securely." Their advantage over E is to possess a random, shared secret key a (eg. a DES key) which E doesn't know.^{1,2}

¹ How they come into possession of such a key is not our concern here. Rather, the problem is how to use the key to communicate securely.

² For simplicity we stick to this symmetric case. Our methods can be modified to work in the public key scenario as well.

EX1006 at 3

2.2 The user interface

STARTUP INFORMATION. Each potential user (application) A of the secure transport protocol has a table T_A whose entries are pairs: $T_A = (A_1, a_1), \dots, (A_N, a_N)$ for some N . Each A_i is the identity of another user, and a_i is the key shared by A and A_i (but not known to any other user or the adversary). Henceforth this key will be referred to as the *long lived* key (LL-key).

CONNECTION INITIATOR INTERFACE. Now suppose A wishes to initiate a connection with B , with whom it shares LL-key a . It interfaces with a local piece of software to which it passes the identity B and the key a . It can also control the nature and strength of the security to be provided by specifying some options which will be discussed below. This software creates for her an entity we call her *secure transport machine* and denote by M_A . Once initialized, A can send (and receive) data via the secure transport machine just as it would via the original (insecure) version. That is, when it wants to send a message m , it passes it to the secure transport machine, and the latter is responsible for relaying the message, applying any necessary cryptographic functions to it, taking care of retransmission, and in general providing all transport facilities.

RESPONDER INTERFACE. At the receiving end, party B has also initialized a secure transport machine M_B . This machine has access to the table T_B of identities and associated keys. An incoming packet identifies the identity of the (claimed) sender, and the receiving machine then uses the table to get the associated key. With this key it can now verify signatures or perform any other necessary cryptography.

SUMMARY. In summary, we view A as having a secure machine M_A with which it interfaces. Similarly, B interfaces with its secure machine M_B . The cross network communication is performed by the two machines, who send each other packets across the network and are responsible for the secure transport mechanism.

EX1006 at 8

82. A person of ordinary skill in the art would have understood that the session key α for performing cryptographic functions in Basturk's protocol is generated and exchanged at the transport layer—not at the application layer. EX1006 at 9-10. Basturk explains that when application A passes B 's identity and the shared key a to transport machine M_A at the transport layer, “the machine initiates and executes an authentication and key exchange protocol with M_B ,” which “results in the distribution of a new (fresh) session key α while making sure that this key is authentic (ie. known only to M_A and M_B).” EX1006 at 9. That is, “[t]he secure session begins with an authentication and key exchange protocol which yields to the parties a session key α to be used (to sign and encrypt) in the rest of the session.” EX1006 at 10. Basturk clearly indicates that the session key α is not known at the application layer—Basturk expressly states that applications “ A and B never see it,” and that it is “known only to M_A and M_B ,” which are part of the transport layer. EX1006 at 10.

2.3 Overview of design

We divide the design into the following components:

- (1) *Initialization: Authentication and session key exchange.* When A initializes M_A (by passing it B 's identity and the shared key a) the machine initiates and executes an authentication and key exchange protocol with M_B . Specifically, the protocol used is the one chosen by the initiator in the options specified above. This protocol results in the distribution of a new (fresh) *session key* α while making sure that this key is authentic (ie. known only to M_A and M_B).
- (2) *Steady state:* Initialization completed, M_A is ready to receive data from the application A and send it to B ; similarly, M_B is ready to receive data from M_A and pass it to B .
 - (2.1) *Integrity via signing.* If the integrity option is selected then data is *signed* by M_A under the session key α (a signature is also called a MAC, or message authentication code). The signature covers not only the data payload but also the transport header (as explained later). On receiving a signed packet, M_B uses α to verify the signature.

- (2.2) *Confidentiality via encryption.* If the confidentiality option is selected then the data payload is encrypted by M_A under the session key α . On receiving a packet, M_B uses α to decrypt the payload and then passes the plaintext to B .

EX1006 at 9-10

2.4 Authentication and session key exchange

As discussed above, the secure session begins with an authentication and key exchange protocol which yields to the parties a session key α to be used (to sign and encrypt) in the rest of the session. The first thing that needs to be explained is: Since the parties already share a key (namely the LL-key a) why get another one? The use of a session key has several motivations. First, it helps avoid cross session attacks in which data from one session are replayed in another. Second, it minimizes exposure of the long lived key. It is a standard technique in cryptographic practice.

Notice that only the transport machines M_A and M_B know about the session key. A and B never see it.

EX1006 at 10

2.2 The user interface

STARTUP INFORMATION. Each potential user (application) A of the secure transport protocol has a table T_A whose entries are pairs: $T_A = (A_1, a_1), \dots, (A_N, a_N)$ for some N . Each A_i is the identity of another user, and a_i is the key shared by A and A_i (but not known to any other user or the adversary). Henceforth this key will be referred to as the *long lived* key (LL-key).

CONNECTION INITIATOR INTERFACE. Now suppose A wishes to initiate a connection with B , with whom it shares LL-key a . It interfaces with a local piece of software to which it passes the identity B and the key a . It can also control the nature and strength of the security to be provided by specifying some options which will be discussed below. This software creates for her an entity we call her *secure transport machine* and denote by M_A . Once initialized, A can send (and receive) data via the secure transport machine just as it would via the original (insecure) version. That is, when it wants to send a message m , it passes it to the secure transport machine, and the latter is responsible for relaying the message, applying any necessary cryptographic functions to it, taking care of retransmission, and in general providing all transport facilities.

RESPONDER INTERFACE. At the receiving end, party B has also initialized a secure transport machine M_B . This machine has access to the table T_B of identities and associated keys. An incoming packet identifies the identity of the (claimed) sender, and the receiving machine then uses the table to get the associated key. With this key it can now verify signatures or perform any other necessary cryptography.

SUMMARY. In summary, we view A as having a secure machine M_A with which it interfaces. Similarly, B interfaces with its secure machine M_B . The cross network communication is performed by the two machines, who send each other packets across the network and are responsible for the secure transport mechanism.

EX1006 at 8

2.4 Authentication and session key exchange

As discussed above, the secure session begins with an authentication and key exchange protocol which yields to the parties a session key α to be used (to sign and encrypt) in the rest of the session. The first thing that needs to be explained is: Since the parties already share a key (namely the LL-key a) why get another one? The use of a session key has several motivations. First, it helps avoid cross session attacks in which data from one session are replayed in another. Second, it minimizes exposure of the long lived key. It is a standard technique in cryptographic practice.

Notice that only the transport machines M_A and M_B know about the session key. A and B never see it.

EX1006 at 10

83. A person of ordinary skill in the art would have also understood that Basturk's signature (MAC_α) is inserted into a transport layer packet as part of a transport layer protocol. Basturk's protocol computes the signature (*i.e.*, MAC_α) under the session key α (which is known only at the transport layer). EX1006 at 9-10 ("If the integrity option is selected then data is signed by M_A under the session key α (a signature is also called a MAC, or message authentication code.>"). Indeed, Basturk's MAC is denoted as MAC_α , which is "a message authentication code under α ," and is computed over a portion $\bar{p}$ of a transport layer packet p (not an application layer packet). EX1006 at 11-12.

2.3 Overview of design

We divide the design into the following components:

- (1) *Initialization: Authentication and session key exchange.* When A initializes M_A (by passing it B 's identity and the shared key a) the machine initiates and executes an authentication and key exchange protocol with M_B . Specifically, the protocol used is the one chosen by the initiator in the options specified above. This protocol results in the distribution of a new (fresh) *session key* α while making sure that this key is authentic (*ie.* known only to M_A and M_B).
- (2) *Steady state:* Initialization completed, M_A is ready to receive data from the application A and send it to B ; similarly, M_B is ready to receive data from M_A and pass it to B .

- (2.1) *Integrity via signing.* If the integrity option is selected then data is *signed* by M_A under the session key α (a signature is also called a MAC, or message authentication code). The signature covers not only the data payload but also the transport header (as explained later). On receiving a signed packet, M_B uses α to verify the signature.

- (2.2) *Confidentiality via encryption.* If the confidentiality option is selected then the data payload is encrypted by M_A under the session key α . On receiving a packet, M_B uses α to decrypt the payload and then passes the plaintext to B .

EX1006 at 9-10

VL Collective Exhibit 2004
IPR2023-00923

Meta Platforms, Inc. v. VL Collective IP LLC
Page 71 of 136

2.5 Integrity and signatures

Signatures are used in the AKE protocols and also, if the integrity option is selected, in the steady state. In the latter, their use will guarantee a broad set of properties including protection of the data against modification, secure delivery, and protection against replay attacks. Let's begin by describing the mechanisms and see why they yield the above mentioned properties in the steady state. Then we will look at design choices which are motivated by performance issues and which apply to the use of signatures in both the AKE protocols and the steady state.

We denote by MAC_α a message authentication code under α . It has, roughly speaking, the

following property. If an adversary sees a sequence $(m_1, \text{MAC}_\alpha(m_1)), \dots, (m_k, \text{MAC}_\alpha(m_k))$ of messages together with their MACs (or signatures), then it is infeasible for this adversary to produce a pair of the form $(m, \text{MAC}_\alpha(m))$ in which $m \notin \{m_1, \dots, m_k\}$.

In the steady state (integrity option assumed selected) the sender attaches to each packet p the value $\text{MAC}_\alpha(\bar{p})$ where $\bar{p}$ is the subset of p which includes the transport header, the full data payload, and the trailer of the packet p . We will see that of particular importance is the fact that the transport header includes the sequence number of the packet which is thereby covered by the MAC. We assume that all packets in each direction, even those which are purely transport level acknowledgements (ACKs), can be differentiated. A simple solution is to include in each packet a large enough (to avoid wrap-around) *signature* counter. We also assume that the transport header contains information which identifies the sender so that the identity of the sender is covered by the MAC as well. On receiving p, τ the receiver checks that $\tau = \text{MAC}_\alpha(\bar{p})$ and rejects the packet unless this is true.

EX1006 at 11-12

4. Basturk's "on-the-fly" signature computation is not enabled by the use of a trailer.

84. Basturk states that "[t]he provision of security is bound to somewhat degrade performance," EX1006 at 6, including as a result of "the additional processing required by security functions," EX1006 at 7. To purportedly minimize performance degradation caused by adding transport level security, Basturk suggests performing the transport level security computations (*i.e.*, "signature generation and verification or encryption/decryption") in a manner that is called "on-the-fly."

EX1006 at 7-8. One of Basturk’s requirements for enabling “on-the-fly” computation is that “[s]ignature computation starts before the retrieval of connection context” for messages received by a receiver. EX1006 at 8.

The provision of security is bound to somewhat degrade performance, but the goal is to keep this impact at a minimum. A first generic step towards minimizing the perceived performance impact of security is to provide a range of security “options” to users. Each option corresponds to a different trade-off between the level of security it affords and its cost in terms of performance. For example, users with different requirements may want to choose between encryption algorithms of different complexity and strength. It is, therefore, important to incorporate the flexibility of such choices when designing a secure transport protocol. Accordingly, our design provides a wide range of security options which are further detailed later.

EX1006 at 6

INCREASED PROCESSING. The other aspect of security that has the potential to degrade performance is the additional processing required by security functions. This is most significant for long connections as it applies to all steady state transmissions. There are, however, again some rules that can be followed to limit this impact.

The main one is that all computations should be performed in the most timely fashion possible. In other words, while an increase in the amount of computation is unavoidable, it is important to ensure that nothing slows down or delays this additional processing. In all instances we would like all computations to be carried out “on-the-fly” as data is being transmitted or received. This minimizes latency.

When applied to security functions such as signature generation and verification or encryption/decryption, the requirement of on-the-fly processing has several implications. For example, we would like decryption of a received packet to proceed independently of that of any other packet (note that this also allows parallelization) as packets may be received out of order.

Computation of cryptographic functions can be greatly speeded up by use of dedicated hardware. Our design does not require such hardware to be available, but adopts various design principles that would allow this hardware, if available, to be exploited in the best possible way. On-the-fly processing is important in this regard as it facilitates the eventual off-loading to the hardware.

EX1006 at 7

Both signature computation and encryption/decryption rely on the use of secret information (key) shared by the two communicating parties. Retrieval of this information is, therefore, necessary for such processing. While one can assume its immediate availability when transmitting data, it must typically be retrieved upon receipt of a packet and this context retrieval step can introduce latency. It is, therefore, useful to design security functions, in particular signature computation, which allow processing to start immediately upon arrival of data, without waiting for the associated context retrieval to complete.

To summarize, our requirement for timely and on-the-fly security computations translates into three specific design rules:

- (1) Signature computation starts before the retrieval of connection context;
- (2) Packets are independently encrypted;
- (3) Hardware utilization is facilitated wherever possible.

EX1006 at 8

85. Basturk states that “[t]he ability to achieve such on-the-fly processing depends on two major components. The first is the latency in retrieving the information needed to perform the necessary computations,” known as “context” (“such as the shared session key, the counter value of the last accepted packet, the type of cryptographic function used, etc., which are central to the verification of a packet integrity”). EX1006 at 12. Basturk teaches that this context retrieval step creates latency because signature computation relies on the use of “secret information (key) shared by the two communicating parties,” which is “necessary for such processing” and “must typically be retrieved upon receipt of a packet and this context retrieval step can introduce latency.” EX1006 at 8. Basturk suggests reducing this context retrieval latency by putting the context information early in

each packet, namely in a security segment in the packet's header. EX1006 at 12, 13, 15-16.

We now turn to design issues motivated by performance. Let us recall some of the issues identified in Section 2.1. It is essential that the steady state overhead involved in computing and checking signatures and counters be kept to a minimum. Ideally, we want signatures to be computed and verified on-the-fly; i.e., MAC generation (verification) should be carried out as data transmissions (receptions) proceeds rather than before (after) these operations have completed. The goal is to avoid slowing down the regular flow of data, and allow the possible off-load of signature computations and verifications to dedicated hardware. The ability to achieve such on-the-fly processing depends on two major components. The first is the latency in retrieving the information needed

to perform the necessary computations. The second is the nature of the computations themselves. The first component is most relevant at the receiver, while the second is equally important at both the transmitter and the receiver.

The main source of latency when receiving a new packet is the time needed to recover the associated connection context. The context contains information such as the shared session key, the counter value of the last accepted packet, the type of cryptographic function used, etc., which are central to the verification of a packet integrity. Some delay is obviously unavoidable when retrieving this information, but it should be made as small as possible and its impact on the packet verification process should be kept to a minimum. A first step in achieving this is to allow context retrieval to be performed as quickly as possible by specifying early on in the packet all the required information. While rapid context retrieval certainly helps it does not eliminate all initial latency, and it is therefore still important to allow computations to start even before the full context has been retrieved. Finally, latency can also be kept small by ensuring that computations are of minimal complexity while meeting our security requirements.

These different requirements have been incorporated in our design, where the goal of minimal latency is reflected in the following two rules.

First, all data used to generate the signature is present in the packet. Let's illustrate with an example from the AKE protocols. In such challenge response protocols, it is common that one party (say B) proves his identity by signing a challenge R_A sent by A . In such a case, rather than returning $\langle R_A, \sigma \rangle$ with $\sigma = \text{MAC}_\alpha(R_A)$, party B could return just the short tag σ ; after all, A already knows R_A (it is in A 's context) so why waste bandwidth sending it? The reason for preferring to send the full $\langle R_A, \sigma \rangle$ is that signature verification can begin without waiting for the outcome of context retrieval. On the other hand, the cost of resending R_A is minimal since bandwidth is not that expensive at these speeds. In general, this translates into the following packet format: a signed packet will be of the form $\langle D, \sigma \rangle$ where σ is a signature of packet D under key α .

Note that we cannot prevent context retrieval of R_A in the above example since A must definitely check that what is received is the signature of the right challenge (namely the challenge A just sent). But putting the information in the flow decouples the signature checking from this context retrieval.

3 Implementation Overview

In this last section we briefly outline the structure of our implementation of a secure transport protocol based on a version of an existing, high-speed optimistic transport protocol similar to XTP [6, 4]. The implementation is in C and has been developed on an RS6000 workstation. All the features introduced in the previous sections (authentication, confidentiality, integrity, choice of the level of security) have been incorporated and tested.

An application which wants to use the transport protocol has to link with the transport machine object code (see Section 2.2). The code basically consists of an interface with the application and of two threads, one for transmission and one for reception of packets. The interface requests and responses are communicated to the threads through message queues.

To add security to the non-secure version of the protocol, four parts of the protocol have to be modified: the interface, the connection resources (contexts), the state machine and the format of the packets.

The application communicates with the RTP machine through the interface by using structures called *verbs* which contain data specific to particular actions of the protocol: OPEN, CONNECT,

LISTEN, SEND, RECEIVE, DISCONNECT, CLOSE. Typically, OPEN allocates local resources, CONNECT allows the setting of the address to be called, SEND gives some buffers of data to be sent on a specific connection, etc. These verbs can be chained together and passed in one call to the RTP machine, which will then process them. For example, OPEN and the chained verbs CONNECT-SEND allows to specify a recipient and the content of a message. The transport machine will then packetize the message and send it over the network to the specified recipient. Once the message has been delivered, the application is notified. The connection can then be closed and resources freed by issuing a CLOSE verb. Verbs, besides their chaining capability, have the advantage that they can easily be extended.

The interface to this protocol has been extended while remaining compatible with the previous non-secure version. Beyond simple compatibility advantages, this also greatly eases the work of “securing” existing applications built on top of this protocol. When security is provided, the CONNECT and LISTEN verbs have dedicated fields for security options, such as a long-lived key and bits to select the cryptographic function and whether integrity and/or confidentiality must be provided.

The protocol associates to each connection a resource called its *context*, which contains all the information necessary to provide the required transport services, such as states, pending data, timers, etc. Information relative to the security options and cryptographic keys is also stored in the context which, along with the RTP state machine, has also been expanded to include states relative to the authentication process.

A transport packet consists of a header followed by data, if any. The format of the packets has been designed to be extensible. Most transport control functions are present in optional fields called segments which are part of the transport header. A security segment has been added to include information needed for authentication and confidentiality. Following data, a trailer has been added that contains the packet signature.

A library of fast implementation of cryptographic functions has been used in order to implement the encryption algorithm and the signature scheme. Currently, MD5, SHA, DES, Double DES are the functions the upper layer can choose from.

EX1006 at 15-16

Both signature computation and encryption/decryption rely on the use of secret information (key) shared by the two communicating parties. Retrieval of this information is, therefore, necessary for such processing. While one can assume its immediate availability when transmitting data, it must typically be retrieved upon receipt of a packet and this context retrieval step can introduce latency. It is, therefore, useful to design security functions, in particular signature computation, which allow processing to start immediately upon arrival of data, without waiting for the associated context retrieval to complete.

EX1006 at 8

86. The second component for achieving on-the-fly processing “is the nature of the computations themselves,” meaning “MAC computations are kept simple and can start prior to context retrieval.” EX1006 at 13 (Simplicity is achieved, according to Basturk, by using less computationally intensive standard functions. EX1006 at 13.). The ability for MAC computations to start prior to context retrieval is achieved because, although the MAC computation $MAC_{\alpha}(m)$ requires computing a function of α (*i.e.*, f_{α} , where α is the session key included in the context information being retrieved), computing the function also includes computing a hash function (“ $h(m)$ ”) that “*depend[s] neither on α nor on the cryptographic primitive chosen in the initial option.*” EX1006 at 13 (emphasis added). In other words, as a person of ordinary skill in the art would have understood, Basturk teaches using a style of MAC computation that allows part of the computation to be performed on a received message *before* retrieval of the session key α or other cryptographic information from the context information is complete (*i.e.*, a computation of the form $MAC_{\alpha}(m) = f_{\alpha}(h(m))$, which includes a hash algorithm $h(m)$ that does not require knowing the session key α or other cryptographic information). EX1006 at 13.

Second, MAC computations are kept simple and can start prior to context retrieval. In particular, we rely on MAC implementations of the form $\text{MAC}_\alpha(m) = f_\alpha(h(m))$, where h is a (hash) function depending neither on α nor on the cryptographic primitive chosen in the initial options. The use of such an implementation will enhance performance in our setting as MAC computation does not have to wait for α and the cryptographic function to be retrieved from the context; we can begin computing $h(m)$ straight away. By the time we need to compute f_α the key α and cryptographic function will have been retrieved. In addition, since h can be chosen to be a standard hash function and only f_{alpha} needs to involve more computationally intensive cryptographic functions, the overall cost of signature generation/verification can be kept low.

The authentication protocols and packet formats of our implementation guarantee minimal latency by following the above two rules.

EX1006 at 13

87. In view of the foregoing, a person of ordinary skill in the art would have understood Basturk to teach that “on-the-fly” processing is enabled, in part, by locating the context information for signature verification (*i.e.*, the session key α and cryptographic information) early in the packet, namely in the header, not by locating the signature (the value being computed with the context information) at the end of the packet in a trailer. A person of ordinary skill in the art would have understood that this placement of context information is how the reduction in context retrieval latency for “on-the-fly” processing would be achieved. A person of ordinary skill in the art would have also understood that Basturk’s placement of the packet *signature* (*i.e.*, the MAC_α) in a trailer is not what enables the retrieval of the *context information* to be performed “as quickly as possible.” EX1006 at 12. That is, a person of ordinary skill in the art would have understood Basturk to teach that the speed of context retrieval is determined by the location of required information in

the packet that is needed to perform context retrieval, not the location of the signature. In other words, placing the signature value in a trailer would not result in a reduction in context retrieval latency.

The main source of latency when receiving a new packet is the time needed to recover the associated connection context. The context contains information such as the shared session key, the counter value of the last accepted packet, the type of cryptographic function used, etc., which are central to the verification of a packet integrity. Some delay is obviously unavoidable when retrieving this information, but it should be made as small as possible and its impact on the packet verification process should be kept to a minimum. A first step in achieving this is to allow context retrieval to be performed as quickly as possible by specifying early on in the packet all the required information. While rapid context retrieval certainly helps it does not eliminate all initial latency, and it is therefore still important to allow computations to start even before the full context has been retrieved. Finally, latency can also be kept small by ensuring that computations are of minimal complexity while meeting our security requirements.

EX1006 at 12

88. Furthermore, a person of ordinary skill in the art would have understood Basturk to teach that “on-the-fly computation” is further enabled by choosing a style of signature computation (*i.e.*, $\text{MAC}_\alpha(m) = f_\alpha(h(m))$) that permits part of the computation to be performed on a received message before context is retrieved, not by placing the signature in a trailer. A person of ordinary skill in the art would have understood that the signature computation style depends on the algorithm selected, *not where* the signature value is located in the packet. In other words, placing the signature value in a trailer would not result in a style of signature computation that can be started before context is retrieved.

89. A person of ordinary skill in the art would therefore have understood that neither of the two “components” for enabling Basturk’s “on-the-fly” computation requires or is enabled by locating the signature value (MAC_a) in a trailer. A POSITA would therefore not have been motivated to use a trailer to achieve “on-the-fly” processing.

C. PacketCable (EX1007)

1. PacketCable’s MTA provides transport services.

90. According to PacketCable, “[t]he PacketCable architecture” is an “end-end broadband architecture that supports voice, video, and other multimedia services.” EX1007 at 10. With respect to the PacketCable architecture, “The requirements for these functional components and *the standardized interfaces between components* are defined in detail in the PacketCable 1.0 specifications.” EX1007 11.

91. PacketCable says that the PacketCable architecture “contains three networks: the ‘DOCSIS HFC Access Network,’ the ‘Managed IP Network’ and the PSTN.” EX1007 at 12, Figure 1 (reproduced below). Figure 12 shows an “Embedded MTA Client” displaying two distinct components, a “MTA” and a “Cable Modem” situated on the “HFC access-network (DOCSIS).” PacketCable states “An MTA is a PacketCable client device that **contains a subscriber-side**

interface to the subscriber's CPE (e.g., telephone) and a network-side signaling interface to call control elements in the network.” PacketCable also uses the term “**MTA Application**” when describing a conceptual functional architecture. EX1007 at 20. PacketCable’s use of the phrases “MTA Application”, as distinct from the “MTA” device that “contains a subscriber-side interface” is consistent with PacketCable’s motivation to support the “development and deployment of packetized voice and video *applications*.” EX1007 at 9 (“1.2 PacketCable Motivation”).

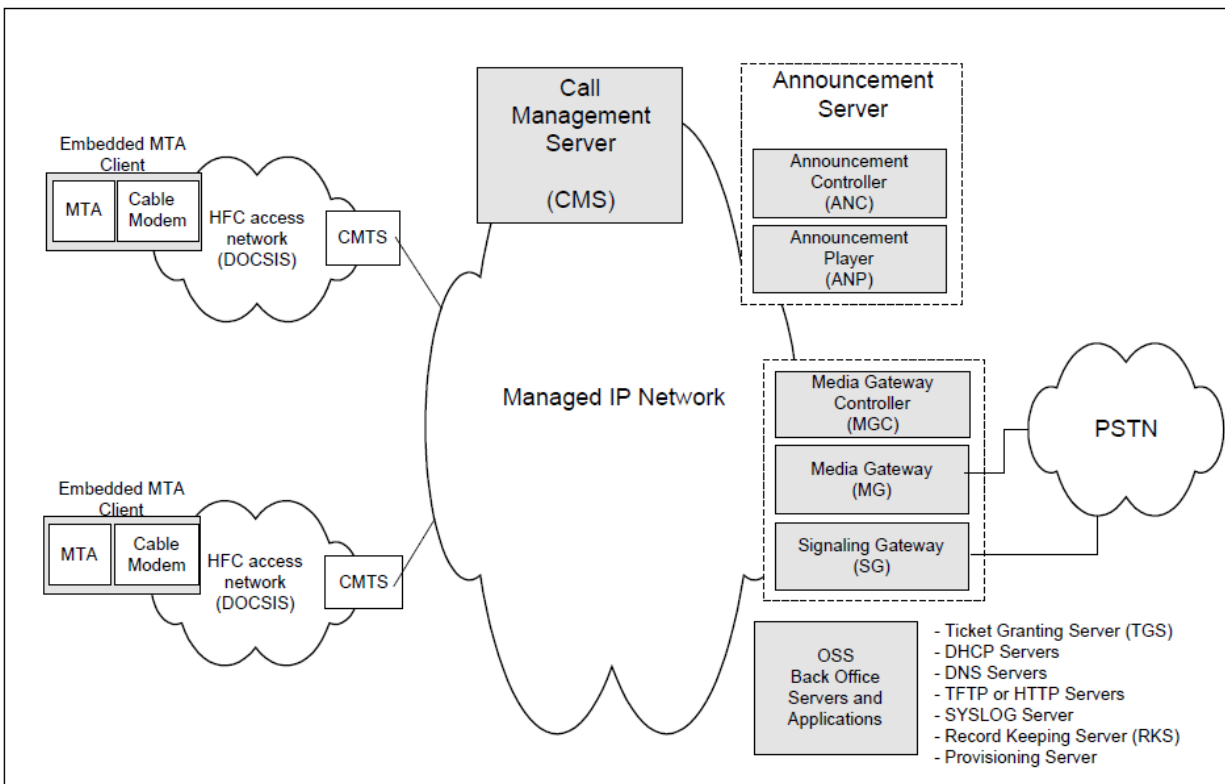


Figure 1. PacketCable Reference Architecture

EX1007 at 12

VL Collective Exhibit 2004
IPR2023-00923

Meta Platforms, Inc. v. VL Collective IP LLC
Page 82 of 136

92. According to PacketCable, “The DOCSIS HFC access network includes the following functional components: the Cable Modem (CM), Multimedia Terminal Adapter (MTA), and the Cable Modem Termination System (CMTS).” EX1007 at 12. The PacketCable 1.0 provides “specifications” that “define protocols” in the area of “Media Stream *Transport* and Encoding” and “Security and Privacy”. EX1007 at 14 (emphasis added)

2.4 PacketCable 1.0 Design Considerations

In order to enable real-time multimedia communications across the cable network infrastructure, PacketCable specifications define protocols in the following areas:

- Call Signaling
- Quality of Service
- Media Stream Transport and Encoding
- Device Provisioning
- Event Messaging
- Security and Privacy
- Operational Support Systems

EX1007 at 14

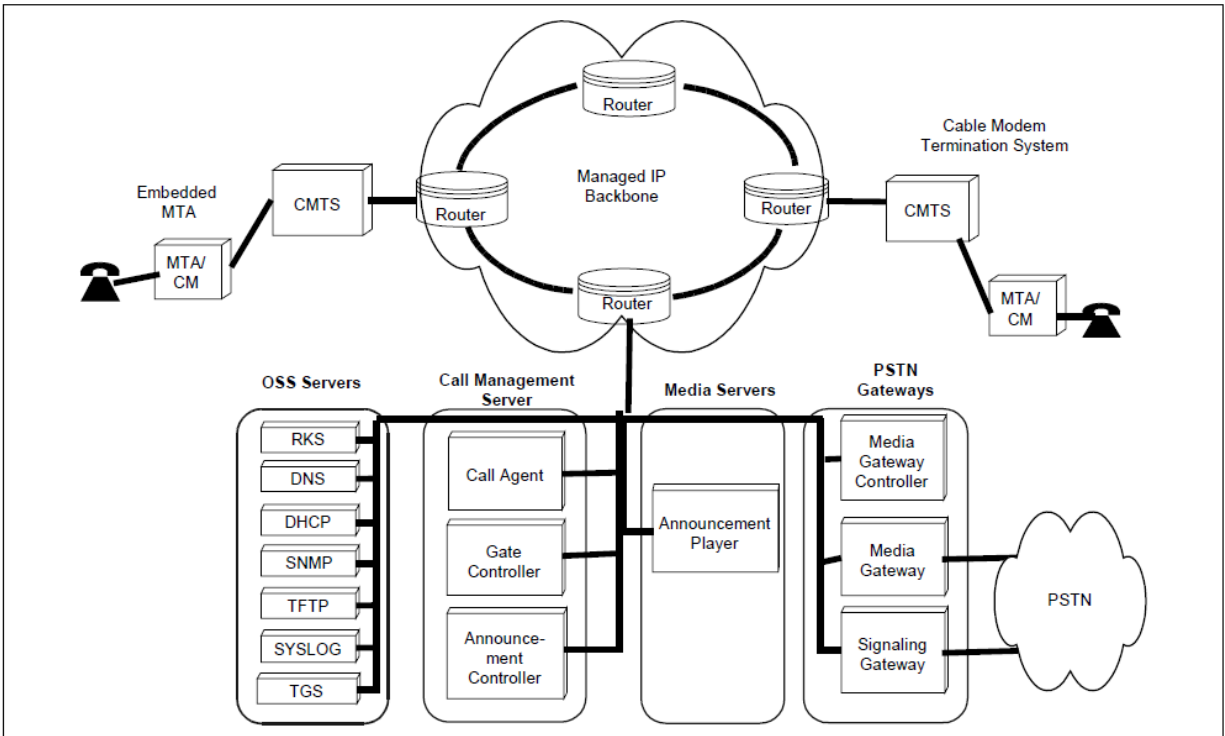


Figure 3. PacketCable Component Reference Model

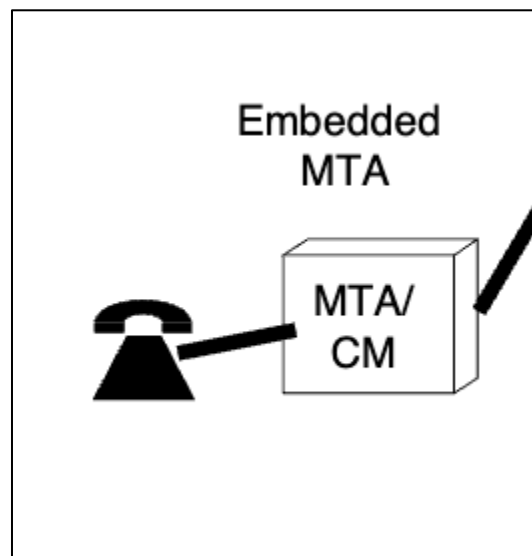
EX1007 at 18

93. According to PacketCable, “An embedded MTA (E-MTA) is a single hardware device that incorporates a DOCSIS 1.1 cable modem as well as a PacketCable MTA component.” EX1007 at 18.

94. According to PacketCable, “[a]n **MTA** is a PacketCable client device that contains a subscriber-side **interface to the subscriber’s CPE (e.g., telephone)** and a network-side signaling interface to call control elements in the network. An MTA provides codecs and **all signaling and encapsulation functions required for**

media transport and call signaling.” EX1007 at 18; *see also id.* at 64 (definition of “MTA”).

95. An enlarged portion of Figure 3 shows the “Embedded MTA” which comprises the “MTA” (i.e. Media Terminal Adapter) and “CM” (i.e. cable modem) interfacing with the subscriber’s “CPE” (customer’s premise equipment).



EX1007 at 18 (Portion of the Figure 3)

96. According to PacketCable, [i]n order to enable real-time communications across the cable network infrastructure,” PacketCable defines protocols for “*Media Stream Transport and Encoding*.” EX1007 at 14.

97. PacketCable states that one of its “General Architectural Goals” is to “Leverage existing protocol standards,” including “protocols approved by the IETF,” one of which is RTP. EX1007 at 15, 31. PacketCable states that “[t]he IETF

standard RTP (RFC 1899 - Real-Time Transport Protocol) is used to transport all media streams in the PacketCable network.” and “PacketCable uses the RTP profile for audio and video streams as defined in RFC 1890.” *Id.* 31; *see also id.* at 57-58 (“‘RTP: A Transport Protocol for Real-Time Application,’ IETF RFC 1889.”), (“‘RTP Profile for Audio and Video Conferences with Minimal Control,’ IETF RFC 1890.”).

2.4.1 General Architectural Goals

- Enable voice quality capabilities comparable to or better than the PSTN as perceived by the end-user.
- Provide a network architecture that is scalable and capable of supporting millions of subscribers.
- Ensure the one-way delay for local IP access and IP egress (i.e. excluding the IP backbone network) is less than 45ms.
- Support primary and secondary line residential voice communications capabilities.
- Leverage existing protocol standards. PacketCable strives to specify open, approved industry standards that have been widely adopted in other commercial communication networks. This includes protocols approved by the ITU, IETF, IEEE, Telcordia and other communications standards organizations.

EX1007 at 15

4.2 Media Streams

The IETF standard RTP (RFC 1899 - Real-Time Transport Protocol) is used to transport all media streams in the PacketCable network[32]. PacketCable utilizes the RTP profile for audio and video streams as defined in RFC 1990[35].

The primary media flow paths in the PacketCable network architecture are shown in Figure 6 and are further described below.

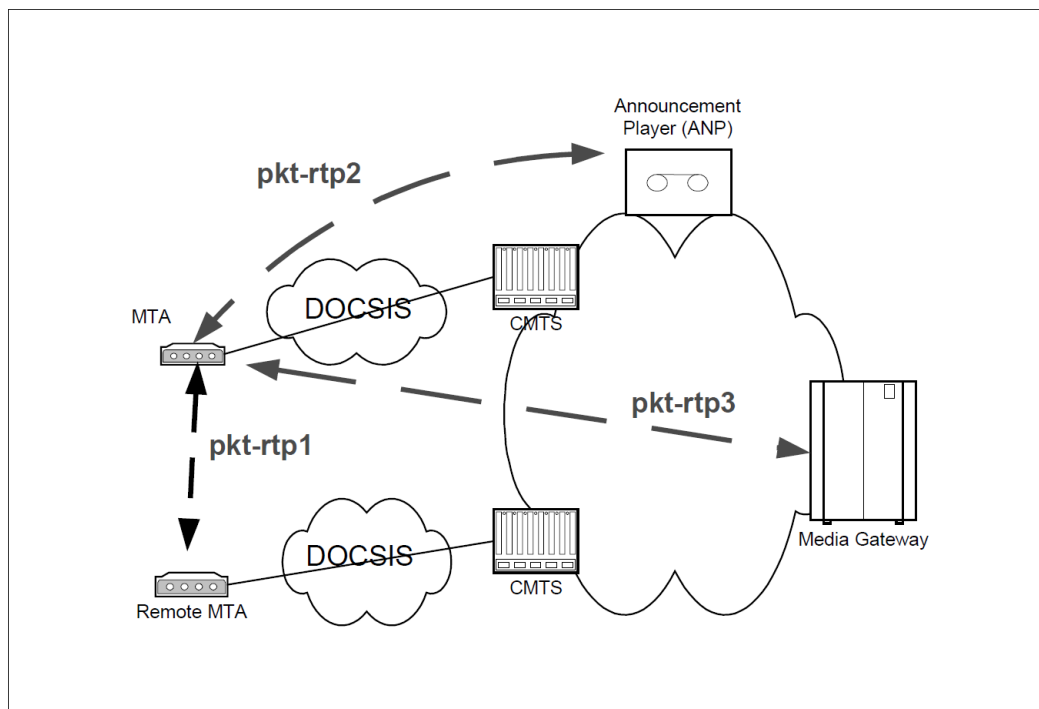


Figure 6. RTP Media Stream Flows in a PacketCable Network

EX1007 at 31

98. PacketCable relies specifically on the MTA to provide standardized services. Consistent with providing secure transport services, PacketCable states, “An MTA is responsible for ... Authentication, confidentiality and integrity of some messages between the MTA and other PacketCable network elements” EX1007 at 19.

99. The description of the “3.1.1 MTA Functional Requirements” demonstrates that the “MTA” that is “responsible for ... Authentication, confidentiality and integrity” is not the MTA Application, but rather the MTA device, which provides networking functionality such as “Authentication, confidentiality and integrity of some messages between the MTA and other PacketCable network elements.” EX1007 at 19. This indicates the absence of an MTA Application (e.g. a video application).

3.1.1 MTA Functional Requirements

An MTA is responsible for the following functionality:

- NCS call signaling with the CMS
- QoS signaling with the CMS and the CMTS
- Authentication, confidentiality and integrity of some messages between the MTA and other PacketCable network elements
- Mapping media streams to the MAC services of the DOCSIS access network
- Encoding/decoding of media streams
- Providing multiple audio indicators to phones, such as ringing tones, call-waiting tones, stutter dial tone, dial tone, etc.
- Standard PSTN analog line signaling for audio tones, voice transport, caller-id signaling, DTMF, and message waiting indicators
- The G.711 audio codec
- One or more RJ11 analog interface(s) as defined by Bellcore TR-909

EX1007 at 19.

100. Throughout PacketCable, “the term MTA refers to an embedded MTA,” which PacketCable defines with the following functional architecture that shows RTP carried over UDP. EX1007 at 20.

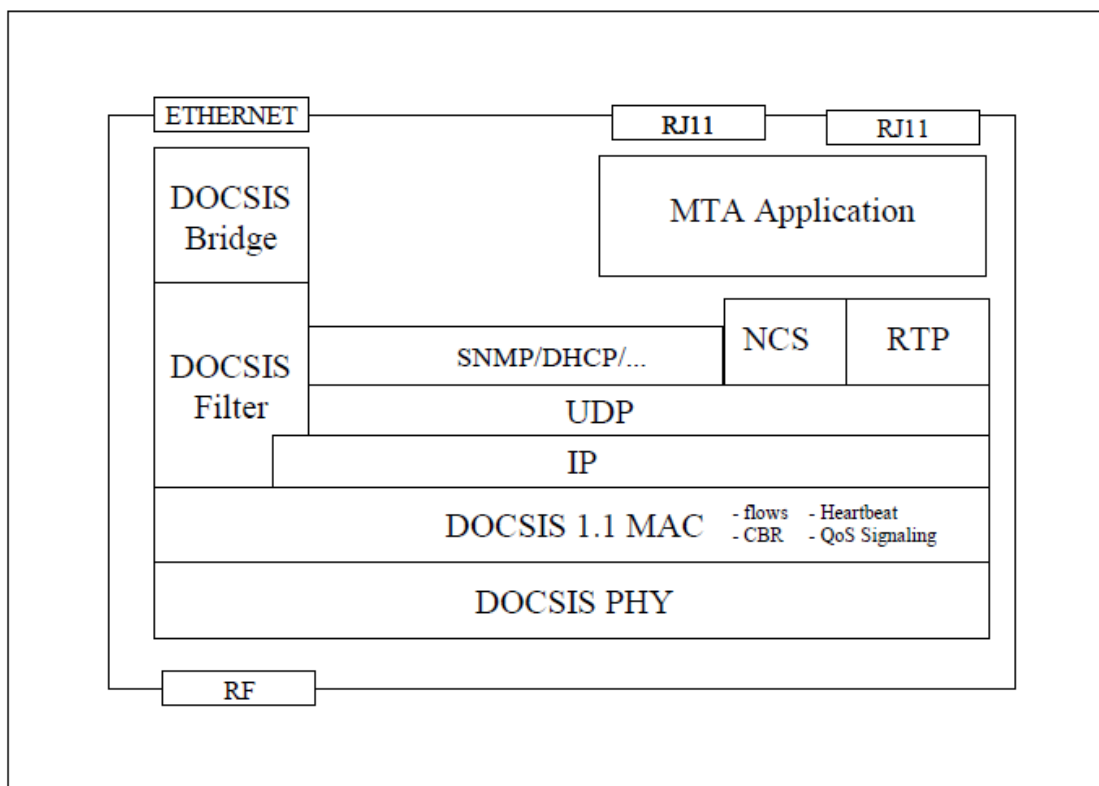
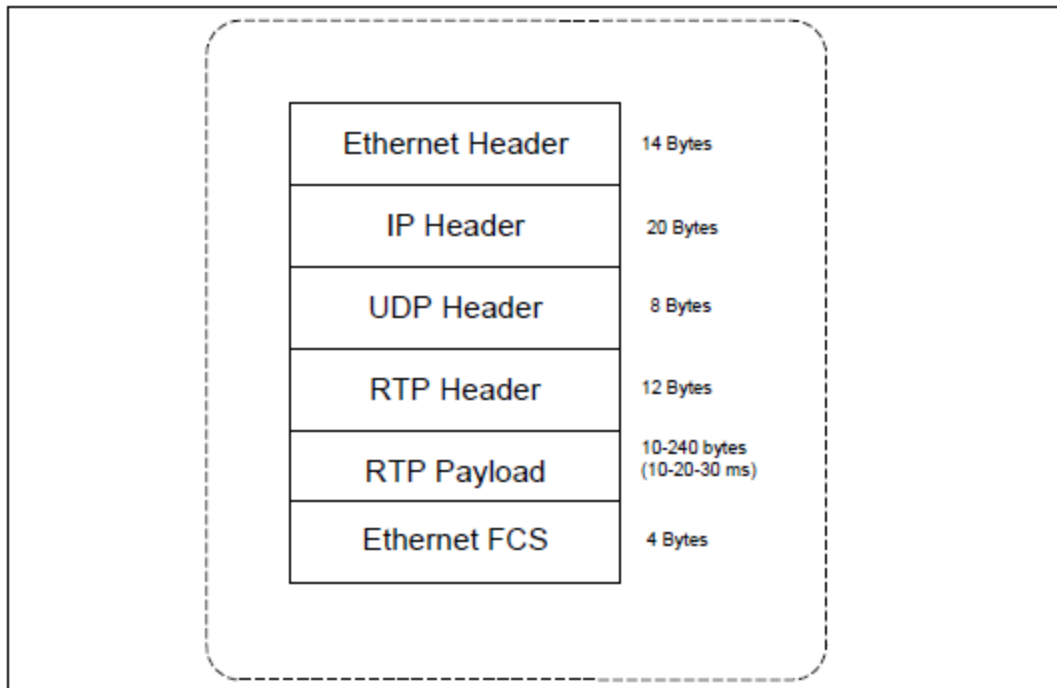


Figure 4. E-MTA Conceptual Functional Architecture

EX1007 at 20

101. PacketCable states, when referring to the E-MTA, that “[t]he packet format for RTP data transmitted over IP over Ethernet is depicted in Figure 7,” which shows RTP encapsulated by UDP. EX1007 at 32. In keeping with encapsulation and decapsulation principles, UDP (ISO layer 4) is encapsulated with the Internet Protocol (IP) (ISO layer 3). Further, the IP packet is encapsulated by Ethernet. EX1007 at 32.



102. EX1007 at 32

103. The PacketCable specification describes how PacketCable provides its secure transport services, stating that it provides “*End-to-end media packets between two MTAs*” and describes the “interface” called “pkt-s6”. Herein, MTAs—not the applications that interface with the MTAs—provide secure transport services. MTAs provide packets between two MTAs that “*are encrypted directly with RC4, without any additional security layers.*” This further supports a POSITA’s understanding that MTA *devices* provide the secure transport services, not an “MTA Application.” With respect to “end-to-end media packets between two MTAs” and the “pk-s6” protocol interface, PacketCable states “*An MMH-based MAC (Message Authentication Code) optionally provides message integrity.*”

VL Collective Exhibit 2004
IPR2023-00923

Meta Platforms, Inc. v. VL Collective IP LLC
Page 90 of 136

pkt-s6	MTA – MTA	End-to-end media packets between two MTAs, or between MTA and MG. RTP packets are encrypted directly with RC4, without any additional security layers. An MMH-based MAC (Message Authentication Code) optionally provides message integrity. Keys are distributed by the CMS to the two endpoints.
--------	-----------	--

EX1007 at 45

X. DR. HOUH DOES NOT SHOW THAT ANY CHALLENGED CLAIM WOULD HAVE BEEN OBVIOUS

A. Dr. Houh does not demonstrate that Handley in combination with Basturk (Ground I) discloses or renders obvious any challenged claim.

1. Dr. Houh does not demonstrate that a POSITA would have been motivated to combine Handley and Basturk to arrive at the claimed invention.

a. A POSITA would not have been motivated to modify Handley with Basturk as suggested by Dr. Houh.

104. Dr. Houh states that “[a] POSITA would have been motivated to modify *Handley* with *Basturk* for many reasons.” EX1003 ¶ 63. I disagree that a POSITA would have been motivated to modify Handley with Basturk as suggested by Dr. Houh..

105. Dr. Houh asserts that it would have been obvious to modify Handley with Basturk’s “on-the-fly” computation methods because it would purportedly “minimize[] latency” and “facilitate[] the eventual off-loading to the hardware” to “speed up the process.” EX1003 ¶¶ 63-64. Specifically, Dr. Houh claims that Basturk’s “on-the-fly” processing would enable “MAC generation [to] be carried

out *as data transmissions proceeds* rather than before *these operations* have completed,” and that “data transmission occurs faster using on-the-fly signature computation because the signature is done *in parallel with the transmission of the packet*, and then the computed signature is placed *at the end of the packet just as the transmission is completing*.” EX1003 ¶ 65 (emphasis added). I disagree with Dr. Houh for multiple reasons.

106. First, Dr. Houh’s characterization of Basturk’s “on-the-fly” processing is incorrect, and “on-the-fly” processing does not require or motivate placing the signature at the end of the packet (*i.e.*, in Basturk’s trailer) to achieve faster packet transmission or reception. Basturk’s statement that “MAC generation should be carried out as data transmission proceeds rather than before *these operations* have completed” is not referring to computing a MAC while the *packet* is being transmitted, as Dr. Houh suggests. EX1003 ¶ 65.

107. Instead, Basturk is referring to starting the process of computing the style of MAC during (or in parallel with) the process of *retrieving context data* for performing cryptographic operations. In other words, Basturk is referring to starting the process of MAC computation (*i.e.*, computing a hash function portion of a particular style of MAC) while the process of *retrieving context information* for

computing a cryptographic portion of the MAC is being performed. *See* § IX.B.3, *supra*. EX1006 at 8.

108. Basturk’s context retrieval process is not “the transmission of the *packet*,” as Dr. Houh suggests. EX1003 ¶ 65 (emphasis added). Indeed, Dr. Houh’s interpretation of Basturk is inaccurate, as a POSITA would understand that “packets” or Protocol Data Units (PDUs) at higher layers are encapsulated by the next lower layer, and the data is not “transmitted” bit-by-bit until the data reaches the physical layer (in conjunction with the link layer). *See* §§ VI.A (OSI) and VI.E (Ethernet). As such, PDUs are passed up and down the layers of a protocol stack only after PDU formation is completed, and transmission of the data doesn’t occur until it reaches the physical layer (in conjunction with the link layer) where the PDU for transmission is a bit. *See* §§ VI.A (OSI) and VI.E (Ethernet). POSITA would have understood that if a packet requires information to be computed (in Dr. Houh’s hypothetical, a signature), that information will always have to be computed before passing the PDU to the next layer. Thus, a POSITA would have understood that even with Basturk’s “on-the-fly” processing, packets would not be passed to the next layer until all computations for that packet are complete. A POSITA would therefore not have understood Basturk’s “on-the-fly” computation to be “done in parallel with the transmission of the packet” such that “the computed signature is placed at the

end of the packet just as the transmission is completing,” as Dr. Houh incorrectly suggests. EX1003 ¶ 65. A POSITA would therefore have understood that Basturk’s “on-the-fly” processing does not require the use of a trailer or placing the signature at the end of a packet.

109. Furthermore, as I explained above in Section IX.B.3, a POSITA would have understood that Basturk’s desire to start the MAC computation process before the completion of context retrieval does not require or motivate placing a MAC at the end of a packet in a trailer. *See* § IX.B.3, *supra*. As explained above, the “two major components” that Basturk teaches for achieving “on-the-fly” processing are (1) reducing the latency in retrieving the context information by putting the context information *early* in each packet (*i.e.*, in the packet’s header), and (2) using a select style of MAC that allows for beginning the signature (MAC) computation before receiving the MAC key from a context retrieval. *See* § IX.B.3, *supra*; EX1006 at 12-13. A POSITA would have understood that neither component is enabled using a trailer. *See* § IX.B.3, *supra*. Accordingly, a POSITA would not have been motivated to use a trailer to achieve “on-the-fly” processing. *See* § IX.B.3, *supra*.

110. In paragraphs 66 through 70 of his report, Dr. Houh suggests that Basturk’s “secure transport protocol is compatible with *Handley’s* internet conferencing architecture,” and that a POSITA would have found it obvious to use

Basturk’s “on-the-fly” authentication method with Handley’s internet conferencing architecture. EX1003 ¶¶ 66-70. I disagree. Basturk’s secure transport protocol is a transport level protocol that specifically operates at the transport layer, not at the application layer. *See* § IX.B.2. As Dr. Houh points out, “Basturk teaches its security features ‘reflect[] the need to add security functionality in a way which is consistent with, and best matches, the existing communication structure’ and so that ‘[a]pplications using the regular transport protocol [will] not have to be modified at all.’” EX1003 ¶ 66 (citing EX1006, 3, 5). Basturk accomplishes that goal by expressly *not* implementing its security features at the application layer, and instead providing security at the transport layer using key features (including a session key α that is generated at the transport layer and is known only to transport machines M_A and M_B and is unknown to the application layer) for separating its transport level security functions from (and not duplicating them in) the application layer. *See* § X.A.1.a.

b. Basturk’s protocol for reliable transport is incompatible with Handley’s use of RTP over UDP.

111. It would not have been obvious to a POSITA to modify Handley’s architecture, which uses RTP over the UDP, with Basturk’s transport protocol, which “is essentially an extension of the standard transport level *reliability* feature.” EX1006 at 4. A POSITA would have understood that “the standard transport level

reliability feature” is a feature provided by protocols, such as the well-known Transmission Control Protocol (“TCP”). EX2006. Reliability is the ability to “recover from data that is damaged, lost, duplicated, or delivered out of order,” and is achieved through timeout and retransmission policies, reordering packets received out of order based on sequence numbers, and eliminating duplicates, among other things. EX2006 at 10; *id.* at 16 (“Transmission is made reliable via the use of sequence numbers and acknowledgments.”). A POSITA would have understood that the increase in reliability of TCP comes at the cost of end-to-end delay caused by its reliability features.

112. A POSITA would also have understood that Basturk’s protocol, as “an extension of the standard transport level *reliability* feature,” would involve similar delays as other reliability protocols like TCP. Indeed, Basturk’s protocol achieves “secure delivery” by providing acknowledgements that include sequence numbers and are signed. *See* § IX.B.1; EX1006 at 12 (“Secure delivery is guaranteed because the ACKs include sequence numbers and are signed.”). A POSITA would therefore have understood that Basturk’s protocol is a reliable delivery protocol like TCP and would therefore have similar end-to-end delay drawbacks.

113. Handley teaches away from using reliability protocols like Basturk’s for transmitting RTP traffic. Handley teaches that “[r]eal-time internet traffic is

defined as datagrams that are delay sensitive,” such as where “where exceeding an end-to-end delay bound of a few hundred milliseconds renders the datagrams useless for the purpose they were intended.” EX1005 at 4-5. Handley states that “For the purposes of this definition, TCP traffic is normally not considered to be real-time traffic.” EX1005 at 5. A POSITA would have understood that Handley considers TCP traffic to be non-real-time because of the delay associated with its reliability features. A POSITA would understand that this is why Handley implements real-time RTP traffic over UDP, which is not a reliable delivery protocol, and distinguishes between real-time traffic (RTP over UDP) from non-real-time traffic (traffic over TCP) for prioritized processing. *See* § IX.A.2, *supra*; EX2007 at 1 (“[UDP] is transaction oriented, and delivery and duplicate protection are not guaranteed. Applications requiring ordered reliable delivery of streams of data should use the Transmission Control Protocol (TCP).”).

114. A POSITA would have understood that because Basturk’s protocol is a reliable delivery protocol like TCP, it would be unsuitable for transporting Handley’s real-time RTP traffic for the same reasons that Handley teaches TCP is unsuitable. Accordingly, a POSITA would have been discouraged and led away from modifying Handley’s implementation of RTP over UDP with Basturk’s reliable transport protocol.

115. Furthermore, A POSITA would have understood that because Basturk's protocol is a reliably delivery protocol like TCP, it would be unsuitable for use with Handley which relies on a multicast network for delivery of packets. Applications relying on multicast delivery networks can easily scale to many receivers because the senders do not need to do any additional work when a new receiver joins (or leaves) a multicast group. The same applies to broadcast delivery. Furthermore, a POSITA would understand that a multicast delivery is incompatible with a reliable delivery protocol such as TCP or any protocol based on TCP or that has similar functionality. This is because TCP is a connection-oriented protocol, which means it establishes a direct connection between two endpoints before data can be sent, where the sender and receiver know who each other are. EX2006 at 11 ("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). ... 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."). This connection setup is referred to as a three-way handshake. EX2006 at 33-34, 36 ("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.”). Using a reliability protocol like TCP or Basturk would be incompatible with Handley’s statement of “Senders never *need* to know who the receivers are.” EX1005 at 3.

116. TCP requires the sender to participate in a three-way handshake called connection setup with each new receiver. *See* EX2006 at 11, 33-34, 36. To do this, the sender needs to use a particular IP address unique to the receiver. *See* EX2006 at 16 (“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.”). Thus, senders must know who each of the receivers are. EX2006 at 11, 16. Use of such a reliable delivery protocol is further disadvantageous to the sender because the sender needs to process acknowledgement messages sent by receivers to determine if a retransmission of a message is required by a sender. *See, e.g.*, EX2006 at 10 (“If the ACK is not received within a timeout interval, the data is retransmitted.”). Basturk’s protocol similarly uses acknowledgements, and thus would be similarly disadvantageous and thus would not be combined with Handley for similar reasons. EX1006 at 4 (“Then, she sends a transport level acknowledgement of received data to the sending port.”), 4-5 (“It can also sign the acknowledgments messages which exist only at the transport level.”), 5 (“Note that acknowledgments, which have no data, are also signed.”), 12

(“Secure delivery is guaranteed because the ACKs include sequence numbers and are signed.”).

2. **Dr. Houh does not demonstrate that a POSITA would have found claim 1 obvious over Handley and Basturk.**
 - a. **[Element 1.1] “providing transmitter-to-receiver authentication at a Real Time Transport Protocol (RTP) packet level as an application protocol on an application layer”**
 - i. **Basturk teaches away from modifying Handley’s application layer with Basturk’s transport protocol.**

117. A person of ordinary skill in the art would have understood that Basturk teaches away from modifying the application layer with its transport level protocol. Basturk expressly states that its design of a secure transport protocol “*reflects the need to add security functionality in a way which is consistent with, and best matches, the existing communication structure*” and so that “[a]pplications using the regular transport protocol should not have to be modified at all.” EX1006 at 3, 5. Furthermore, by providing its transport level protocol with “the standard application level security features” as options, Basturk teaches eliminating security features from the application layer and providing them as optional *transport layer* security features to reduce computational “burden” and “cost” when those features are not wanted. EX1006 at 6, 9. A POSITA would have understood Basturk to

discourage and lead away from modifying an existing application layer communication structure to include the features of Basturk's protocol.

118. A POSITA would have also understood that Basturk teaches away from modifying an application to implement its transport protocol, as Basturk indicates that it would be "poor design," "belies the layered communication structure," would "adversely affect performance," would "mak[e] implementation significantly harder," duplicate functionality" and "degrade performance." EX1006 at 4 n.3, 12. A POSITA would have understood this to teach away from and discourage implementing Basturk's transport protocol at the application layer.

**ii. Dr. Houh does not show that Handley or
Basturk discloses this claim element.**

119. Dr. Houh opines that "a POSITA would have understood that *Handley* discloses the implementation of *authentication* at the RTP packet level because using *encryption* schemes like DES or MD5 to manipulate the RTP packet directly occurs at the RTP packet level since Handley teaches that *the encryption is specified by RTP* to encrypt RTP packets" EX1003 ¶ 78 (emphasis added). I disagree. A POSITA would have understood that while the RTP specification includes provisions for packet *encryption*, RTP explicitly states that "[a]uthentication and message integrity are not defined" in the RTP specification. *See* § VI.B.2; EX1008 at 50. Further a POSITA would not consider encryption for the purposes of

“confidentiality” as implemented in RTP to be the same as authentication much less “providing transmitter-to-receiver authentication at a Real Time Transport Protocol (RTP) packet level as an application protocol on an application layer”. *See* EX1008 at 49. Accordingly, I disagree that a POSITA would have understood Handley’s mentioning of RTP’s encryption schemes to disclose providing authentication at the RTP packet level.

120. Dr. Houh concedes that “Basturk does not explicitly disclose RTP.” EX1003 ¶ 66. He identifies nothing in Basturk that discloses “providing transmitter-to-receiver authentication at a Real Time Transport Protocol (RTP) packet level as an application protocol on an application layer.” *See* EX1003 ¶¶79-82.

iii. A POSITA would not have found it obvious to combine the teachings of Handley and Basturk as Dr. Houh suggests.

121. For the reasons explained in Section X.A.1 above, I disagree with Dr. Houh’s opinion that “a POSITA would have found it obvious to implement Basturk’s secure transport protocol and its authentication techniques to Handley’s internet conferencing architecture.” EX1003 ¶80 (citing Section VII.A.1).

122. I also disagree with Dr. Houh’s other opinions that Element [1.1] would have been obvious to a POSITA in view of Handley in combination with Basturk.

123. Dr. Houh points to Basturk’s statements that “[a]pplication level security features,” including encrypting and signing data before passing it to the transport layer, “are easily (and usually) provided by application level security.” EX1003 ¶ 82. I disagree that Basturk’s statements disclose or render obvious Element [1.1] to a POSITA. Basturk does not disclose or suggest “providing transmitter-to-receiver authentication at a [RTP] packet level” or that doing so is “easily” or “usually” done. Indeed, Dr. Houh concedes that “Basturk does not explicitly disclose RTP.” EX1003 ¶ 66. Furthermore, the RTP specification itself—the document that defines the features of RTP—makes clear RTP does not provide authentication or integrity, EX1008 at 50 (“*[a]uthentication and message integrity are not defined*”), and it expresses the clear expectation that “***authentication and integrity services will be provided by lower layer protocols***,” *i.e.*, at a layer lower than RTP, and thus not at the application layer. *See* § VI.B.2, *supra*; EX1008 at 50 (emphasis added). Further a POSITA would not consider encryption for the purposes of “confidentiality” as implemented in RTP to be the same as authentication much less “providing transmitter-to-receiver authentication at a Real Time Transport Protocol (RTP) packet level as an application protocol on an application layer”. *See* EX1008 at 49. A POSITA would therefore not have found it obvious to provide

authentication at an RTP packet level as an application protocol on an application layer in view of the combination of Handley and Basturk.

124. Dr. Houh also points to Basturk's transport level integrity feature whereby "data is signed by [transport machine] M_A under the session key α " and "[o]n receiving a signed packet, M_B uses α to verify the signature." EX1003 ¶ 81. Dr. Houh then states that "key elements" of this technique "takes place as an application protocol on an application layer" because "cryptographic functions such as key exchange and selection of a key to use for communication, occur at the application layer, since such keys are passed from the application layer to the transport layer." I disagree.

125. As explained above, it is the session key α , not the long-lived key a , that is the key used to apply cryptographic functions in Basturk's protocol. § IX.B.2. Once the application A passes the long-lived key a to the transport level, *i.e.*, to the secure transport machine M_A , "***the machine*** initiates and executes an authentication and key exchange protocol with M_B ," which "results in the distribution of a new (fresh) session key α while making sure that this key is authentic (ie. known only to M_A and M_B)."
See § IX.B.2., *supra*; EX1006 at 9 (emphasis added). Then, "the task of applying any cryptographic functions or taking other security steps" is left to the transport machine at the transport layer. *See* § IX.B.2., *supra*; EX1006 at 4, 8.

Basturk is clear that “only the transport machines M_A and M_B know about the session key,” and that it is never seen at the application layer. See § IX.B.2; EX1006 at 10 (“ A and B never see it.”). Accordingly, a POSITA would have understood that Basturk’s protocol, which Basturk makes clear is a transport level protocol below the application layer, does not provide authentication as an application protocol on an application layer.

b. [Element 1.2] “by inserting, at the transmitter, authentication data at end of a whole RTP packet payload;”

126. Dr. Houh opines that “a POSITA would have understood and found obvious” that in the proposed combination of Handley and Basturk, authentication data (a MAC) “would be placed at the end of the whole RTP packet payload *to enable ‘on-the-fly’ signature computation and verification, allowing the transmitter to transmit data while simultaneously computing the authentication data.*” EX1003 ¶ 85 (emphasis added). I disagree for multiple reasons.

127. As I explained above in Sections IX.B.4 and X.A.1.a, I disagree with Dr. Houh’s characterization of Basturk’s “on-the-fly” processing, which a POSITA would have understood does not require and is not “enabled” by placing the signature (MAC) in a trailer. See §§ IX.B.4 and X.A.1.a. As also explained above, a POSITA would have understood that Basturk’s “on-the-fly” technique permits signature

generation/verification to begin in parallel with *context retrieval*, not in parallel with *packet transmission*, and thus does not require or motivate the use of a trailer or placing the signature at the end of the packet. See §§ IX.B.4 and X.A.1.a.

128. I disagree with Dr. Houh’s opinion that “the Ethernet standard ... employs the same design choice reasoning to include a ‘Frame Check Sequence’ (‘FCS’) at the end of the Ethernet frame *so it can be computed during transmission and appended to the end of the frame.*” EX1003 ¶ 85. In the context of Ethernet, transmission refers to generating and transmitting electromagnetic waves on a physical medium at ISO layer 1 (in conjunction with the link layer ISO layer 2) where the PDU is a single bit. See EX1014 at 32, 142-143; see § VI.E, *supra*.

129. The PDU needs to go through the process of encapsulation down the protocol stack until it reaches the physical layer, which is where bit-by-bit transmission on the physical layer occurs. See §§ VI.A (OSI) and VI.E (Ethernet), *supra*. This transmission at the physical layer, however, does not happen before the processing at higher layers is complete. Accordingly, the use of on-the-fly processing and/or a trail in a higher layer PDU would not allow “signature computation [to be] done in parallel with the transmission of the packet,” as Dr. Houh incorrectly suggests. EX1003, ¶85. Thus, a POSITA would not have been

motivated to add a trailer to an RTP packet or use on-the-fly processing based on the physical layer transmission process that Dr. Houh describes. EX1003, ¶85.

130. Simply put, Dr. Houh’s arguments regarding “signature computation [being] done in parallel with the transmission of the packet” are inaccurate because bit-by-bit transmission occurs at the physical layer (not at the RTP packet level), and the *location* of authentication data within the higher layer PDUs (such as an RTP packet) would not make that transmission faster as Dr. Houh suggests.

131. To the extent Dr. Houh is suggesting that on-the-fly processing/using a trailer somehow results in bit-by-bit processing benefits occurring across all layers going down to the physical layer where transmission occurs bit-by-bit; he hasn’t shown this. This is contrary to how network protocol stacks operate. PDUs are created at higher layers and passed to lower layers for encapsulation. *See* § VI.A, *supra*.

132. To the extent Dr. Houh is suggesting that somehow bit-by-bit processing occurs at one or more layers above the physical layer; he hasn’t shown how this decreases latency in transmission at the physical layer (where transmission occurs). For example, it makes no sense to pass data bit by bit from the transport layer (e.g. UDP or TCP) to the networking layer (e.g. IP). (Furthermore, it makes no sense to create IP, UDP, or TCP payloads of individual bits.) This is not how

protocol stacks are implemented. At each layer, encapsulation requires the receipt of the entire PDU before passing it down to the lower layer. Protocol stacks operating bit by bit would cause tremendous computational complexity issues and would lead to increased latency and storage issues. This is because doing so would require the task of managing system state information to try to keep track of individual bits associated with partial data being communicated at each layer instead of processing entire PDUs.

133. Thus, for all these reasons, a POSITA would not have been motivated to use on-the-fly process or add a trailer to an RTP packet based on principles of bit-by-bit transmission at the physical layer or when passing RTP data to a lower layer.

134. Additionally, RTP can look for information in the *header* to check for validity issues. EX1008 at 59. This further demonstrates that a POSITA would not have been motivated to add a trailer to an RTP packet based on the location of error checking data in a lower level protocol. To the contrary, RTP provides a method for extending the RTP *header* to support additional information, which can advantageously be “*less expensive to process because it is not conditional nor in a variable location.*” See EX1008 at 14-15; § VI.B.3, *supra*. Accordingly, a POSITA would not have been motivated to use on-the-fly or a trailer at the end of an RTP packet based on the use of trailers in other lower level protocols.

c. **[Element 1.3] “ascertaining, by the receiver, whether the receiver knows the transmitter based on the RTP packet level authentication data; and”**

135. Dr. Houh concedes that “Handley does not expressly disclose ‘inserting, at the transmitter, authentication data at end of a whole RTP packet payload,’ EX1003 ¶ 64, thus he has not shown that Handley discloses ascertaining “whether the receiver knows the transmitter based on the RTP packet level authentication data,” as the claim requires. *See* § X.A.2.a-b, *supra*.

136. Furthermore, Dr. Houh concedes that Basturk does not disclose RTP, EX1003 ¶¶ 66, 85, 260, thus he has not shown that Basturk discloses ascertaining “whether the receiver knows the transmitter based on the RTP packet level authentication data,” as the claim requires. *See* § X.A.2.a-b, *supra*.

137. Dr. Houh alleges that Element [1.3] would have been obvious over the combination of Basturk and Handley, based on his opinions for Elements [1.1] and [1.2]. *See* EX1003 ¶¶ 90-91. But “*the* RTP packet level authentication data” of Element [1.3] refers to the authentication data “at end of a whole RTP packet payload” that was inserted “as an application protocol on an application layer” per the requirements of Elements [1.1] and [1.2]. For the reasons I explained above, Dr. Houh has not shown that it would have been obvious to a POSITA to combine Handley and Basturk to arrive at the requirements of Elements [1.1] and [1.2]. *See*

§ X.A.2.a-b, *supra*. It is my opinion that Dr. Houh has not shown that Element [1.3] was disclosed or rendered obvious for the same reasons.

3. Independent Claims 25-28

138. Dr. Houh's arguments for independent claims 25-28 rely on his arguments for Elements [1.1]-[1.3]. *See* EX1003 ¶¶132-164. Accordingly, he has failed to show that claims 25-28 would have been obvious over the combination of Handley and Basturk for the reasons explained above with respect to Elements [1.1]-[1.3]. *See* §§X.A.1-2, *supra*.

4. All dependent claims are patentable over Handley in combination with Basturk.

139. Claims 2-24 depend on claim 1. Claim 1 is patentable over Handley in combination with Basturk for the reasons explained above. *See* §§X.A.1-2, *supra*. Accordingly, by their dependency on claim 1, claims 2-24 are patentable over Handley in combination with Basturk.

B. Dr. Houh does not demonstrate that PacketCable in combination with Handley (Ground II) discloses or renders obvious any challenged claim.

1. Dr. Houh fails to show that a POSITA would have found claim 1 obvious over PacketCable and Handley.

a. Element [1.1] "providing transmitter-to-receiver authentication at a Real Time Transport Protocol

(RTP) packet level as an application protocol on an application layer”

140. Dr. Houh opines that “PacketCable discloses this element or at least rendered it obvious.” EX1003, ¶174. Dr. Houh argues that “a POSITA would have understood (1) that manipulating RTP packets, e.g., implementing RC4 or adding a MAC, results in authentication at the packet level and (2) RTP packets are created at the application layer because RTP is an application protocol. As such, PacketCable discloses this element.” *Id.* I disagree that Dr. Houh has shown that PacketCable discloses or would have rendered obvious this claim element.

141. Dr. Houh does not explain why “implementing RC4” indeed “results in authentication” much less “authentication at the packet level”. A POSITA would understand that RC4 is an encryption algorithm for protecting confidentiality of data and not for authentication. Furthermore, Dr. Houh does not show that PacketCable’s alleged “adding a MAC” (with or without RC4) is provided “as an application protocol on an application layer,” as this element requires. Dr. Houh argues that the RTP specification’s description of the use of “application level framing” means that “some protocol functions are implemented in the application,” but Dr. Houh identifies no RTP protocol functions “implemented in the application” in PacketCable, e.g., in an “MTA Application.” *See* EX1003, ¶ 52. Nor does he show that PacketCable’s implementation of RTP uses the application level framing

concept as described in the RTP specification, which is that RTP may “provide the information required by a particular application” and may be “integrated into the application processing rather than being implemented as a separate layer.” *See* EX1008 at 4.

142. To the contrary, as explained above, PacketCable indicates to a POSITA that MTA *devices* generally are responsible for providing secure transport services, including “Authentication, confidentiality and integrity of some messages between the MTA and other PacketCable network elements,” namely those sent over the “pkt-s6” interface, which is a network interface for sending “[e]nd-to-end media packets between two MTAs” generally. *See* § IX.C.1, *supra*. However, PacketCable does not disclose “integration into application processing”, *e.g.*, the “MTA Application.” *Id.* PacketCable indicates to a POSITA that PacketCable uses the MTA device to provide secure transport services whereby “RTP packets are encrypted directly with RC4, without any additional security layers” and “[a]n MMH-based MAC (Message Authentication Code) optionally provides message integrity.” EX1007 at 45. PacketCable, however, does not disclose how this optional additional security feature would be included. PacketCable relies on RTP (RFC 1889) “to transport all media streams,” EX1007 at 31, 57, but authentication and integrity are expressly “not defined” in the RTP protocol specification RFC

1889. EX1008 at 50; *see* § VI.B.2, *supra*. Accordingly, Dr. Houh has not shown that a POSITA would have understood PacketCable’s “implementing RC4 or adding a MAC” to be “providing transmitter-to-receiver authentication at a Real Time Transport Protocol (RTP) packet level as an application protocol on an application layer”. *See* § IX.C.1, *supra*.

143. Dr. Houh further argues that this element “would have been obvious in view of PacketCable in combination with Handley,” incorporating by references his opinions for Element [1.1] under Ground I for Handley combined with Basturk. EX1003, ¶175. I disagree with Dr. Houh for the reasons I explained above regarding Element [1.1] under Ground I. *See* § X.A.2.a, *supra*.

- b. [Element 1.2] “by inserting, at the transmitter, authentication data at end of a whole RTP packet payload;”**
 - i. Dr. Houh does not show that PacketCable discloses providing RTP packet-level authentication “by inserting, at the transmitter, authentication data at end of a whole RTP packet payload.”**

144. Dr. Houh opines that “PacketCable discloses this element or at least rendered it obvious.” EX1003 ¶ 177. However, despite alleging that PacketCable “discloses” this claim element, Dr. Houh does not show where PacketCable allegedly discloses it. He identifies no place in PacketCable that actually discloses

this claim element. *See* EX1003 ¶¶ 176-179. Nor does Dr. Houh show that this claim element is inherently disclosed, as Dr. Houh suggests at least three different places where data could be placed in a packet (*i.e.*, “in the middle of the packet payload,” “in a packet header,” and “in a packet trailer”). EX1003 ¶ 177.

145. A POSITA would have understood that PacketCable does not disclose this element, either expressly or inherently. PacketCable says that the PacketCable architecture “specif[ies], for each defined protocol interface, the underlying security mechanisms (such as IPSec) that provide the protocol interface with the security services it requires, e.g., authentication, integrity, confidentiality.” EX1006 at 42. PacketCable states that for MTA-MTA or MTA-MG communications, “RTP packets are encrypted directly with RC4, without any additional security layers,” and that “[a]n MMH-based MAC (Message Authentication Code) optionally provides message integrity.” *Id.* 45.

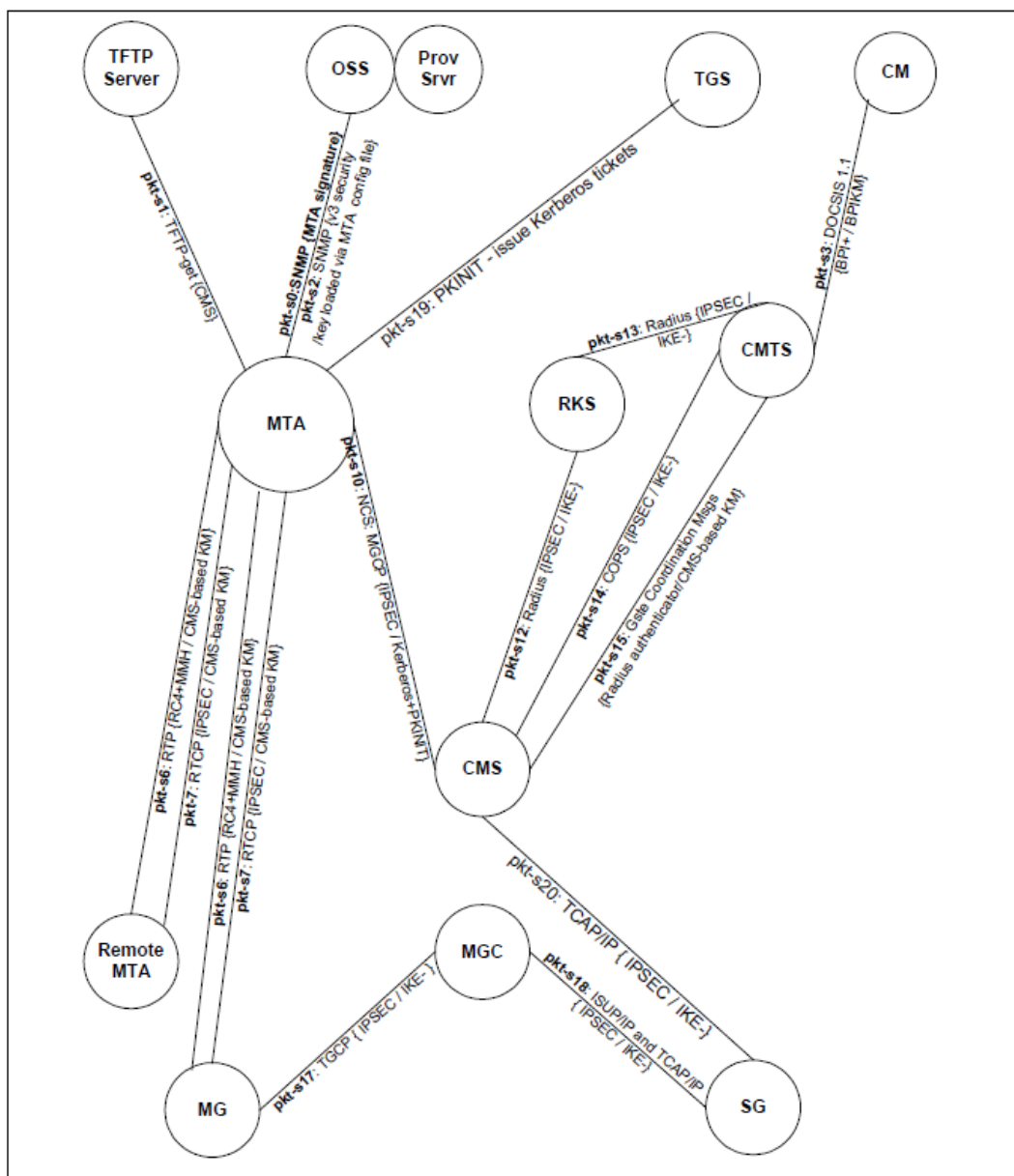


Figure 13. PacketCable Security Interfaces

EX1007, 44

146. According to PacketCable, “[e]ncryption is applied to the packet’s payload but not to its header,” and “[t]he MAC computation” for the optional MAC “spans the packet’s unencrypted header and encrypted payload.” *Id.* 47.

VL Collective Exhibit 2004
IPR2023-00923

Meta Platforms, Inc. v. VL Collective IP LLC
Page 115 of 136

147. Based on the above portions of PacketCable, a POSITA would have understood that PacketCable does not disclose “inserting, at the transmitter, authentication data at end of a whole RTP packet payload,” as required by this claim element. While PacketCable mentions that “[e]ach RTP packet may include an optional message authentication code (MAC)” and that “MAC computation spans the packet’s unencrypted header and encrypted payload,” a POSITA would not have understood this to disclose (explicitly or inherently) inserting the optional MAC “at end of a whole RTP packet payload.” EX1007 at 47. PacketCable is completely silent as to where in an RTP packet the optional MAC would be inserted. Indeed, Dr. Houh does not show that PacketCable discloses placing the MAC at any of the theoretical locations he suggests. *See* EX1003 ¶ 178.

148. PacketCable’s non-disclosure of Element [1.2] is consistent with a POSITA’s understanding of the RTP protocol specification, which explicitly does not define an authentication service, and instead states that “[a]uthentication and message integrity are not defined in the current specification of RTP” and teaches that “[i]t is expected that authentication and integrity services will be provided by lower layer protocols in the future.” EX1008 at 50 (emphasis added). That is, the RTP specification does not disclose inserting a MAC “at end of a whole RTP packet payload,” and, in fact, contemplates authentication and integrity services would be

provided by *lower* layer protocols instead. A person of ordinary skill in the art would therefore have understood that PacketCable does not disclose (explicitly or inherently) “inserting, at the transmitter, authentication data at end of a whole RTP packet payload.”

ii. Dr. Houh does not demonstrate that [Element 1.2] would have been obvious to a POSITA in view of PacketCable and Handley.

149. Dr. Houh asserts that:

A POSITA would have understood and found obvious that placement of authentication data in the middle of the packet payload would negatively impact performance, cause packet fragmentation issues, and complicate the implementation of networking software, since the data in the packet would be interrupted by the authentication data, and extra data fields would be needed to identify the exact location of the authentication data within the packet data. As such, a POSITA would have avoided such placement.

EX1003 ¶ 178. I disagree with Dr. Houh’s broad, general assertions.

150. Dr. Houh does not identify any negative impacts to “performance” that would result from placing, for example, a MAC in “in the middle” of a packet payload or even in the middle of a packet header. In the context of RTP, a POSITA would have understood that RTP provides a header extension mechanisms to allow

additional data to be placed in the header. *See* § VI.B.3, *supra*. A POSITA would have understood that the RTP header is comprised of fixed-length or length-determinable portions (including the extended portion), making it computationally easier (and thus more efficient) to find the data there as opposed to the payload area, which is of a variable (and undefined) length and would require more computational resources to find the data there. *See* § VI.B.3, *supra*.

151. Dr. Houh does not identify any “packet fragmentation issues” that would favor putting data at the end of an RTP packet payload. PacketCable states that “[e]ach RTP packet may include an optional message authentication code (MAC).” EX1007 at 47. Given that “each” packet could include a MAC, Dr. Houh has not identified a reason why putting it in the middle of the payload or in the header would cause a fragmentation issue that would be avoided by putting it at the end.

152. Dr. Houh also identifies no implementation complications that would favor putting PacketCable’s MAC at the end of an RTP packet instead of the middle of the payload area or in the header. Dr. Houh mentions that “data in the packet would be interrupted” and that “extra data fields would be needed to identify the exact location” of the MAC, but he offers no reason why a need for additional data fields to locate data would favor placing PacketCable’s MAC at the end of an RTP packet payload (which has a variable and undefined length). EX1003, ¶178. A

POSITA would have understood that data can be found more quickly in an area with defined and/or fixed-length (or easily-determinable-length) fields, such as a header. And, as explained above, RTP provides mechanisms for placing additional data in the header (*e.g.*, in a header extension), which is advantageously “less expensive to process because it is not conditional nor in a variable location.” *See* EX1008 at 14-15; §§ VI.B.3 and X.A.2.b, *supra*. Moreover, a POSITA would have known and considered that RTP looks for information in the header to check for validity issues, including “to verify that incoming packets have been correctly decrypted.” EX1008 at 59; *See* § VI.B.3, *supra*.

153. Dr. Houh further asserts that:

With this knowledge, a POSITA would have understood available authentication data placement options to include either at the beginning or the end of the payload, either in a packet header (at the beginning as ubiquitous in packets) or in packet trailer (at the end which are already present in certain types of packets such as in Ethernet frames). Further, a POSITA would have understood that the authentication data would go at the end of the whole RTP packet payload because it would reduce the latency of delivering the packet since placing the authentication data at the beginning of the payload requires storing the packet, computing the authentication data, transmitting the packet to the network, and placing the authentication data at the end of the

payload allows the packet to be transmitted to the network while calculating the authentication data and then adding the authentication data to the end of the payload while the packet transmission is being completed. A POSITA would have understood and found obvious that placing authentication at the end of the payload would allow the transmitter to transmit data while simultaneously computing the checksum and preventing any unnecessary transmission delay by having to first compute the checksum and then transmit the payload. Thus, a POSITA would have understood and found obvious that placing authentication at the end of the whole RTP packet payload would be the best solution to increase transmission speeds and decrease memory usage needed to encrypt and decrypt the data, since computations can be performed as the packet is being received.

EX1003 ¶ 178. I disagree with Dr. Houh’s broad and general assertions.

154. I disagree with Dr. Houh’s conclusion based on his oversimplified “header” versus “trailer” dichotomy, as RTP does not define a trailer and instead defines a mechanism for placing additional information in the header, as explained above.

155. Furthermore, as explained above, RTP can rely on the underlying UDP protocol for providing checksum services, EX1008 at 3; § X.A.2.b, *supra*, thus a POSITA would not have been motivated to include a trailer at the end of an RTP packet based on the use of checksums in lower layer protocols. RTP also looks for

VL Collective Exhibit 2004

IPR2023-00923

Meta Platforms, Inc. v. VL Collective IP LLC

Page 120 of 136

information in the *header* to check for validity issues, including “to verify that incoming packets have been correctly decrypted,” EX1008 at 59, which further demonstrates that a POSITA would not have been motivated to add a trailer to an RTP packet based on the location of error checking data in a lower level protocol. *See* § X.A.2.b, *supra*.

156. I also disagree with Dr. Houh’s general suggestion that a POSITA would have understood that “the authentication data would go at the end of the whole RTP packet payload because it would reduce the latency of delivering the packet *since* placing the authentication data at the beginning of the payload requires storing the packet, computing the authentication data, transmitting the packet to the network” I disagree with Dr. Houh’s concern about “storing the packet”. For example, all PDU components need to be stored at least in temporary memory as the PDU is being prepared for passing down the protocol stack to the next layer. A POSITA would not have been motivated to add a trailer to end of the whole RTP packet payload based to avoid storing the packet. I further disagree that “computing the authentication data” would motivate the use of a trailer. The same authentication process would be used to compute the authentication data if the authentication data were placed in the header or the beginning of the packet payload. Thus, a POSITA

would not have been motivated to add a trailer to end of the whole RTP packet based on computing the authentication data.

157. A POSITA also would not have been motivated to add a trailer to end of the whole RTP to “reduce the latency of delivering the packet” at the sender side as part of “transmitting the packet to the network” to “allow the transmitter to transmit data while simultaneously computing the checksum and prevent any unnecessary transmission delay by having to first compute the checksum and then transmit the payload.” EX1003 ¶178.

158. First, a POSITA would have understood that RTP provides a mechanism placing additional information in the header. *See* analysis above. Further, the PDU needs to go through the process of encapsulation down the protocol stack until it reaches the physical layer, which is where bit-by-bit transmission on the physical layer occurs. This transmission at the physical layer, however, does not depend on the location of authentication data within the RTP packet. Higher level protocol layers are simply encapsulated within the data link layer frame. Ethernet does not, for example, investigate these payloads to do anything different based on where a higher layer placed certain data. For similar reasons, the location of authentication data in a higher layer PDU would have no impact on whether one would or would not need to “first compute the checksum and then transmit the

payload.” A POSITA would not have been motivated to add a trailer to an RTP packet based on the physical layer transmission process that Dr. Houh describes.

159. Simply put, Dr. Houh’s arguments regarding “placement of authentication at the end of the whole RTP packet payload” are inaccurate because bit-by-bit transmission occurs at the physical layer (not at the RTP packet level), and the location of authentication data within the higher layer PDUs (such as an RTP packet) would not make that transmission faster as Dr. Houh suggests.

160. To the extent Dr. Houh is suggesting that “placement of authentication data at the end of the whole RTP packet payload” somehow results in bit-by-bit processing benefits occurring across all layers going down to the physical layer where transmission occurs bit-by-bit; he hasn’t shown this. This is contrary to how network protocol stacks operate. PDUs are created at higher layers and passed to lower layers for encapsulation. *See* § VI.A, *supra*.

161. To the extent Dr. Hough is suggesting that somehow bit-by-bit processing occurs at one or more layers above the physical layer; he hasn’t shown how this decreases latency in transmission at the physical layer (where transmission occurs). For example, it makes no sense to pass data bit by bit from the transport layer (e.g. UDP or TCP) to the networking layer (e.g. IP). (Furthermore, it makes no sense to create IP, UDP, or TCP payloads of individual bits.) This is not how

protocol stacks are implemented. At each layer, encapsulation requires the receipt of the entire PDU before passing it down to the lower layer. Protocol stacks operating bit by bit would cause tremendous computational complexity issues and would lead to increased latency and storage issues. This is because doing so would require the task of managing system state information to try to keep track of individual bits associated with partial data being communicated at each layer instead of processing entire PDUs.

162. Thus, for all these reasons, a POSITA would not have been motivated to add a trailer to an RTP packet based on principles of bit-by-bit transmission at the physical layer or when passing RTP data to a lower layer.

163. Accordingly, Dr. Houh has not shown that a POSITA would have found it obvious to put PacketCable's MAC in a trailer at the end of an RTP packet payload.

164. Dr. Houh further incorporates his opinions regarding Element [1.2] under Ground I, which, as I explained above, do not show that a POSITA would have found Element [1.2] obvious. *See* § X.A.2.b, *supra*.

c. [Element 1.3] “ascertaining, by the receiver, whether the receiver knows the transmitter based on the RTP packet level authentication data; and”

165. Dr. Houh alleges that Element [1.3] would have been obvious over the combination of PacketCable and Handley, based on his opinions for Elements [1.1]

and [1.2] under Ground II. *See* EX1003 ¶ 180. But “*the* RTP packet level authentication data” of Element [1.3] refers to the authentication data that was inserted “at end of a whole RTP packet payload” per the requirements of Element [1.2]. For the reasons I explained above, Dr. Houh has not shown that it would have been obvious to a POSITA to combine PacketCable and Handley to arrive at the requirements of Element and [1.2]. *See* §§ X.B.1.a-b, *supra*. It is my opinion that Dr. Houh has not shown that Element [1.3] was disclosed or rendered obvious for the same reasons. Dr. Houh’s additional opinions regarding Element [1.3] do not remedy those deficiencies.

2. Independent Claims 25-28

166. Dr. Houh’s arguments for independent claims 25-28 rely on his arguments for Elements [1.1]-[1.3]. *See* EX1003, ¶¶ 227-257. Accordingly, Petitioners have failed to show that claims 25-28 would have been obvious over the combination of PacketCable and Handley for the reasons explained above with respect to Elements [1.1]-[1.3]. *See* §§ X.B.1, *supra*.

3. Dr. Houh does not demonstrate that a POSITA would have found claims 22 and 24 obvious over PacketCable and Handley.

a. [22] The method as claimed in claim 21, wherein the receiver and transmitter are each disposed in at least one of a switching node and a switching installation.

167. Dr. Houh alleges that PacketCable’s “media gateway” acts as a “switch,” and that “[i]nclusion of a media gateway would have been obvious to a POSITA.” EX1003 ¶ 224. However, Dr. Houh fails to show that PacketCable’s media gateway satisfies the requirements of the “receiver” or “transmitter” of claim 1. Dr. Houh does not explain whether or how the act of switching data “between a particular IP flow and a particular PSTN port on the gateway” involves “inserting... authentication data at end of a whole RTP packet payload” (let alone “as an application protocol on an application layer”) or “ascertaining, by the receiver, whether the receiver knows the transmitter based on the RTP packet level authentication data.” A POSITA would have understood that IP is at a lower layer than the application layer. *See, e.g.*, EX1007 at 20.

b. [24] The method as claimed in claim 23, wherein the receiver and transmitter are each disposed in at least one of a switching node and a switching installation.

168. For the same reasons that I explained for claim 22 under Ground II, Dr. Houh has not shown that the combination of PacketCable and Handley discloses or renders obvious claim 24.

4. All dependent claims are patentable over PacketCable in combination with Handley.

Claims 2-24 depend on claim 1. Claim 1 is patentable over PacketCable in combination with Handley for the reasons explained above. *See* § X.B.1, *supra*. Accordingly, by their dependency on claim 1, claims 2-24 are patentable over PacketCable in combination with Handley.

C. Dr. Houh does not demonstrate that PacketCable in combination with Basturk (Ground III) discloses or renders obvious any challenged claim.

1. Dr. Houh does not demonstrate that a POSITA would have been motivated to combine PacketCable with Basturk to arrive at the claimed invention.

a. Basturk teaches away from modifying PacketCable with Basturk's transport protocol.

169. For the reasons explained above in Section X.A.2.a.1, *supra*, Basturk teaches away from modifying the application layer with its transport level protocol. Basturk teaches away from modifying PacketCable at the application layer for the same reasons it teaches away from modifying Handley at the application layer. *See* § X.A.1.a, *supra*.

b. Basturk's protocol for reliable transport is incompatible with PacketCable's use of RTP over UDP.

170. A POSITA would have understood that end-to-end delay caused by reliability features is undesirable for the transmission of real-time data such as that

carried by RTP. Indeed, Handley discusses how reliable protocols like TCP are generally not suitable for transporting real-time data using RTP, thus it seems to provide a solution for better transport of RTP over UDP. *See* § X.A.1.c, *supra*. PacketCable, like Handley, carries RTP over UDP and thus would be similarly sensitive to performance degradation caused by delays of having a reliable protocol. EX1007 at 20; § IX.C, *supra*. Accordingly, a POSITA would have understood that because Basturk’s protocol is a reliability protocol, it would be unsuitable for transporting PacketCable’s real-time RTP traffic for the same reasons that Handley teaches TCP is unsuitable. *See* § X.A.1.c, *supra*. Accordingly, a POSITA would have been discouraged and led away from modifying PacketCable’s implementation of RTP over UDP with Basturk’s reliable transport protocol.

c. A POSITA would not have been motivated to modify PacketCable with Basturk as suggested by Dr. Houh.

171. Dr. Houh asserts that “[a] POSITA would have been motivated to use Basturk’s ‘on-the-fly’ signature computation with PacketCable’s architecture framework” due to the purported known benefits, including reduced latency and faster transmission, and would have had a reasonable expectation of success in doing so. EX1003 ¶ 258, 263. I disagree.

172. As discussed above, one of Basturk’s requirements for enabling “on-the-fly” computation is that “[s]ignature computation starts before the retrieval of

connection context” for messages received by a receiver, which is enabled by using a style of MAC computation that can be started without knowing the key or other cryptographic information retrieved from context. EX1006 at 8, 13; *see* § IX.B.3, *supra*. This “on-the-fly” processing cannot be achieved using PacketCable’s security framework.

173. According to PacketCable’s security framework, “[t]he MAC computation spans the packet’s unencrypted header and encrypted payload.” EX1007 at 47. Thus, the MAC computation—whether the MMH algorithm of PacketCable or Basturk’s algorithm, cannot even begin to be processed on the transmitter until encryption of the encrypted payload happens. To perform the encryption, context retrieval must necessarily take place prior to the start of any MAC computation over the encrypted payload. This is so even with Basturk’s MAC algorithm (which requires no key to compute the hash portion of the MAC computation) because even the portion that requires no key is computed over the entire encrypted data. A context retrieval is necessary to retrieve the key used to encrypt the payload data. Thus, the basic security structure of the PacketCable design means “on-the-fly” cannot be achieved on the packet transmission side, and thus no purported benefits of “on-the-fly” can be realized.

174. A POSITA would have understood that “on-the-fly” is also not achievable for packet reception in PacketCable. A POSITA would have understood that PacketCable’s “*only*” specified MAC algorithm is incompatible with Basturk’s “on-the-fly” processing. EX1007 at 47. Although PacketCable says that its MAC algorithm can be negotiated, “the only one that is currently specified is MMH.” EX1007 at 47. In other words, the only MAC algorithm specified for PacketCable is a MAC based on the MMH algorithm of which a fast version was known at the time. EX2008 (Halevi).

175. A POSITA would have understood that a MAC based on MMH is incompatible with Basturk. Basturk requires the ability to start processing the message before the cryptographic key needs to be used because the claimed benefit is to reduce latency by performing some computation while waiting for context retrieval. Basturk uses this time to retrieve the key. However, a MAC based on the MMH is unable to start processing the message before retrieving the cryptographic key.

176. For example, the pseudocode for the implementation is described as follows:

A pseudo-code for MMH on such machine may be as follows

MMH(*msg*, *key*)

1. *SumHigh* = *SumLow* = 0
2. For *i* = 1 to *k*
3. load *msg*[*i*]
4. load *key*[*i*]
5. $\langle \textit{ProdHigh}, \textit{ProdLow} \rangle = \textit{msg}[i] * \textit{key}[i]$
6. $\textit{SumLow} = \textit{SumLow} + \textit{ProdLow}$
7. $\textit{SumHigh} = \textit{SumHigh} + \textit{ProdHigh} + \textit{carry}$
8. Reduce $\langle \textit{SumHigh}, \textit{SumLow} \rangle \bmod 2^{32} + 15$ and then $\bmod 2^{32}$

EX2008 at 9

Here we see there are two phases of the algorithm. The first is in the “For *i*= 1 to *k*” loop. The second component starting with “Reduce...” operates on the results of the first portion (i.e., *SumHigh* and *SumLow*). Note that the first phase requires access to both a portion of the message, e.g. *msg*[*i*] as well as a portion of the cryptographic key, e.g. *key*[*i*] for every block of the message *m*[1] to *m*[*k*]. Thus, to even begin processing the secure MAC algorithm based on MMH the cryptographic key is necessary.

177. As such, a context retrieval would be required before processing any portion of the message using the secure MAC based on MMH. Consequently, the delay associated with context retrieval would be unavoidable in PacketCable, meaning that “on-the-fly” cannot be achieved.

2. Dr. Houh fails to show that a POSITA would have found claim 1 obvious over PacketCable and Basturk.

178. For each of Elements [1.1], [1.2], and [1.3], Dr. Houh incorporates by reference his opinions regarding those claim elements from Grounds I-II. It is therefore my opinion, Dr. Houh has failed to show that Elements [1.1], [1.2], and [1.3] would have been obvious over the combination of PacketCable and Basturk for the same reasons I explained with respect to those Elements under Grounds I-II, *supra*, which are incorporated here by reference. *See* §§ X.A-B, *supra*.

a. [Element 1.3] “ascertaining, by the receiver, whether the receiver knows the transmitter based on the RTP packet level authentication data; and”

179. Dr. Houh alleges that Element [1.3] would have been obvious over the combination of PacketCable and Basturk, based on his opinions for Elements [1.2] and [1.3] under Grounds I and II. *See* EX1003 ¶ 267. But “*the* RTP packet level authentication data” of Element [1.3] refers to the authentication data that was inserted “at end of a whole RTP packet payload” per the requirements of Element [1.2]. For the reasons I explained above, Dr. Houh has not shown that it would have been obvious to a POSITA to combine PacketCable and Basturk to arrive at the requirements of Element and [1.2]. *See* §§ X.A.2 and X.B.1., *supra*. It is my opinion that Dr. Houh has not shown that Element [1.3] was disclosed or rendered obvious for the same reasons.

3. Independent Claims 25-28

Dr. Houh arguments for independent claims 25-28 rely on his arguments for Elements [1.1]-[1.3]. *See* EX1003 ¶¶292-319. Accordingly, Dr. Houh has failed to show that claims 25-28 would have been obvious over the combination of PacketCable and Basturk for the reasons explained above with respect to Elements [1.1]-[1.3]. *See* §§X.A.2 and X.B.1, *supra*.

4. Dr. Houh does not demonstrate that a POSITA would have found claims 22 and 24 obvious over PacketCable and Basturk.

- a. [22] The method as claimed in claim 21, wherein the receiver and transmitter are each disposed in at least one of a switching node and a switching installation.

180. For the same reasons that I explained for claim 22 of Ground II, Dr. Houh has not shown that the combination of PacketCable and Basturk discloses or renders obvious claim 24. *See* § X.B.3.a.

- b. [24] The method as claimed in claim 23, wherein the receiver and transmitter are each disposed in at least one of a switching node and a switching installation.

181. For the same reasons that I explained for claim 22 of Ground II, Dr. Houh has not shown that the combination of PacketCable and Basturk discloses or renders obvious claim 24. *See* § X.B.3.b.

5. All dependent claims are patentable over PacketCable in combination with Basturk.

182. Claims 2-24 depend on claim 1. Claim 1 is patentable over PacketCable in combination with Basturk for the reasons explained above. Accordingly, by their dependency on claim 1, claims 2-24 are patentable over PacketCable in combination with Basturk.

D. Dr. Houh does not demonstrate that that Handley in combination with Basturk and PacketCable (Ground IV) discloses or renders obvious claims 22 and 24.

1. [22] The method as claimed in claim 21, wherein the receiver and transmitter are each disposed in at least one of a switching node and a switching installation.

183. For the same reasons that I explained for claim 22 of Grounds II and III, Dr. Houh has not shown that the combination of PacketCable, Handley, and Basturk discloses or renders obvious claim 22.

2. [24] The method as claimed in claim 23, wherein the receiver and transmitter are each disposed in at least one of a switching node and a switching installation.

184. For the same reasons that I explained for claim 22 of Grounds II and III, Dr. Houh has not shown that the combination of PacketCable, Handley, and Basturk discloses or renders obvious claim 24.

185. 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.

Executed on this 13th day of March 2024.

A handwritten signature in cursive script, reading "Stuart Stubblebine", positioned above a horizontal line.

Stuart G. Stubblebine

APPENDIX A

Materials Considered

- U.S. Patent No. 7,266,682 (EX1001)
- File History of U.S. Patent No. 7,266,682 (EX1002)
- Declaration of Dr. Henry Houh, Ph.D. in Support of Petition for *Inter Partes* Review of U.S. Patent No. 7,266,682 (EX1003)
- M. Handley, J. Crowcroft, & C. Bormann, Very Large Conferences on the Internet: The Internet Multimedia Conferencing Architecture and the MBONE (EX1005)
- Erol Basturk, Mihir Bellare, Chee-Seng Chow, & Roch Guérin, Secure Transport Protocols for High-Speed Networks (EX1006)
- PacketCable 1.0 Architecture Framework Technical Report (EX1007)
- RTP: A Transport Protocol for Real-Time Application, IETF RFC 1889 (EX1008)
- Andrew S. Tanenbaum, Computer Networks (3rd ed. 1996) (EX1009)
- Carrier Sense Multiple Access with Collision Detection, IEEE Standard 802.3, IEEE Standards for Local Area Networks (EX1014)
- Patent Owner's Preliminary Response (Paper 11)
- Transmission Control Protocol: RFC 793 (<https://www.rfc-editor.org/rfc/pdf/rfc793.txt.pdf>) (EX2006)
- User Datagram Protocol: RFC 768 (<https://www.rfc-editor.org/rfc/pdf/rfc768.txt.pdf>) (EX2007)
- Shai Halevi and Hugo Krawczyk, MMH: Software Message Authentication in the Gbit/Second Rates (EX2008)