

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

AMERICAN AIRLINES, INC.

and

SOUTHWEST AIRLINES CO.

Petitioners

v.

INTELLECTUAL VENTURES I LLC

Patent Owner

IPR2025-00786

U.S. Patent No. 7,949,785

**DECLARATION OF EREZ ZADOK IN SUPPORT OF PETITION FOR
INTER PARTES REVIEW OF
U.S. PATENT NO. 7,949,785**

Mail Stop Patent Board
Patent Trial and Appeal Board
U.S. Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-145

Table of Contents

I.	ASSIGNMENT.....	1
II.	BACKGROUND AND QUALIFICATIONS.....	1
III.	MATERIALS CONSIDERED.....	12
IV.	LEVEL OF ORDINARY SKILL IN THE ART.....	14
V.	RELEVANT LEGAL STANDARDS.....	15
VI.	SUMMARY OF OPINIONS.....	20
VII.	TECHNOLOGY OVERVIEW.....	20
	A. General Computer Operations.....	20
	B. Networking.....	22
	1. Networking Overview.....	22
	2. Network Protocols	27
	3. The Domain Name System, Hostnames, and IP Addresses	40
	4. IP Security.....	52
	C. Virtualization.....	53
VIII.	THE '785 PATENT.....	57
	A. Overview	57
	B. Prosecution History	65
IX.	THE CLAIMS	67
X.	OVERVIEW OF THE PRIMARY PRIOR ART REFERENCES.....	72
	A. Caronni I.....	72
	B. Caronni II	77
	C. Hipp	80

D.	RFC-1383	83
XI.	CLAIM CONSTRUCTION	86
A.	Claim Construction Generally.....	86
B.	Claim Construction Means Plus Function Terms	87
C.	All Claims: “virtual network address”	88
D.	Claim 30 (“register module”).....	88
E.	Claims 35-37, 77-78: “join module”	90
F.	Other Claims.....	93
XII.	GROUND 1: Claims 1, 30, 35, 36, 37, 38, 48, 62, 75, 77, and 78 are obvious in view of Caronni I in view of Caronni II and in further view of Hipp.....	94
A.	The Independent Claims.....	94
1.	virtual network system or manager.....	94
2.	a virtual network defined by a domain name having an associated public network address	98
a.	a network interface	99
b.	virtual network defined by a domain name having an associated public network address.....	104
3.	registration/distribution of virtual network address to each device which uniquely identifies the device	109
a.	agent associated with a device.....	111
b.	register module/registration request	112
c.	distribute a virtual address	116
d.	identified by a unique virtual network address	117
e.	not directly routable via a public network.....	120
f.	associated private network address	123

g.	the virtual network being accessible by devices in the user set and devices outside of the user set	126
4.	route director/routing encapsulated packets	130
a.	route director.....	131
b.	encapsulated packets.....	133
5.	DNS server, request, and responses.....	138
a.	DNS server for the virtual network.	139
b.	DNS request.....	143
c.	DNS response	145
	(i) public network address of a route director/virtual network manager	146
	(ii) network address of the destination device	148
	(iii) virtual network address that corresponds to the destination device	150
B.	Motivation To Combine Caronni I, Caronni II, and Hipp	150
1.	Virtual network	153
2.	Registration process	154
3.	Packet encapsulation	155
C.	The Dependent Claims	162
1.	Claims 35, 36, 37, 77, and 78 are obvious and unpatentable over the combination of Caronni I, Caronni II, and Hipp	162
a.	Claim 35: The virtual network manager of claim 30 further comprising a join module configured to receive a join request from the agent associated with the device to indicate that the device is connected for data communication within the virtual network, the join module further configured to receive a leave request from	

	the agent associated with the device to indicate that the device will be disconnected from data communication within the virtual network.....	162
b.	Claim 77: The system of claim 75 wherein the virtual network manager includes a member join module.....	162
c.	Claim 36: The virtual network manager of claim 35 wherein the join module is further configured to provide virtual network addresses to the devices that are registered in the virtual network.....	164
d.	Claim 78: The system of claim 77 wherein the member join module provides a virtual network address to a device that is registered as a member of the network...	164
e.	Claim 37: The virtual network manager of claim 35 wherein the join module is further configured to maintain data to associate a virtual network address with a device in the virtual network.....	165
XIII.	GROUND 2: Claims 1, 30, 35, 36, 37, 38, 48, 62, 75, 77, and 78 are obvious in view of Caronni I in view of Caronni II and in further view of RFC-1383	166
A.	The Independent Claims.....	166
1.	preamble: virtual network system or manager.....	166
2.	a virtual network defined by a domain name having an associated public network address	166
3.	registration/distribution of virtual network address to each device which uniquely identifies the device	167
4.	route director/routing encapsulated packets	167
5.	DNS server, request, and responses.....	167
a.	DNS server for the virtual network.	168
b.	DNS request.....	168

c.	DNS response	169
B.	Motivation To Combine Caronni I, Caronni II, and RFC-1383	170
1.	Virtual network	172
2.	Registration process	173
3.	Packet encapsulation	174
C.	The Dependent Claims	182
1.	Claims 35, 36, 37, 77, and 78 are obvious and unpatentable over the combination of Caronni I, Caronni II, and RFC-1383	182
a.	Claim 35: The virtual network manager of claim 30 further comprising a join module configured to receive a join request from the agent associated with the device to indicate that the device is connected for data communication within the virtual network, the join module further configured to receive a leave request from the agent associated with the device to indicate that the device will be disconnected from data communication within the virtual network.....	183
b.	Claim 77: The system of claim 75 wherein the virtual network manager includes a member join module.....	183
c.	Claim 36: The virtual network manager of claim 35 wherein the join module is further configured to provide virtual network addresses to the devices that are registered in the virtual network.....	185
d.	Claim 78: The system of claim 77 wherein the member join module provides a virtual network address to a device that is registered as a member of the network...	185
e.	Claim 37: The virtual network manager of claim 35 wherein the join module is further configured to maintain data to associate a virtual network address with a device in the virtual network.....	186

XIV. SECONDARY CONSIDERATIONS	186
XV. CONCLUSION.....	187

I, Erez Zadok, Ph.D., declare that:

I. ASSIGNMENT

1. I have been retained by American Airlines, Inc. and Southwest Airlines Co. (“Petitioners”) as an independent expert consultant in this proceeding before the United States Patent and Trademark Office (“PTO”).

2. My consulting company, Zadoks Consulting Services, is being compensated for my time at my standard consulting rate. I am also being reimbursed for expenses that I may incur during the course of this work.

3. My compensation is in no way contingent on the nature of my findings, the presentation of my findings in testimony, or the outcome of this or any other proceeding. I have no other interest in this proceeding.

4. I have been asked to consider whether certain references disclose or suggest the features recited in the claims of U.S. Patent No. 7,949,785 (the ’785 patent) (EX1001). Where appropriate, I refer to exhibits I understand will be attached to the petition for inter partes review of the ’785 patent (the “Petition”). My opinions are set forth below.

II. BACKGROUND AND QUALIFICATIONS

5. I am a Professor in the Computer Science Department at Stony Brook University (part of the State University of New York (“SUNY”) system). I direct the Filesystems and Storage Lab (FSL) at Stony Brook’s Computer Science

Department. My research interests include file systems and storage systems, operating systems, transactional systems including database technologies, information technology and system administration, security/privacy and information assurance, networking, energy efficiency, performance and benchmarking, virtualization, cloud systems, compilers, applied machine learning, and software engineering.

6. I studied at a professional high school in Israel, focusing on electrical engineering (“EE”), and graduated in 1982. I spent one more year at the high school’s college division, receiving a special Certified Technician’s degree in EE. I then went on to serve in the Israeli Defense Forces for three years (1983–1986). I received my Bachelor of Science degree in computer science (“CS”) in 1991, my Master’s degree in CS in 1994, and my PhD in CS in 2001—all from Columbia University in New York.

7. When I began my undergraduate studies at Columbia University, I also started working as a student assistant in the various campus-wide computer labs, eventually becoming an assistant to the head labs manager, who was managing all public computer labs on campus. During that time, I also became more involved with research within the CS Department at Columbia University, conducting research on operating systems, file and storage systems, distributed and networked systems, security, and other topics. I also assisted the CS department’s computer

administrators in managing the department's computers, which included storage, IT, networking, and cyber-security related duties.

8. In 1991, I joined Columbia University's CS department as a full-time systems administrator, studying towards my MS degree part-time. My MS thesis topic is related to file system reliability, fault tolerance, replication, and failover in mobile networked storage systems using file virtualization. My main duties as a systems administrator involved installing, configuring, and managing many networked servers, proxies, and desktops running several operating systems, as well as network devices setup; this included many software and hardware upgrades, device upgrades, and BIOS firmware/chipset updates/upgrades. My duties also included ensuring reliable, secure, authenticated access to networked systems/storage and licensed software, as well as software updates, security and bug fixes. Examples of servers and their protocols included email (SMTP), file transfer (FTP), domain names (DNS), network file systems (NFS), network news systems (NNTP), and Web (HTTP).

9. In 1994, I left my systems administrator position to pursue my doctoral studies at Columbia University. My PhD thesis topic was on versatile file system development using stackable (virtualized) file systems, with examples in the fields of security and encryption, efficiency, reliability, and failover. I continued to work part-time as a systems administrator at the CS department, and eventually I was

asked to serve as manager to the entire information technology (“IT”) staff. From 1991 to 2001, I was also a member of the faculty-level Facilities Committee that oversaw all IT operations at the CS department.

10. As part of my PhD studies at Columbia, I collaborated on projects to develop advanced AI-like techniques to detect previously unknown viruses (a.k.a. “zero-day malware”), using data mining and rule-based detection. This work led to several highly cited papers (over 1,700 citations for one of the papers alone), and two patents. I also became a Teaching Assistant (TA) for a first-ever Computer Security course given at Columbia University’s CS department with Dr. Matt Blaze as instructor.

11. From 1990 to 1998, I consulted for SOS Corporation and HydraWEB Technologies, as a systems administrator and programmer, managing data storage use and backup/restore duties, databases, web servers, as well as information assurance and cyber-security (*e.g.*, malware protection, software licensing). From 1994 to 2000, I led projects at HydraWEB Technologies, and then became the Director of Software Development—overseeing the development of several products and appliances such as stateful firewalls and HTTP load-balancers, utilizing network-virtualization and high-availability techniques. From 2009 to 2019, I have consulted for Packet General Networks, a startup specializing in secure, virtualized, network storage and applications’ data security in the cloud.

12. In 2001, I joined the faculty of Stony Brook University, a position I have held since that time. In 2002, I joined the Operations Committee, which oversees the IT operations of the CS department at Stony Brook University. From 2006 to 2010, I was the Director of IT Operations of the CS department. My day-to-day duties included setting policies regarding computing, hiring and training new staff, assisting any staff with topics of my specialty, defining requirements for new software/hardware, and purchasing. From 2010 to 2015, I had served as the Co-Chair to the Operations Committee. From 2016 to 2019, I oversaw the IT Operations as the Chair of the Operations Committee. A significant component of these duties included defining and helping implement policies for data management, to ensure the security of users and their data, and data reliability and availability, while minimizing the inconvenience and performance impact to users. I personally helped setup and maintain an initial virtual-host infrastructure in the department. Since late 2019, I've been a member of the department's Executive Committee that also oversees all IT operations.

13. In 2017, I became the department's Graduate Academic Adviser, advising all Master students (over 400 annually on average) and many other graduate students on an assortment of academic matters. In August 2024, I took over as the department's Graduate Program Director, overseeing the entire graduate CS program (700-800 students annually on average).

14. Since 2001, I have personally configured and managed my own research lab's network. This includes setting up and configuring multiple storage systems (*e.g.*, NFS, CIFS/SMB, NAS), virtual and physical environments, applications such as database (*e.g.*, MySQL, Postgresql), Web servers (*e.g.*, Apache), and mail servers; user access control (*e.g.*, NIS, LDAP), backups and restores, snapshot policies, and more. I've personally installed, configured, changed, replaced parts, and upgraded components in numerous devices including mobile devices, laptops, desktops, and servers, both physical and virtual.

15. Since 1995, I have taught courses on operating systems, storage and file systems, advanced systems programming in Unix/C, systems administration, data structures, data/software security, and more. My courses often use storage, file systems, distributed systems, and system/network security as key teaching principles and practical examples for assignments and projects. I have taught these concepts and techniques to my students, both to my direct advisees as well as in my courses. For example, in my graduate Operating Systems course, I often cover Linux's kernel mechanisms to protect users, applications, and data files, virtual file systems, as well as distributed storage systems (*e.g.*, NFS). And in the System Administration undergraduate course, I covered many topics such as networking, storage, backups, and configuring complex applications such as mail, web, and database servers.

16. My research often investigates computer systems from many angles: security, efficiency, energy use, scalability, reliability, portability, survivability, usability, ease-of-use, versatility, flexibility, and more. My research gives special attention to balancing five often-conflicting aspects of computer systems: performance, reliability, energy use, security, and ease-of-use.

17. Since joining Stony Brook University in 2001, my group in the File-systems and Storage Lab (FSL) has developed many file systems and operating system extensions; examples include a highly-secure cryptographic file system, a portable copy-on-write (COW) versioning file system, a tracing file system useful to detect intrusions, a replaying file system useful for forensics, a snapshotting and sandboxing file system, a namespace unification file system (that uses stackable, virtualized, file-based COW), an anti-virus file system, an integrity-checking file system, a load balancing and replication/mirroring file system, network file system extensions for security and performance, distributed secure cloud-based storage systems, transactional key-value stores and file systems, OS-level embedded databases, a compiler to convert user-level C code to in-kernel efficient yet safe code, GCC plugins, stackable file system templates, and a Web-based backup system. Many of these projects used one form of virtualization or another (storage, network, host, etc.). I continue to maintain and release newer versions of some of these file systems and software.

18. I have published over 120 refereed publications (in ACM, IEEE, USENIX, and more). To date, my publications have been cited more than 10,000 times (as per Google Scholar as of March 30, 2025). My papers cover a wide range of related technologies such as file systems, storage systems, transactional systems, security, clouds and virtualization, performance benchmarking and optimization, energy efficiency, system administration, web systems, and more. I also published a book titled “Linux NFS and Automounter Administration” (Sybex, 2001), covering systems administration topics related to network storage and data security.

19. Some of my research has led to public software releases that have been used worldwide. I have publicly maintained the Amd Berkeley Automounter in a package called “am-utils” since 1992; this software helps administrators manage the multitude of file system mounts on dozens of different Unix systems, especially helping to automate access to multiple NFS/NAS storage volumes. Since 1997, I have maintained and released several stackable (virtualized) file system software projects for Linux, FreeBSD, and/or Sun Solaris, in a package called FiST. One of my stackable file system encryption projects, called Cryptfs, became the basis for IBM’s public release of eCryptfs, now part of Linux. Packet General Networks, for whom I have provided consulting services between 2009 and 2019, licensed another encryption file system called Ncryptfs. Another popular file system released in 2003, called Unionfs, offers virtual namespace unification, transparent shadow

copying (a.k.a. copy-on-write or COW), file system snapshotting (*e.g.*, useful for forensics and disaster recovery), and the ability to save disk space by sharing a read-only copy of data among several computers, among other features.

20. My research and teaching make extensive use of data security features. For example, each time I taught the graduate operating system course, the first homework assignment includes the creation of a new system call that performs new or added functionality, often for encrypting a file or verifying its integrity; many of my other assignments cover topics of user/process access control, anti-virus filtering, and more. Since 2001, over 1,000 graduate students were exposed to these principles directly through my teaching and research at Stony Brook University.

21. Moreover, in an undergraduate course titled “Advanced Systems Programming in Unix/C,” I cover many topics of system security and vulnerabilities, such as the structure of UNIX processes, and memory segments such as the heap and stack. This course covers details of several hundred Linux system calls. Often, the first assignment for this course is to develop a tool to encrypt/decrypt files using advanced ciphers, use digital signatures to certify the cipher keys used, and reliably recover files in case of failures. Since 2001, several hundred undergraduate students were exposed to these principles directly through my teaching and research at Stony Brook University.

22. In another undergraduate course, System Administration, I taught network configuration, security, and storage configuration and reliability. In a special topics course on Storage Systems, I covered many topics such as data deduplication, RAID, transactional storage, storage hardware including modern Flash based ones, virtual storage, backup/restore, snapshots and continuous data protection (CDP), NAS and SAN, and NFS.

23. Overall, in addition to the aforementioned experience, my technical experience relevant to the challenged patent(s) at the time of the alleged invention(s) included the following. I configured, used, researched, developed, programmed, and/or used these technologies and systems: network services and protocols including HTTP, DNS, DHCP; network security systems including firewalls, VPN, and NAT; and virtualization at the storage, file system, and network layers; configured hostname-to-address resolution via local /etc/hosts files; and I designed and developed virtualized file systems and networked one, including as part of my Masters and PhD dissertations.

24. My research has been supported by many federal and state grants as well as industry awards, including an NSF CAREER award, two IBM Faculty awards, two NetApp Faculty awards, a Western Digital award, a Facebook award, several Dell-EMC awards, and several equipment gifts. I received the 2008 SUNY Chancellor's Excellence in Teaching award, and the 2022 SUNY Chancellor's

Award for Excellence in Scholarship and Creative Activities (both awards can be given only once in a lifetime). In 2021 I was named an ACM Distinguished Member for “Outstanding Scientific Contributions to Computing.”

25. My service record to the community includes serving as the co-chair for the USENIX Annual Technical Conference in 2020 (ATC’20); serving as the co-chair for USENIX File and Storage Technologies (FAST’15) in 2015 and on the FAST Conference Steering Committee from 2015 to 2023; serving on the ACM HotStorage Steering Committee since 2021; and serving as the co-chair in 2012 and on the Steering Committee of the ACM SYSTOR conference since 2012. I have served as an Associate Editor to the ACM Transactions on Storage (TOS) journal from 2009 to 2022; in 2022, I was named the Editor-in-Chief for ACM’s TOS journal.

26. I am a named inventor on four patents, two titled “Systems and Methods for Detection of New Malicious Executables” (U.S. Patent No. 7,487,544, issued February 3, 2009; and U.S. Patent No. 7,979,907, issued July 12, 2011); and two more titled “Multi-Tier Caching,” (U.S. Patent No. 9,355,109, issued May 31, 2016; and U.S. Patent 9,959,279, issued May 1, 2018).

27. I have been disclosed as a testifying expert in 18 cases (including *inter partes* review (IPR) proceedings) in the past four years. I have been deposed 13 times and testified in trial twice.

III. MATERIALS CONSIDERED

28. The opinions contained in this declaration are based on the documents I reviewed, my professional judgment, as well as my education, experience, and knowledge regarding electrical engineering, computer science, video and audio and communications sciences and technologies.

29. In forming my opinions expressed in this declaration, I reviewed the following materials:

Exhibit No.	Description
1001	U.S. Patent No. 7,949,785 (“’785 Patent”)
1002	File History for U.S. Patent No. 7,949,785 (Appl. No. 10/403,818)
1003	U.S. Patent No. 6,970,941 (“Caronni I”)
1004	U.S. Patent No. 7,814,228 (“Caronni II”)
1005	U.S. Patent No. 6,766,371 (“Hipp”)
1006	Network Working Group Request for Comment (RFC): 1383, entitled <i>An Experiment in DNS Based IP Routing</i> , by C. Huitema, 1992 (RFC-1383)
1007	Intellectual Ventures I LLC <i>et al.</i> v. American Airlines, Inc., No. 4:24-cv-980 (E.D. Tex.) – Complaint
1008	Intellectual Ventures I LLC <i>et al.</i> v. Southwest Airlines Co., No. 7:24-cv-277 (W.D. Tex.) – Complaint
1009	Intellectual Ventures v. Liberty Mutual, No. 23-cv-525 (E.D. Tex.) – Complaint
1010	Intellectual Ventures v. Liberty Mutual, No. 23-cv-525 (E.D. Tex.) – Motion to Dismiss – Voluntary Dismissal (FRCP 41(a))
1011	Reserved
1012	Curriculum Vitae of Dr. Erez Zadok
1013	Reserved
1014	Microsoft Computer Dictionary, 3rd ed., 1997, excerpts (Microsoft Computer Dictionary)
1015	Re-Exam 90/0019,519 filed on May 22, 2024 (’785 re-exam)
1016	Reserved

Exhibit No.	Description
1017	Andrew S. Tanenbaum, <i>Computer Networks</i> , 2nd ed., 1988, excerpts (Tanenbaum)
1018	W. Richard Stevens, <i>TCP/IP Illustrated Volume 1, The Protocols</i> , 1994, excerpts (Stevens)
1019	William R. Cheswick & Steven M. Bellovin, <i>Firewalls and Internet Security, Repelling the Wily Hacker</i> , 1994, excerpts (Cheswick)
1020	Domain Names - Concepts and Facilities, RFC-1034, IETF, 1987, https://www.rfc-editor.org/rfc/pdf/rfc1034.txt.pdf
1021	Domain Names - Implementation and Specification, RFC-1035, IETF, 1987, https://www.rfc-editor.org/rfc/pdf/rfc1035.txt.pdf
1022	IP Encapsulating Security Payload (ESP), RFC-2406, IETF, 1998, https://www.rfc-editor.org/rfc/pdf/rfc2406.txt.pdf
1023	Dynamic Host Configuration Protocol, RFC-2131, IETF, 1997, https://www.rfc-editor.org/rfc/pdf/rfc2131.txt.pdf
1024	Linux DNS Server Administration, Craig Hunt, 2000, excerpts (Hunt)
1025	Claim Chart for '785 Patent, <i>Intellectual Ventures I LLC et al. v. American Airlines, Inc.</i> , No. 4:24-cv-980-ALM (E.D. Tex. Nov. 2, 2024) (Exhibit 10 to Complaint, Dkt. 1-11) ('785 claim chart)
1026	Claim Chart for '785 Patent, <i>Intellectual Ventures I LLC et al. v. Southwest Airlines Co.</i> , No. 7:24-cv-277 (W.D. Tex. Nov. 2, 2024) (Exhibit 3 to Intellectual Ventures I LLC's and Intellectual Ventures II LLC's Preliminary Infringement Contentions) ('785 claim chart)
1027	Network Working Group Request for Comment (RFC): 2401, entitled <i>Security Architecture for the Internet Protocol</i> , by S. Kent <i>et al.</i> , 1998 (RFC-2401).
1028	Andrew S. Tanenbaum, <i>Computer Networks</i> , 4th ed., 2003, excerpts (Tanenbaum2)

30. I have also reviewed any other materials I refer to in this declaration in support of my opinions.

31. All of the opinions contained in this declaration are based on the documents I reviewed and my knowledge, experience, and professional judgment. My opinions have also been guided by my appreciation of how a person of ordinary

skill in the art would have understood the claims and the specification of the '785 patent at the time of the alleged invention, which I have been asked to initially consider as no earlier than March 31, 2003. My opinions reflect how one of ordinary skill in the art would have understood the '785 patent, the prior art to the patent, and the state of the art at the time of the alleged invention.

32. Based on my experience and expertise, it is my opinion that certain references disclose and/or suggest all the features recited in claims 1, 30, 35-38, 48, 62, 75, and 77-78 ("challenged claims") of the '785 patent, as I discuss in detail below.

IV. LEVEL OF ORDINARY SKILL IN THE ART

33. I have been informed and understand that, in the context of an invalidity analysis, a person having ordinary skill in the art ("POSITA") is a hypothetical person who looks to prior art at the time of the invention. I further understand that the factors that may be considered in determining the level of ordinary skill include: (1) the problems encountered in the art; (2) the prior art solutions to the problems encountered in the art; (3) the rapidity of innovation; (4) the sophistication of the technology; and (5) the education level of active workers in the field. I understand that these factors need not all be considered for the analysis and that one or more of these factors may control.

34. I was asked to provide my opinion on the level of one of ordinary skill in the art with respect to the alleged invention of the '785 patent as of March 31, 2003. In my opinion a POSITA would have had a bachelor's degree in computer science, computer engineering, or an equivalent degree. A POSITA would also have had approximately two years of experience working in the fields of networking including network virtualization, security, and management. Additional experience might substitute for less education and vice versa.

35. As of March 31, 2003, I met, and in fact exceeded, the qualifications of a person of ordinary skill in the art. To be clear, all of my opinions in this declaration are from the perspective of one of ordinary skill in the art as I have defined it here during the relevant timeframe.

V. RELEVANT LEGAL STANDARDS

36. I am not an attorney and offer no legal opinions, but in the course of my work, I have had experience studying and analyzing patents and patent claims from the perspective of a person skilled in the art.

37. For the purposes of this declaration, I have been informed about certain aspects of the law that are relevant to forming my opinions. My understanding of the law is as follows.

38. Petitioners' counsel has informed me and I understand that a published nonprovisional application is entitled to claim priority to the filing date of its

provisional application (as opposed to the filing date of the published non-provisional application) if (1) the provisional application discloses the portions of the published non-provisional application relied upon as prior art, and (2) at least one claim of the published non-provisional claim is supported by the provisional application, *i.e.*, the provisional application must disclose the subject matter of the claim and enable a person of ordinary skill in the art to make and use the claim.

39. Petitioners' counsel has informed me and I understand that for the prior art to inherently disclose a claimed limitation, the prior art need not expressly disclose the limitation, so long as the claimed limitation necessarily flows from a disclosure in the prior art. I also understand that it is acceptable to examine evidence outside the prior art reference (extrinsic evidence) in determining whether a feature, while not expressly discussed in the reference, is necessarily present in that reference.

40. Petitioners' counsel has informed me and I understand that a patent claim can be considered to have been obvious to a person having ordinary skill in the art at the time the application was filed. I am informed that this means that, even if all of the requirements of a claim are not found in a single prior art reference, the claim is not patentable if the differences between the subject matter in the prior art and the subject matter in the claim would have been obvious to a person having ordinary skill in the art at the time of the invention.

41. I have been informed by Petitioners' counsel and I understand that a determination of whether a claim would have been obvious should be based upon several factors, including, among others:

- the level of ordinary skill in the art at the time the application was filed;
- the scope and content of the prior art; and
- what differences, if any, existed between the claimed invention and the prior art.

42. I have been informed by Petitioners' counsel and I understand that a single reference can render a patent claim obvious if any differences between that reference and the claims would have been obvious to a person having ordinary skill in the art. Alternatively, I understand that the teachings of two or more references may be combined in the same way as disclosed in the claims, if such a combination would have been obvious to one having ordinary skill in the art. In determining whether a combination based on either a single reference or multiple references would have been obvious, it is appropriate to consider, among other factors:

- whether the teachings of the prior art references disclose known concepts combined in familiar ways, and when combined, would yield predictable results;
- whether a person having ordinary skill in the art could implement a predictable variation, and would see the benefit of doing so;

- whether the claimed elements represent one of a limited number of known design choices, and would have a reasonable expectation of success by those skilled in the art;
- whether a person having ordinary skill would have recognized a reason to combine known elements in the manner described in the claim;
- whether there is some teaching or suggestion in the prior art to make the modification or combination of elements claimed in the patent; and
- whether the innovation applies a known technique that had been used to improve a similar device or method in a similar way.

43. I have been informed by Petitioners' counsel and I understand that one of ordinary skill in the art has ordinary creativity and is not an automaton. Petitioners' counsel has also informed me that in considering obviousness, obviousness may not be determined using the benefit of hindsight, including hindsight derived from the patent being considered.

44. I further understand that a hypothetical person of ordinary skill in the art is assumed to know and to have all relevant prior art in the field of endeavor covered by the patent at issue. I have been informed that there are two criteria for determining whether prior art is analogous and thus can be considered prior art: (1) whether the art is from the same field of endeavor, regardless of the problem addressed, and (2) if the reference is not within the field of the patentee's endeavor,

whether the reference still is reasonably pertinent to the particular problem with which the patentee is involved. I have also been informed that the field of endeavor of a patent is not limited to the specific point of novelty, the narrowest possible conception of the field, or the particular focus within a given field. I have also been informed that a reference is reasonably pertinent if, even though it may be in a different field from that of the patentee's endeavor, it is one which, because of the matter with which it deals, logically would have commended itself to a patentee's attention in considering their problem.

45. I further understand that, in making a determination as to whether the claimed invention would have been obvious to a person of ordinary skill, the Board may consider certain objective factors if they are present, such as: commercial success of products practicing the claimed invention; long-felt but unsolved need; teaching away; unexpected results; copying; and praise by others in the field. These factors are generally referred to as "secondary considerations" or "objective indicia" of nonobviousness. I understand, however, that for such objective evidence to be relevant to the obviousness of a claim, there must be a causal relationship (called a "nexus") between the claim and the evidence, and that this nexus must be based on a novel element of the claim rather than something in the prior art. I also understand that even when they are present, secondary considerations may be unable to

overcome primary evidence of obviousness (such as motivation to combine with predictable results) that is sufficiently strong.

VI. SUMMARY OF OPINIONS

46. This Declaration explains the conclusions I have formed based on my analysis. These conclusions are summarized in the table below:

Ground	Basis	Relied-On References	Claim(s)
1	Obviousness	Caronni I, Caronni II, and Hipp	1, 30, 35, 36, 37, 38, 48, 62, 75, 77, and 78
2	Obviousness	Caronni I, Caronni II, and RFC-1383	1, 30, 35, 36, 37, 38, 48, 62, 75, 77, and 78

VII. TECHNOLOGY OVERVIEW

47. In this section, I provide an overview of the background art as understood by a person of ordinary skill in the art as of March 31, 2003.

A. General Computer Operations

48. Computers are hardware and software devices that execute instructions. The hardware often includes a central processing unit (CPU) or other microprocessors, typically called “processor,” volatile memory (*e.g.*, RAM (random-access memory), or DRAM (dynamic random-access memory)), non-volatile (persistent) storage (*e.g.*, a hard disk or flash drive), and input/output (I/O) devices (*e.g.*, network interface card, mouse, keyboard, display). RAM, hard disks,

and flash drives are considered, generally, “storage” or “memory” devices because they can store and hold bits of information (whether persistently or not). Computers can have multiple storage devices. Processors are coupled to main memory using a memory bus and coupled to peripherals such as hard disks and network interface cards using one or more I/O busses.

49. Memory and storage devices have considerable differences in their speeds, capacities, and costs. Non-volatile storage devices such as hard disks have the largest capacity, are least expensive per gigabyte, and are slowest (typically operating at millisecond speeds). Volatile DRAM devices are smaller, more expensive, and faster than hard disks (typically operating at microsecond speeds). CPUs’ own memory caches and operations are the smallest, most expensive, and fastest (typically operating at nanosecond speeds). Overall, CPUs are about 1,000× faster than DRAM, which in turn is about 1,000× faster than hard disks.

50. Computer programs are often written in a high-level human-readable language (*e.g.*, C, C++, Java, Perl, PHP, Python), then translated using a compiler or script processor to machine instructions understood by the CPU (*e.g.*, Intel or AMD processor). Machine instructions—generally called “software”—are stored in files on persistent media (*e.g.*, HDD or ROM), loaded into DRAM, and then loaded into the CPU where they can be executed. All modern computers operate in this manner, regardless of their purpose: a small laptop, a server in a data center, a

mainframe computer, a personal workstation, a proxy device, a gateway, a firewall, a router, an appliance, a virtual machine, a handheld or mobile device, smart card, SD card, etc. Computer systems may run multiple software programs, from background servers and daemons to user applications.

51. Users (including administrators) can login to one or more computers (*e.g.*, desktop, laptop, server) and execute one or more applications (*e.g.*, word processing, web browsing, backup tool). Some computers offer external interfaces and APIs to communicate with certain devices and especially peripherals. For example, both a USB storage device as well as a printer use a well-defined “protocol” to allow users to communicate with them through the main computer.

52. “IPC” or “interprocess communication” is “[t]he ability of one task or process to communicate with another in a multi-tasking operating system. Common methods include pipes, semaphores, shared memory, queues, signals, and mailboxes. *Acronym: IPC.*” EX1014 (Microsoft Computer Dictionary) at 261, 264.

B. Networking

1. Networking Overview

53. A Local Area Network (LAN) is typically a network at a local site where network latencies are short and physical distances between computers are small. A LAN and its computers are often protected from the rest of the Internet using a firewall. EX1014 (Microsoft Computer Dictionary) at 276 (“**LAN** *n.*

Acronym for **local area network**. A group of computers and other devices dispersed over a relatively limited area and connected by a communications link that enables any device to interact with any other on the network. LANs commonly include microcomputers and shared resources such as laser printers and large hard disks. The devices on a LAN are known as nodes, and the nodes are connected by cables through which messages are transmitted. *See also* baseband network, broadband network, bus network, collision detection, communications protocol, contention, CSMA/CD, network, ring network, star network, token bus network, token passing, token ring network. *Compare* wide area network.”).

54. Another form of a local area network is one that operates inside a single computer. A “localhost” is “[t]he name that is used to represent the same computer on which a TCP/IP message originates. An IP packet sent to localhost has the IP address 127.0.0.1 and does not actually go out to the Internet. *See also* IP address, packet (definition 1), TCP/IP.” EX1014 (Microsoft Computer Dictionary) at 287. Often, when users/processes want to communicate with other users/processes, using TCP/IP communications on the same machine, they can use the localhost address: it is considered a trusted, high-speed internal network.

55. Conversely, a Wide-Area Network (WAN) refers to computers across the entire Internet, which are often distant from each other, exhibit longer latencies, and afford little protection. EX1014 (Microsoft Computer Dictionary) at 507 (“**wide**

area network *n.* A communications network that connects geographically separated areas. *Acronym: WAN.*”).

56. A firewall typically intercepts all traffic between computers in a LAN and the WAN, monitors and controls access, and can filter out undesired traffic. Often, LAN computers are far more trusted than WAN computers; therefore, a firewall can be said to separate the trusted network (LAN or localhost) from the untrusted network (WAN). EX1014 (Microsoft Computer Dictionary) at 197 (“**firewall** *n.* A security system intended to protect an organization’s network against external threats, such as hackers, coming from another network, such as the Internet. A firewall prevents computers in the organization’s network from communicating directly with computers external to the network and vice versa. Instead, all communication is routed through a proxy server outside of the organization’s network, and the proxy server decides whether it is safe to let a particular message or file pass through to the organization’s network.”).

57. A server is a computer or software that provides a service to other computers, called clients. There are many servers on the Internet, for example, Web servers and Mail servers. EX1014 (Microsoft Computer Dictionary) at 430 (“**server** *n.* 1. On a local area network (LAN), a computer running administrative software that controls access to the network and its resources, such as printers and disk drives, and provides resources to computers functioning as workstations on the network. 2.

On the Internet or other network, a computer or program that responds to commands from a client. For example, a file server may contain an archive of data or program files; when a client submits a request for a file, the server transfers a copy of the file to the client. *See also* client/server architecture. *Compare* client (definition 3).”

58. A client is typically an end-point computer or software, sometimes used by a user, that communicates across LAN and WAN networks; a client may contact servers to request some service. A user’s desktop, laptop, and handheld device are examples of clients. (Note that a computer can be both a client and a server.) EX1014 (Microsoft Computer Dictionary) at 92 (“**client** *n.* 1. In object-oriented programming, a member of a class (group) that uses the services of another class to which it is not related. *See also* inheritance (definition 1). 2. A process, such as a program or task, that requests a service provided by another program—for example, a word processor that calls on a sort routine built into another program. The client process uses the requested service without having to ‘know’ any working details about the other program or the service itself. *Compare* child (definition 1), descendant (definition 2). 3. On a local area network or the Internet, a computer that accesses shared network resources provided by another computer (called a server). *See also* client/server architecture, server.”).

59. A client/server architecture describes how client computers interact with server computers. EX1014 (Microsoft Computer Dictionary) at 92

(“**client/server architecture** *n.* An arrangement used on local area networks that makes use of distributed intelligence to treat both the server and the individual workstations as intelligent, programmable devices, thus exploiting the full computing power of each. This is done by splitting the processing of an application between two distinct components: a ‘front-end’ client and a ‘back-end’ server. The client component is a complete, stand-alone personal computer (not a ‘dumb’ terminal), and it offers the user its full range of power and features for running applications. The server component can be a personal computer, a minicomputer, or a mainframe that provides the traditional strengths offered by minicomputers and mainframes in a time-sharing environment: data management, information sharing between clients, and sophisticated network administration and security features. The client and server machines work together to accomplish the processing of the application being used. Not only does this increase the processing power available over older architectures but it also uses that power more efficiently. The client portion of the application is typically optimized for user interaction, whereas the server portion provides the centralized, multiuser functionality. *See also* distributed intelligence.”).

60. An “agent” is “1. A program that performs a background task for a user and reports to the user when the task is done or some expected event has taken place.
2. A program that searches through archives or other repositories of information on

a topic specified by the user. Agents of this sort are used most often on the Internet and are generally dedicated to searching a single type of information repository, such as postings on Usenet groups. Spiders are a type of agent used on the Internet. *Also called* intelligent agent. *See also* spider. 3. In client/server applications, a process that mediates between the client and the server. 4. In Simple Network Management Protocol (SNMP), a program that monitors network traffic. *See also* SNMP.” EX1014 (Microsoft Computer Dictionary) at 19.

2. Network Protocols

61. To be useful, computer systems are connected using a worldwide network known as the Internet. Without being able to offer services over the network or access networked services, the Internet would not have become so pervasive in our lives. Indeed, one of the early pioneers in the field of networking, a then fledgling startup named Sun Microsystems (now acquired by Oracle), coined the visionary marketing phrase “the network *is* the computer.”

62. To ensure that networked systems could inter-operate, the community banded together starting in the late 1970’s and came up with several networking standards. One of the most common and heavily used is the Open Systems Interconnection (OSI) model, administered by the International Organization for Standardization (ISO). *See generally* EX1017 (Tanenbaum) at pp. 6-13, Figure 1-7 (reproduced below):

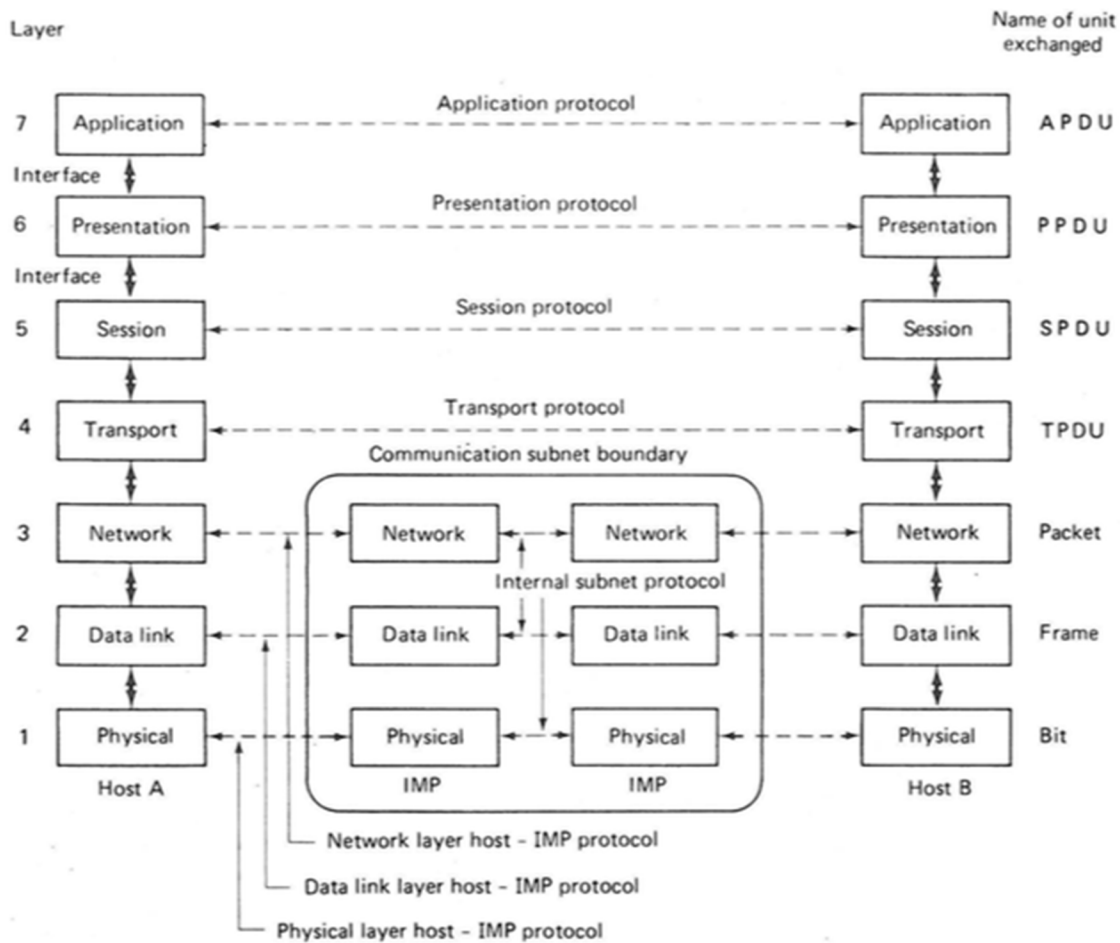


Fig. 1-7. The network architecture used in this book. It is based on the OSI model.

63. The OSI model defines the following seven logical abstraction layers:

64. *Layer 1: Physical Layer*, is the lowest abstraction layer, and defines how bits are communicated over physical media (e.g., wires). Examples here include twisted-pair cables with RJ45 connectors, used by Ethernet.

65. *Layer 2: Data Link Layer*, defines how data frames are transmitted reliably between two nodes over the physical layer. A popular example here is Ethernet.

66. *Layer 3: Network Layer*, defines how to communicate across multiple nodes, including how to route and address packets comprising multiple data frames. The most common example of this layer is the Internet Protocol (IP). This layer is where most traditional firewalls operate.

67. *Layer 4: Transport Layer*, defines how to transmit segments of data comprising multiple packets, and control the traffic (*e.g.*, ordering, acknowledgment, handling missing or duplicate packets, bandwidth control, timeouts and retransmissions, and more). The most popular examples of this layer are the Transmission Control Protocol (TCP), User Datagram Protocol (UDP), and Internet Control Message Protocol (ICMP).

68. *Layer 5: Session Layer*, defines continuous exchanges or communications among parties (*e.g.*, Remote Procedure Calls or RPCs).

69. *Layer 6: Presentation Layer*, defines data translations between a networked service and applications (*e.g.*, encryption, compression, encoding). See VII.b.4 below.

70. *Layer 7: Application Layer*, is the highest abstraction layer and directly corresponds to an application's use of the network. Examples include Web services (HTTP) and URL filtering, email (eSMTP) and anti-virus filtering, file transfer (FTP), and many more. This layer is where most application-level firewalls and filtering devices operate.

71. In a typical communication network, electrical signals arrive on the wire and are captured by a Network Interface Card (NIC). The NIC translates these signals into bits (1's and 0's) and passes these bits as short frames up to the Operating System (OS) for further processing. The frames are often checked for validity and passed on to a software module at the network layer. Again, the data is validated and processed further and passed up to the next layer up the chain.

72. Sometimes, data is deemed to be invalid (*e.g.*, doesn't pass a checksum check), or is a duplicate of data already seen, or perhaps was not intended for the receiving host. In such cases, the data packets are often "dropped" or discarded, and no acknowledgments are sent to the sender. If all goes well, however, the data eventually reaches an application that can interpret it (*e.g.*, a Web browser displaying a JPEG image whose bits arrived from a remote Web server).

73. UDP (User Datagram Protocol) is a connectionless transport layer protocol providing no reliability. *See* EX1018 (Stevens) at p. 12 ("UDP provides no reliability: it sends the datagrams that the application writes to the IP layer, but there is no guarantee that they ever reach their destination."). This means that the packets could be lost, reordered, or duplicated on their way to a destination host, and the sender would have no way of knowing this. Senders using UDP typically have to arrange other means to detect and retransmit lost packets.

74. TCP is a connection-oriented transport protocol, layered on top of the connectionless IP protocol. *See* EX1018 (Stevens), p. 14 (“Even though TCP and UDP use the same network layer (IP), TCP provides a totally different service to the application layer than UDP does. TCP provides a connection-oriented, reliable, byte stream service. The term connection-oriented means the two applications using TCP (normally considered a client and a server) must establish a TCP connection with each other before they can exchange data.”). For example, TCP tracks all packets to ensure that lost packets are retransmitted and acknowledged, to ensure reliable connections between the two TCP endpoints.

75. TCP requires a 3-way handshake sequence before the connection can be established and data can flow in both directions. First, a sending host initiates communications by sending a SYN packet. Second, the recipient prepares for the communications by reserving a network slot and responds with a SYN-ACK message. Finally, the initiating host responds back with an ACK message. At this stage both sides know for certain that bi-directional communication is possible. *See* EX1017 (Tanenbaum) at p. 19 (“**The Transport Layer in the ARPANET (TCP)** ... ARPANET grew into the ARPA Internet ... led to ... introduction of ... **TCP (Transmission Control Protocol)**”) (emphasis in original¹), 20 (“The *SYN*

¹ Hereinafter emphasis original unless otherwise noted.

bit is used to establish connections. The connection request has *SYN* = 1 and *ACK* = 0 to indicate that the piggyback acknowledgement field is not in use. The connection reply does bear an acknowledgement, so it has *SYN* = 1 and *ACK* = 1. In essence the *SYN* bit is used to denote *CONNECTION REQUEST* and *CONNECTION CONFIRM*, with the *ACK* bit used to distinguish between those two possibilities.”) (emphasis in original); *see also* EX1018 (Stevens) at p. 19 (“**Connection Establishment Protocol** ... To establish a TCP connection: 1. The requesting end (normally called the *client*) sends a SYN segment ... 2. The server responds with its own SYN segment ... The server also acknowledges the client’s SYN by ACKing the client’s ISN plus one ... 3. The client must acknowledge this SYN from the server by ACKing the server’s ISN plus one (segment 3). These three segments complete the connection establishment. This is often called the *three-way handshake*.”) (emphasis in original), 20 (“The side that sends the first SYN is said to perform an *active open*. The other side, which receives this SYN and sends the next SYN, performs a *passive open*.”) (emphasis in original), Fig. 18.3 at p. 20 (“Time line of connection establishment and connection termination.” below).

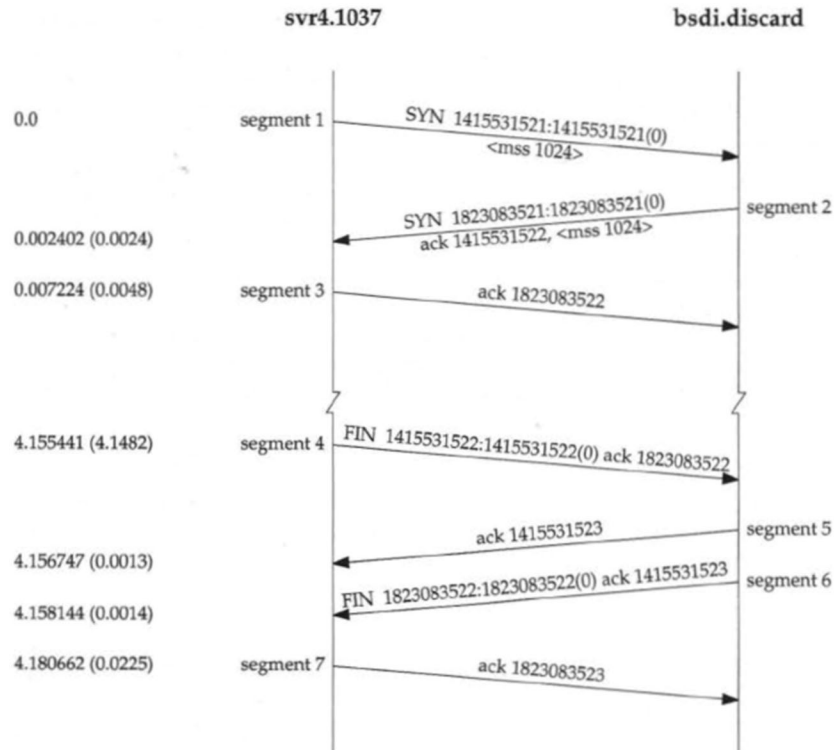


Figure 18.3 Time line of connection establishment and connection termination.

76. IP headers include the source and destination addresses. *See* EX1018 (Stevens) at p. 7 (Figure 3.1, reproduced below):

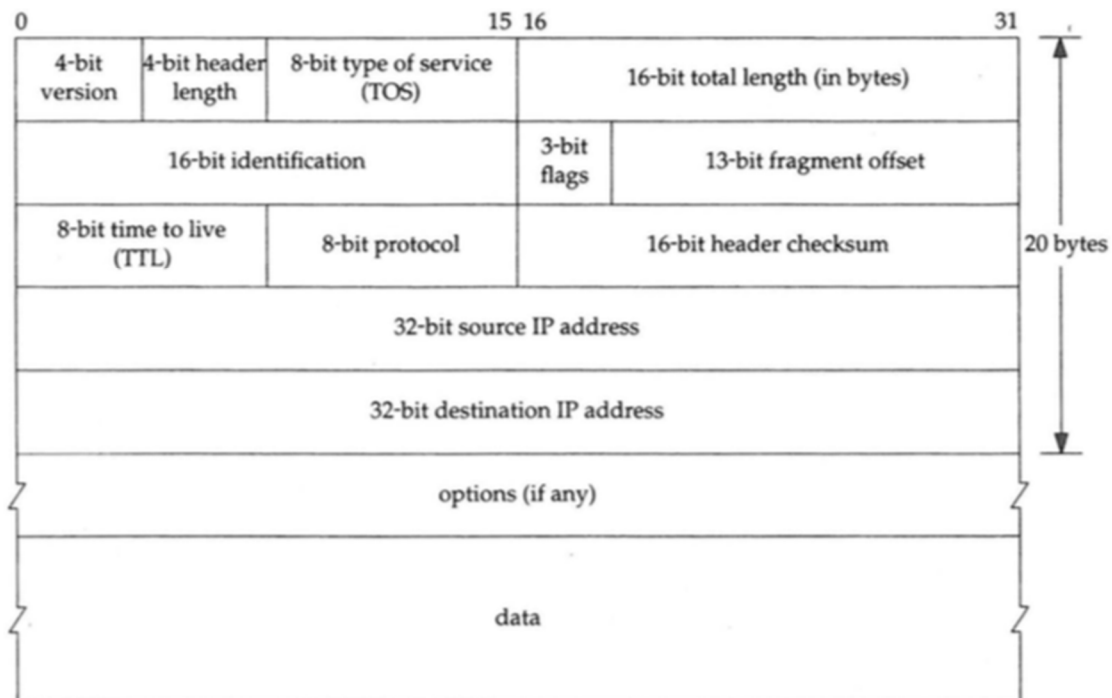


Figure 3.1 IP datagram, showing the fields in the IP header.

77. At the transport layer, UDP packets include the source and destination ports. See EX1018 (Stevens) at p. 13 (Figure 11.2, reproduced below):

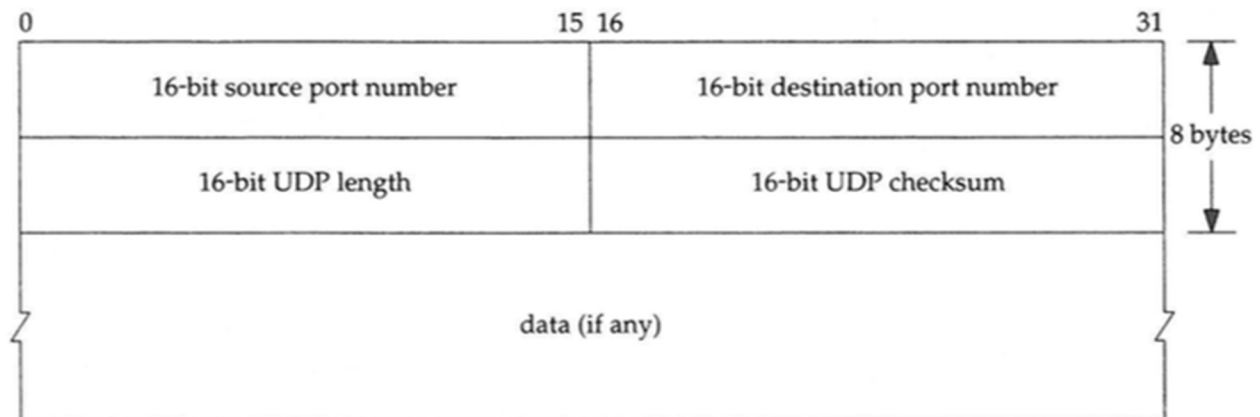


Figure 11.2 UDP header.

78. At the transport layer, TCP packets also include the source and destination ports. See EX1018 (Stevens) at p. 16 (Figure 17.2, reproduced below):

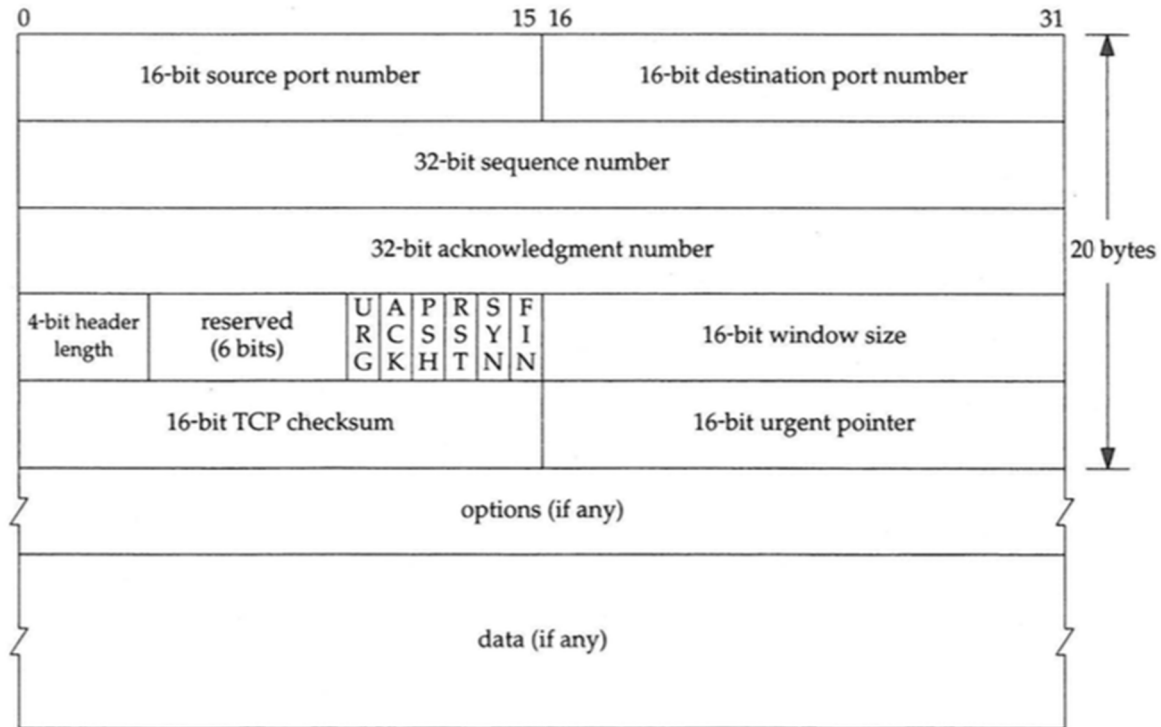


Figure 17.2 TCP header.

79. Therefore, TCP and UDP packets using IP include two IP addresses and two ports—for the source and destination ones.

80. ATM is a networking technology that establishes a full circuit between two endpoints, akin to the telephone system. ATM is thus said to be a circuit switched network. *See* EX1014 (Microsoft Computer Dictionary) at pp. 33-34 (“**ATM** *n.* 1. Acronym for **A**synchronous **T**ransfer **M**ode. A network technology capable of transmitting data, voice, video, and frame relay traffic in real time. Data, including frame relay data, is broken into packets containing 53 bytes each, which are switched between any two nodes in the system at rates ranging from 1.5 Mbps

to 622 Mbps. ATM is defined in the broadband ISDN protocol at the levels corresponding to levels 1 and 2 of the ISO/OSI model. ...”).

81. A network flow can be thought of as a logical communication or a stream of bytes in a session between at least two entities. *See* EX1018 (Stevens) at p. 17 (“If we consider the stream of bytes flowing in one direction between two applications, TCP numbers each byte with a sequence number. ... This means that data can be flowing in each direction, independent of the other direction. Therefore, each end of a connection must maintain a sequence number of the data flowing in each direction.”). TCP provides implicit flow control, where both ends of a TCP connection can control their transmission rates. *See* EX1018 (Stevens) at p. 15 (“TCP also provides flow control. Each end of a TCP connection has a finite amount of buffer space. A receiving TCP only allows the other end to send as much data as the receiver has buffers for. This prevents a fast host from taking all the buffers on a slower host.”), 18 (“TCP’s flow control is provided by each end advertising a *window size*.”). In the early 1990’s, TCP implementations advanced (in no small part thanks to Van Jacobson) to improve TCP’s reliability through flow control (*e.g.*, using “exponential back-off” during packet transmission timeouts, and “slow start” to gradually increase transmission rates).

82. Proxies, routers, bridges, and routers share much in common: they receive network packets on one end, process them, then transmit them on another end: their difference often boils down to which OSI network layer they operate at.

83. A “router” is “[a]n intermediary device on a communications network that expedites message delivery. On a single network linking many computers through a mesh of possible connections, a router receives transmitted messages and forwards them to their correct destinations over the most efficient available route. On an interconnected set of local area networks (LANs) using the same communications protocols, a router serves the somewhat different function of acting as a link between LANs, enabling messages to be sent from one to another. *See also* bridge, gateway.” EX1014 (Microsoft Computer Dictionary) at 415.

84. A “bridge” is “[a] device that connects networks using the same communications protocols so that information can be from one to the other. Compare gateway. 2. A device that connects two local area networks, whether or not they use the same protocols. A bridge operates at the ISO/OSI data-link layer. *See also* data-link layer. *Compare* router.” EX1014 (Microsoft Computer Dictionary) at 63.

85. A “bridge router” is “[a] device that supports the functions of both a bridge and router. A bridge router links two segments of a local or wide area network, passing packets of data between the segments as necessary, and uses Level

2 addresses for routing. *Also called* Brouter. *See also* bridge (definition 2), router.” EX1014 (Microsoft Computer Dictionary) at 63.

86. A “gateway” is “[a] device that connects networks using different communications protocols so that information can be passed from one to the other. A gateway both transfers information and converts it to a form compatible with the protocols used by the receiving network. *Compare* bridge.” EX1014 (Microsoft Computer Dictionary) at 215.

87. A “proxy server” is “[a] firewall component that manages Internet traffic to and from a local area network (LAN) and can provide other features, such as document caching and access control. A proxy server can improve performance by supplying frequently requested data, such as a popular Web page, and can filter and discard requests that the owner does not consider appropriate, such as requests for unauthorized access to proprietary files. *See also* firewall.” EX1014 (Microsoft Computer Dictionary) at 387.

88. Over the years, many firewalls, proxies, routers, network devices, and security gateways were developed. Earlier ones tended to operate at one or a few of the lower OSI network layers and applied simpler stateless rules; subsequent firewalls were stateful and understood the legitimate life cycle of a protocol (*e.g.*, TCP’s 3-way handshaking). *See* EX1019 (Cheswick) at pp. 6-7 (“**Packet-Filtering Gateways...** Packet filters can provide a cheap and useful level of gateway security.

Used by themselves, they are cheap: the filtering abilities come with the router software. Since you probably need a router to connect to the Internet in the first place, there is no extra charge.”), 7 (“Packet filters work by dropping packets based on their source or destination addresses or ports... Most security policies require finer control than this: they need to define access to specific services for hosts that are otherwise untrusted.”), 8 (“**Filtering Other Protocols...** Other protocols are layered on top of IP as well; depending on your environment, these may need to be filtered also. Of particular import is ICMP, the Internet Control Message Protocol.”), 9 (“Many advanced gateway designs rely in part on packet filtering. They are likely to work well, but pure packet filters leave us feeling uncomfortable.”).

89. As the field quickly matured, more advanced gateways began to operate at the application layer (*e.g.*, anti-virus, URL filtering). *See* EX1019 (Cheswick) at p. 9 (“**Application-Level Gateways...** An application-level gateway represents the opposite extreme in firewall design. Rather than using a general-purpose mechanism to allow many different kinds of traffic to flow, special-purpose code can be used for each desired application. Although this seems wasteful, it is likely to be far more secure than any of the alternatives... Application gateways have another advantage that in some environments is quite critical: it is easy to log and control *all* incoming and outgoing traffic.”).

90. Principles of network quality of service (QoS, sometimes called traffic shaping) were well known in the 1980's, including explicit bandwidth control and reservations. *See* EX1017 (Tanenbaum) at p. 18 (“The *qos* parameter is actually two lists of values that determine the quality of service provided by the connection. The first list gives the goal—what the caller really wants. The second list gives the minimum values considered acceptable. If the network service is unable to provide at least the minimum value for any parameter specified by either the caller (calling party) or the callee (called party), the connection establishment fails. The values that may be specified include throughput, delay, error rate, secrecy, and cost, among others.”).

3. The Domain Name System, Hostnames, and IP Addresses

91. As discussed in §VII.A above, every computer has an internal local hostname called “localhost” with an IP address 127.0.0.1. This IP address is accessible only internally on the same host. Computers, however, can have one or more external host names and IP addresses, used to reach (and be reached by) other hosts on the Internet.

92. A “host name” is “[t]he name of a specific server on a specific network within the Internet, leftmost in the complete host specification. For example, *www.microsoft.com* indicates the server called ‘www’ within the network at Microsoft Corporation.” EX1014 (Microsoft Computer Dictionary) at 236.

93. A “domain name” is “[a]n address of a network connection that identifies the owner of that address in a hierarchical format: server.organization.type. For example, www.whitehouse.gov identifies the Web server at the White House, which is part of the U.S. government.” *Id.* at 158.

94. A “top-level domain” is, “[i]n the domain-name system of Internet addresses, any of the broadest category of names, under which all domain names fit. Top-level domains for sites in the United States include .com, .edu, .gov, .net, and .org. *See also* DNS (definition 1), major geographic domain.” *Id.* at 472.

95. For example, “www.microsoft.com” is often called the “fully qualified host name” (FQHN) as it comprises several parts: the short host name “www” and the fully qualified domain name (FQDN) “microsoft.com,” which itself comprises a top-level domain (TLD) of “.com” and a sub-domain “microsoft.com”. The entity Microsoft Corporation is delegated (by Internet-wide authorities) the right to define any sub-domain and hostname under “microsoft.com.”

96. Host names are intended for people to use because they are easier to remember. But internally, for computers to be able to communicate with each other, they must do so with a unique, publicly accessible 32-bit number (called an “IP address”) assigned to each host across the entire Internet. Host names can change and may not even be unique. This is akin to how every U.S. person has a name that

is not unique and can change, but each is assigned a unique social security number at birth.

97. An “IP address” is “Short for Internet Protocol address. A 32-bit (4-byte) binary number that uniquely identifies a host (computer) connected to the Internet to other Internet hosts, for the purposes of communication through the transfer of packets. An IP address is expressed in ‘dotted quad’ format, consisting of the decimal values of its 4 bytes, separated with periods; for example, 127.0.0.1. The first 1, 2, or 3 bytes of the IP address, assigned by InterNIC Registration Services, identify the network the host is connected to; the remaining bits identify the host itself. The 32 bits of all 4 bytes together can signify almost 2^{32} , or roughly 4 billion, hosts. (A few small ranges within that set of numbers are not used.) *See also* host, InterNIC, IP, packet (definition 2). *Compare* domain name.” *Id.* at 264.

98. Managing the many domains, host names, and IP addresses on the Internet is accomplished with a massive, distributed network of “Domain Name System” servers that each maintain their own small view of the Internet’s naming.

99. “DNS” is an “1. Acronym for Domain Name System. The system by which hosts on the Internet have both domain name addresses (such as *bluestem.prairienet.org*) and IP addresses (such as 192.17.3.4). The domain name address is used by human users and is automatically translated into the numerical IP address, which is used by the packet-routing software. *See also* domain name

address, IP address. 2. Acronym for Domain Name Service. The Internet utility that implements the Domain Name System (see definition 1). DNS servers, also called *name servers*, maintain databases containing the addresses and are accessed transparently to the user.” *Id.* at 155.

100. A “DNS server” is “[a] computer that can answer Domain Name Service (DNS) queries. The DNS server keeps a database of host computers and their corresponding IP addresses. Presented with the name apex.com, for example, the DNS server would return the IP address of the hypothetical company Apex. *See also* DNS (definition 2), IP address.” *Id.* at 155.

101. The Domain Name System was so important to the very foundations of the Internet, that it was one of the earliest standards to be defined in the 1980s. Well before the WWW even existed, computers needed a way to identify each other and access other computers consistently.

102. “The DNS has three major components:” (1) “The DOMAIN NAME SPACE and RESOURCE RECORDS” which is a database that maps host names and IP addresses to each other, among other pieces of information; (2) “NAME SERVERS” that respond to queries from anywhere in the world to translate hostnames to IP addresses and vice versa, where “[a] name server may cache structure or set information about any part of the domain tree...” EX1020 (Domain Names Concepts, RFC-1034) at 6.

103. The third part are software components used to perform these name/IP mapping, called “resolvers.” (3) “RESOLVERS are programs that extract information from name servers in response to client requests. Resolvers must be able to access at least one name server and use that name server’s information to answer a query directly, or pursue the query using referrals to other name servers. A resolver will typically be a system routine that is directly accessible to user programs; hence no protocol is necessary between the resolver and the user program.” *Id.* at 6.

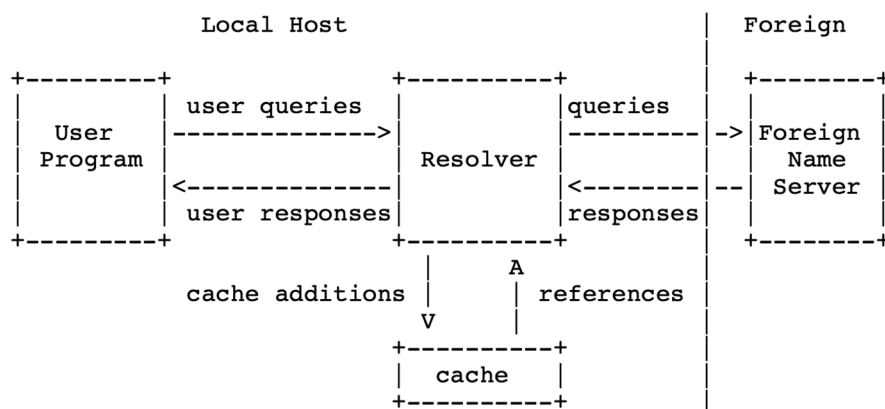
104. “Resolvers are programs that interface user programs to domain name servers. In the simplest case, a resolver receives a request from a user program (*e.g.*, mail programs, TELNET, FTP) in the form of a subroutine call, system call etc., and returns the desired information in a form compatible with the local host’s data formats. The resolver is located on the same machine as the program that requests the resolver’s services, but it may need to consult name servers on other hosts. Because a resolver may need to consult several name servers, or may have the requested information in a local cache, the amount of time that a resolver will take to complete can vary quite a bit, from milliseconds to several seconds.” *Id.* at 29.

105. Indeed, name resolution in the world-wide DNS system can be slow, as it may require querying multiple DNS servers all around the globe. Many such

queries are performed every second on the Internet. Thus, DNS resolutions can be slow, and why caching is employed extensively.

106. “A very important goal of the resolver is to eliminate network delay and name server load from most requests by answering them from its cache of prior results. It follows that caches which are shared by multiple processes, users, machines, etc., are more efficient than non-shared caches.” EX1020 (Domain Names Concepts, RFC-1034) at 29. “7.4. Using the cache [] In general, we expect a resolver to cache all data which it receives in responses since it may be useful in answering future client requests.” EX1021 (Domain Names - Implementation and Specification, RFC-1035) at 47.

107. The overall architecture of a DNS system, including clients, resolvers, caches, and DNS servers, can be seen in the following figure:



Id. at 4.

108. DNS describes the address (‘A’) records to define the IP address(es) of a named host. “A RDATA format ... Hosts that have multiple Internet addresses

will have multiple A records. ... The RDATA section of an A line in a master file is an Internet address expressed as four decimal numbers separated by dots without any imbedded spaces (e.g., '10.2.0.52' or '192.0.5.6')." EX1021 (Domain Names - Implementation and Specification, RFC-1035) at pp. 20-21.

109. A host can have multiple records such as 'A' address records, and the DNS server will return all of them to a query. "In Listing 6.4, every host name has a unique address, and every address has a unique host name. This does not have to be the case, as illustrated in Listing 6.5. ... In Listing 6.5, owl is a multi-homed host. The name owl is assigned both of the host's addresses, so a query for owl returns both 172.16.5.15 and 172.16.7.32. The remote system's resolver can sort out which address it wants to use. (As you learned in Chapter 4, 'Configuring the Resolver,' the resolver sortlist option can be used to control which address is used.)" EX1024 (Hunt) at p. 144, Listing 6.5 (reproduced below):

Listing 6.5 Address Records for Multi-Homed Hosts

owl	A	172.16.5.15
owl	A	172.16.7.32
owl5	A	172.16.5.15
owl7	A	172.16.7.32

110. DNS supports multiple records. *See generally*, EX1021 (Domain Names - Implementation and Specification, RFC-1035). One extensible record is the "text" (TXT) record, a flexible record into which any information can be placed

and subsequently processed by other programs and scripts. “TXT RDATA format ... where: TXT-DATA [is] One or more <character-string>s. TXT RRs are used to hold descriptive text. The semantics of the text depends on the domain where it is found. ... **Text Record (TXT)** The Text record is used to define free-form information about the named object. Its format is simple ... Because of its free-form nature, the TXT record has been used over time for special purposes, such as providing input to locally developed scripts that collect domain information. An earlier version of BIND that runs on some Unix systems even uses these records to define security information.” EX1024 (Hunt), p. 366.

111. Putting it all together, consider a user using a Web browser to browse <http://www.google.com/some/index.html>. The user first types the URL into the browser. The browser then attempts to resolve the hostname www.google.com using the client’s DNS resolver libraries. These libraries parse the hostname into three parts, delimited by the ‘.’, and issue DNS queries to the right servers across the Internet. The resolver library consults local host and site DNS caches for speedier resolution. Once the IP address is found, say 142.251.35.164, the Web browser attempts to open a TCP/IP connection to the Web server at that IP address, using default port 80, using the HTTP protocol. If successful, the browser will issue an HTTP command such as “GET /some/index.html” to request the retrieval of the server’s content for the file “/some/index.html”. Once that content arrives back at

the browser, it translates the content as HTML and displays it on the browser (*e.g.*, text, graphics, hyper-links, and more).

112. It should be noted that translating hostnames to IP addresses does not have to be accomplished using DNS, but can also be done for individual hosts using a simple configuration file called `/etc/hosts` (*e.g.*, popular on Unix/Linux hosts). The local `/etc/hosts` file is a simple text file that can map one or more hostnames to an IP address. Often, the same hostname resolution libraries used for DNS can also consult `/etc/hosts`. In fact, computer systems can resolve addresses using DNS, `/etc/hosts`, other types of network databases, or a combination of any. (I personally have used `/etc/hosts` files for many years; *see* §II above.)

113. “In fact, all IP network servers have a host table. If you have experience with Unix, you recognize the name `/etc/hosts` because that is where Unix systems store the host table. If your background is Windows NT, you know the host table as the `%SystemRoot%\System32\Drivers\etc\hosts` file. The structure and content of the hosts file is the same on all of these systems.” EX1024 (Hunt) at p. 4.

114. Hunt’s “Listing 1.1 contains an example from a Linux system that illustrates the structure of the hosts file:”

Listing 1.1 Structure of the /etc/hosts File

```
$ cat /etc/hosts
#
# Table of IP addresses and host names
#
127.0.0.1          localhost
172.16.5.5         crow
172.16.5.1         wren ns1
172.16.5.4         hawk gw5
172.16.5.20        kestrel kestral
172.16.5.2         robin redbreast bob
172.16.5.6         eagle www
172.16.5.7         bluebird blue news
```

Id., p. 4; *see also id.*, pp. 3-7.

115. Next, Web browsers may be configured to contact a local Web cache or proxy, rather than contacting the actual remote server (*e.g.*, www.google.com) directly. In that case, the Web proxy appears to the Web client as any compliant Web server. In turn, the proxy acts as a Web client and communicates with the actual remote Web server, who may be unaware that it is being contacted by a proxy on behalf of a user's Web browser.

116. Hostnames on the Internet can be assigned a fixed (static) IP address or get one assigned dynamically. "DHCP" is "[a]cronym for **D**ynamic **H**ost **C**onfiguration **P**rotocol. A TCP/IP protocol that enables a network connected to the Internet to assign a temporary IP address to a host automatically when the host

connects to the network. *See also* IP address, TCP/IP. *Compare* dynamic SLIP.” EX1014 (Microsoft Computer Dictionary) at pp. 142, 166.

117. The DHCP protocol supports multiple messages to allow a client to join and leave an IP network: DHCPDISCOVER lets a “Client broadcast to locate available servers”; DHCPOFFER is a message from “[s]erver to client in response to DHCPDISCOVER with offer of configuration parameters”; DHCPACK is an acknowledgement message from the “[s]erver to client with configuration parameters, including committed network address”; and DHCPRELEASE is a message from “[c]lient to server relinquishing network address and cancelling remaining lease.” EX1023 (Dynamic Host Configuration Protocol, RFC-2131) at p. 14.

118. “Network Address Translation” or “NAT” is an “[a]cronym for **n**etwork **a**ddress **t**ranslation. The process of converting between IP addresses used within an intranet or other private network (called a stub domain) and Internet IP addresses. This approach makes it possible to use a large number of addresses within the stub domain without depleting the limited number of available numeric Internet IP addresses. *See also* intranet, IP address.” EX1014 (Microsoft Computer Dictionary) at pp. 323, 327.

119. On the Internet, IP and other packets can be wrapped inside other packets, enabling one protocol to be overlayed on top of other protocol. To “tunnel”

is “[t]o encapsulate or wrap a packet or a message from one protocol in the packet for another. The wrapped packet is then transmitted over a network via the protocol of the wrapper. This method of packet transmission is used to avoid protocol restrictions. *See also* communications protocol, packet (definition 2).” EX1014 (Microsoft Computer Dictionary) at p. 479; *see* §VII.B.4 below.

120. The IETF standardized various forms of IP packet encapsulation, as they are often useful for security purposes (*e.g.*, “IP Security”). “The Encapsulating Security Payload (ESP) header is designed to provide a mix of security services in IPv4 and IPv6. ESP may be applied alone, in combination with the IP Authentication Header (AH) [KA97b], or in a nested fashion, *e.g.*, through the use of tunnel mode (see ‘Security Architecture for the Internet Protocol’ [KA97a], hereafter referred to as the Security Architecture document). Security services can be provided between a pair of communicating hosts, between a pair of communicating security gateways, or between a security gateway and a host. ... The ESP header is inserted after the IP header and before the upper layer protocol header (transport mode) or before an encapsulated IP header (tunnel mode).” EX1022 (IP Encapsulating Security Payload (ESP), RFC-2406) at p. 2; *see* §VII.B.4 below.

4. IP Security

121. RFC-2401, titled “Security Architecture for IP” “specifies the base architecture for IPsec compliant systems. The goal of the architecture is to provide various security services for traffic at the IP layer, in both the IPv4 and IPv6 environments. This [RFC-2401] document describes the goals of such systems ... Subsequent documents will address additional architectural details of a more advanced nature, e.g., use of IPsec in NAT environments and more complete support for IP multicast.” EX1027 (Security Architecture for IP, RFC-2401) at p. 3.

122. IPsec secures data by encrypting packets. “3.1 What IPsec Does [] IPsec provides security services at the IP layer by enabling a system to select required security protocols, determine the algorithm(s) to use for the service(s), and put in place any cryptographic keys required to provide the requested services. ... The set of security services that IPsec can provide includes ... confidentiality (encryption), and limited traffic flow confidentiality.” EX1027 (Security Architecture for IP, RFC-2401) at p. 6.

123. IPsec can also compress data packets to save space. “The IPsec DOI also supports negotiation of IP compression [SMPT98], motivated in part by the observation that when encryption is employed within IPsec, it prevents effective compression by lower protocol layers.” EX1027 (Security Architecture for IP, RFC-2401) at p. 6.

124. IPsec encapsulates packets (placing one as the payload of another). “3.2 How IPsec Works [] IPsec uses two protocols to provide traffic security -- Authentication Header (AH) and Encapsulating Security Payload (ESP). Both protocols are described in more detail in their respective RFCs [KA98a, KA98b]. • The IP Authentication Header (AH) [KA98a] provides connectionless integrity, data origin authentication, and an optional anti-replay service. • The Encapsulating Security Payload (ESP) protocol [KA98b] may provide confidentiality (encryption), and limited traffic flow confidentiality.” EX1027 (Security Architecture for IP, RFC-2401) at p. 6.

C. Virtualization

125. Among computer scientists, there is an axiom stating that “you can solve almost any problem by adding another level of indirection.” Indeed, virtualization is an indirection technique that makes one thing seem like another; virtualization often presents a physical-world artifact as one that appears as an illusion or “fake.” For something to be “virtual”, it is “[o]f or pertaining to a device, service, or sensory input that is perceived to be what it is not in actuality, usually as more ‘real’ or concrete than it actually is.” EX1014 (Microsoft Computer Dictionary) at p. 497.

126. Indeed, the Microsoft Computer Dictionary includes 35 examples and definitions of virtualization. EX1014 (Microsoft Computer Dictionary) at pp. 497-500.

127. In one example, a “virtual machine” is “[s]oftware that mimics the performance of a hardware device, such as a program that allows applications written for an Intel processor to be run on a Motorola chip. *Acronym: VM.*” EX1014 (Microsoft Computer Dictionary) at p. 498.

128. In another example, “virtual memory” is “[m]emory that appears to an application to be larger and more uniform than it is. Virtual memory may be partially simulated by secondary storage such as a hard disk. Applications access memory through virtual addresses, are translated (mapped) by special hardware and software onto physical addresses. *Acronym: VM. Also called disk memory. See also paging, segmentation.*” EX1014 (Microsoft Computer Dictionary) at p. 498.

129. In yet another example, “virtual reality” is “[a] simulated 3-D environment that a user can experience and manipulate as if it were physical. The user sees the environment on display screens, possibly mounted in a special pair of goggles. Special input devices, such as gloves or suits fitted with motion sensors, detect the user’s actions. *Acronym: VR.*”. EX1014 (Microsoft Computer Dictionary) at p. 499.

130. Virtualization had been applied to storage systems, disk devices, and drivers. A “virtual device driver” is “[s]oftware in Windows 95 that manages a hardware or software system resource. If a resource retains information from one access to the next that affects the way it behaves when accessed (for example, a disk controller with its status information and buffers), a virtual device driver must exist for it.” EX1014 (Microsoft Computer Dictionary) at p. 498.

131. A “virtual disk” or “RAM disk” is “[s]hort for random access memory disk. A simulated disk drive whose data is actually stored in RAM memory. A special program allows the operating system to read from and write to the simulated device as if it were a disk drive. RAM disks are extremely fast, but they require that system memory be given up for their use. Also, RAM disks usually use volatile memory, so the data stored on them disappears when power is turned off. Many portables offer RAM disks that use battery-backed CMOS RAM to avoid this problem. *See also* CMOS RAM. *Compare* disk cache.” EX1014 (Microsoft Computer Dictionary) at pp. 395-396, 498.

132. Virtualization is sometimes said to be “logical” representation of physical reality. “Logical” is “[o]f or pertaining to a conceptual piece of equipment or frame of reference, regardless of how it may be realized physically. *Compare* physical.” EX1014 (Microsoft Computer Dictionary), p. 288 (“logical” definition 2).

133. A “logical drive” or “logical device” is “[a] device named by the logic of a software system, regardless of its physical relationship to the system. For example, a single floppy disk drive can simultaneously be, to the MS-DOS operating system, both logical drive A and drive B.” EX1014 (Microsoft Computer Dictionary) at p. 288.

134. Moreover, one can virtualize files and file systems. This was the subject of much of my graduate-level research and became part of my Masters and PhD Dissertations. *See* §II above and EX1012 (Zadok Curriculum Vitae).

135. Virtualization, of course, has been applied to networking as well. A “virtual network” is “[a] part of a network that appears to a user to be a network of its own. For example, an Internet service provider can set up multiple domains on a single HTTP server so that each one can be addressed with its company’s registered domain name. *See also* domain name, HTTP server (definition 1), ISP.” EX1014 (Microsoft Computed Dictionary) at p. 499.

136. A “virtual LAN” is “[s]hort for **virtual local area network**. A local area network consisting of groups of hosts that are on physically different segments but that communicate as though they were on the same wire. *See also* LAN.” EX1014 (Microsoft Computed Dictionary) at p. 498.

137. One of the most famous uses of a (secure) virtual network is a private one. A “virtual private network” (VPN) is “[a] set of nodes on a public network

such as the Internet that communicate among themselves using encryption technology so that their messages are as safe from being intercepted and understood by unauthorized users as if the nodes were connected by private lines. 2. A wide area network formed of permanent virtual circuits (PVCs) on another network, especially a network using technologies such as ATM or frame relay. *Acronym:* VPN. *See also* ATM (definition 1), frame relay, PVC.” EX1014 (Microsoft Computed Dictionary), p. 499. A POSITA would have understood definition 2 of VPN to be a form of an overlay network because it is a collection of individual virtual network segments. *See* §VII.B.4 above.

VIII. THE '785 PATENT

A. Overview

138. The claims of the '785 patent relate to a “virtual network manager” or “virtual network system” and include several key components: (1) a computing device or “network interface configured for data communication,” (2) a “register module configured to register devices in a virtual network,” (3) a router or route director, and (4) a “DNS server configured to receive a DNS request” from a first [source] device and return three addresses to the source, including “a network address associated with a network route director, a private network address associated with a second [destination] device in the virtual network, and a virtual

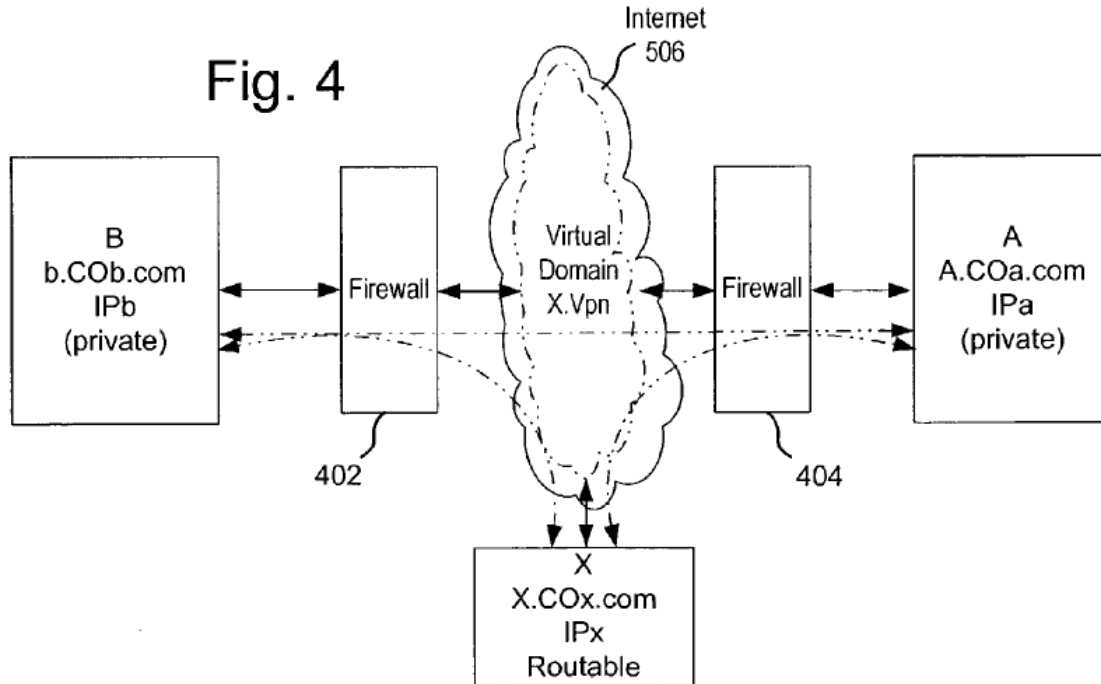
network address associated with the second [destination] device.” EX1001 (’785 patent) at 36:37-56 (claim 30).

139. The ’785 patent identifies the so-called IP address depletion problem as one of the challenges it was trying to solve in order to “enable the rapid creation of secure means that allows local and remote specified entities [computer devices] to communicate and collaborate from any location via a standard Internet connection.” *Id.* at 1:32-37. The ’785 patent indicates that, due to its own popularity, the Internet is “running out of [IP] addresses” that can be assigned to each device, thereby limiting the ability for devices in private networks, for example, to communicate or be accessible via the Internet. *Id.* at 1:50-55. The ’785 patent also purports to address the “mobility problem,” involving devices being assigned constantly changing or dynamic IP addresses, which other devices on the Internet wishing to communicate with them might find difficult to track. *Id.* The ’785 patent thus describes “a need for a system that provides for local and remote entities to communicate and collaborate using the Internet, [a system that] can work with existing NAT devices and firewalls, and allows for devices to move to different physical networks.” *Id.* at 2:45-48.

140. In the “Description of the Related Art” section, the ’785 admits that NAT solutions were well known in the art, and used to transparently translate between private and public IP addresses.

141. “One popular solution to the depleting address problem is Network Address Translation (NAT). This concept includes predefining a number of network addresses to be private addresses and public addresses. Public addresses are unique addresses that should only be used by one entity having access to the Internet. That is, no two entities on the Internet should have the same public address. Private addresses are not unique and are typically used for entities not having direct access to the Internet. Private addresses are used in private networks. Private addresses can be used by more than one organization or network. NAT assumes that all of the machines on a network will not need to access the Internet at all times. Therefore, there is no need for each machine to have a public address. A local network can function with a small number of one or more public addresses assigned to a NAT device. The remainder of the machines on the network will be assigned private addresses. Since entities on the network have private addresses, the network is considered to be a private network.” EX1001 (’785 patent) at 1:36-2:7.

142. One embodiment of the ’785 patent discloses a “secure Virtual Community Network or (‘VCN’),” described as “a private dynamic network which acts a private LAN for computing devices coupled to public networks or private networks.” *Id.* at 8:66-9:5, Fig. 4.

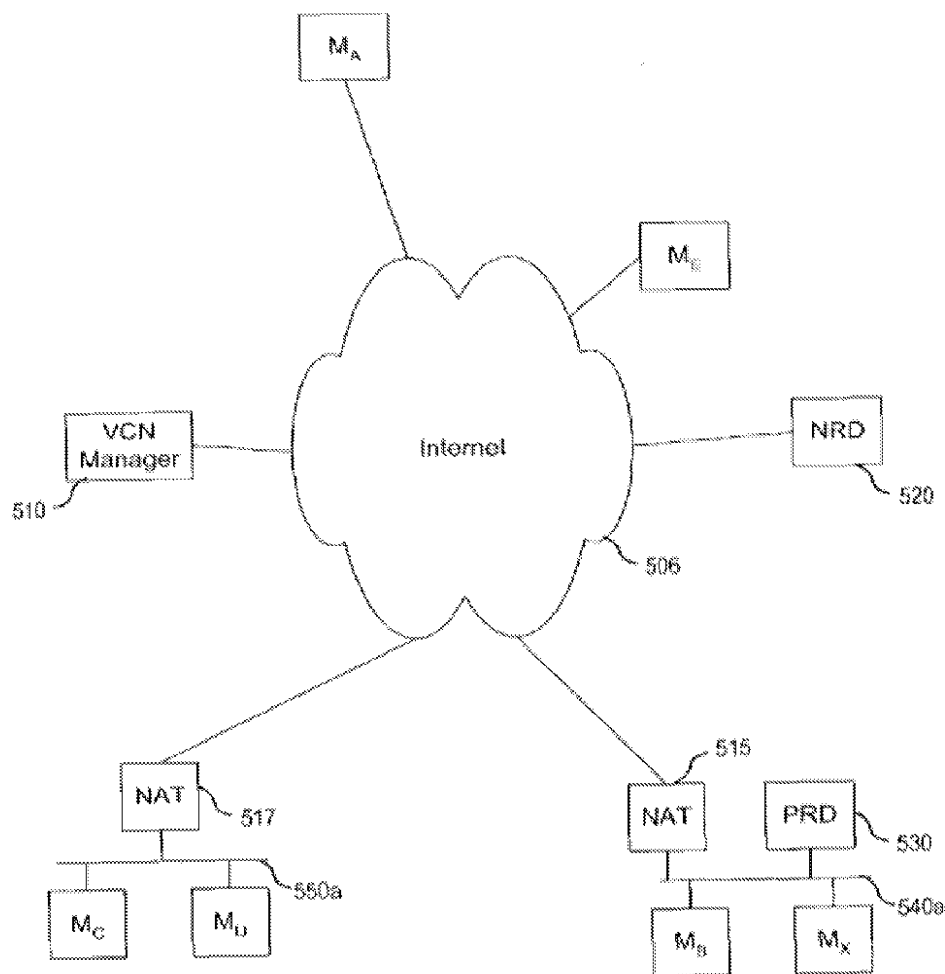


'785 patent, Fig. 4.

143. Figure 4 shows an exemplary system of a VCN, which comprises a “computer or device B (host name—b.Cob.com) in a first private domain and computer or device A (host name—a.Coa.com) in a second private domain, both of which are coupled to the Internet by firewall devices **402, 404.**” *Id.* at 9:9-13. The firewall devices are Network Address Translation (“NAT”) devices. *Id.* at 9:13-15. Moreover, “[c]omputer or device X is coupled directly to the Internet and has a public IP address.” *Id.* at 9:16-17. “Devices A, B, and X can join the VCN, leave the VCN, or allows other devices in the VCN to communicate with them.” *Id.* at 9:18-23. “All members of the VCN are accessible as if they were part of a physical local network” (e.g., all in the same private network)—allowing direct communication between devices within the virtual domain. *Id.* at 9:22-27.

144. The '785 patent specifies that “hardware architecture for the machines, servers, or other devices used to implement the present invention should be well understood to one of ordinary skill in the art,” and outlines suitable hardware or software components, including “one or more processors, a memory, a mass storage device, a portable storage device, a first network interface, a second network interface, and I/O devices, in communication with each other.” *Id.* at 9:42-48. The '785 patent acknowledges those are conventional computer components, noting that “the memory can be any conventional computer memory,” “[t]he choice of processor is not critical,” and “[t]he network interface can include a network card for connecting to an Ethernet or other type of LAN.” *Id.* at 9:48-58. The patent also discloses the network interface can include or be connected to a firewall. *Id.* The network interfaces are inside a computer and when the computer also functions as a “router,” it “includes two or more network interfaces.” *Id.* However, “in other embodiments, the computer may include only one network interface.” *Id.*

145. Another example of the system is illustrated in Figure 5A.

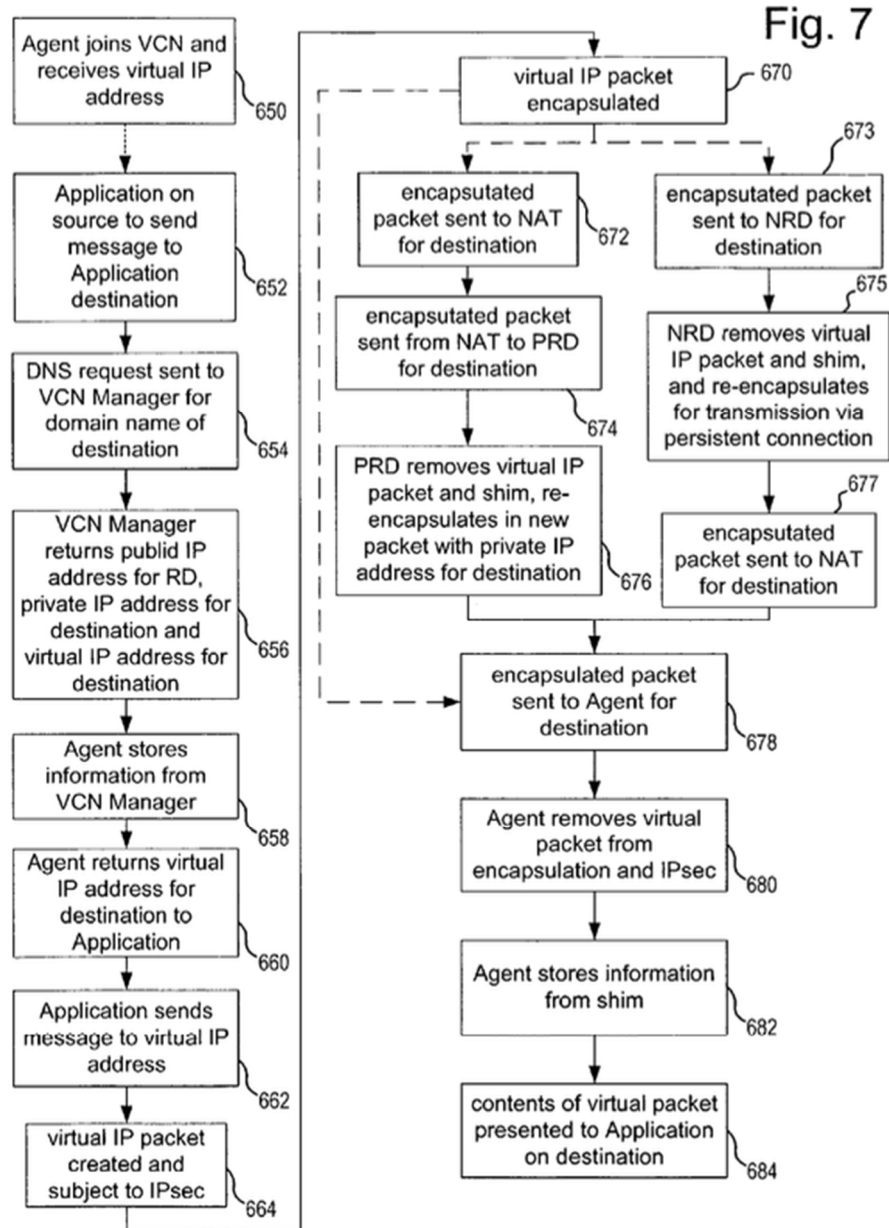


3

Id. at Fig. 5A.

146. In Figure 5A, the VCN includes a VCN Manager **510**, a Network Route Director **520** and/or Private Route Director **530**, and a number of user devices (*e.g.*, M_C , M_X , M_A , etc.). *Id.* at 10:24-27. The '785 patent explains that “[t]he VCN Manager **510** is a central server or server cluster providing management, connection, and security services for the VCN.” *Id.* at 10:66-11:1. The “various user machines, M_n , are coupled to the Internet **506**, as well as other devices.” *Id.* at 10:17-20. “In

order to establish or participate in communication within the VCN, a device must register with and join the VCN, thereby becoming a member.” To join, “[t]he Agent will send a message to the VCN Manager 510 and receive a virtual IP address in return.” *Id.* at 14:10-15. This is shown in step 650 of Figure 7. *Id.*



'785 patent, Fig. 7.

147. The '785 patent describes the virtual IP or network address as preferably another private network address that is chosen in a well-defined set of addresses (virtual address realm) such that each virtual network address occurs only once across the virtual network and thus is a unique identifier of the device within the network address. EX1001 ('785 patent) at 12:50-54. To that end, the '785 patent describes this virtual network address in open terms, adding that it can be a string, an IPV4, or IPV6 compliant address. *Id.* at 12:40-44.

148. A new member device wanting to send a message to another device in the VCN “will initiate a DNS request,” which is then sent to the VCN Manager **510**, as shown in step 654 of Fig. 7 above. *Id.* at 14:32-34. In response, the “VCN Manager 510 returns the public address of the Route Director for the destination, the private address for the destination device, and a virtual IP address for the destination device,” as shown in step 656 of Fig. 7 above. *Id.* at 14:34-38. “This information is then stored in a table, or any other data structure” for creating a virtual IP packet to be forwarded to the destination address. *Id.* at 14:48-15:15. When the destination wishes to respond to the source or the source wishes to send additional information, the process of Fig. 7 can be repeated, although the steps of performing the DNS request do not need to be performed again, since the information between the two devices has already been stored. *Id.* at 15:58-67.

149. Finally, I note that the '785 patent specification describes that the three addresses are not always returned, for instance, in the case where “the destination machine has a public address (and, thus does not use a routing director) then step 656 will only include returning the public IP address for the destination.” *Id.* at 14:40-44.

B. Prosecution History

150. The '785 patent application was filed on March 31, 2003 with a total of 109 claims. During prosecution, the examiner rejected the '785 patent claims on §101, 102, and 103 grounds based on several prior art references. EX1002 ('785 FH), 145 (non-final rejection); 202 (final office action); 393 (non-final rejection); 613 (final office action); 689 (rejection).

151. Of particular relevance to allowance, in an office action dated November 22, 2006, the examiner rejected all claims as anticipated by U.S. Patent Publication No. 2003/0041136 (Cheline). *Id.* at 143-162. In response, the applicant amended the claims by adding specifically that the network traffic router “uses router data to route traffic between members of the virtual community,” and that “a first member of the virtual community communicates with another member in the virtual community by querying the DNS server to determine a virtual network address and a private network address for the other member, and to determine a public address of the network traffic router.” *Id.* at 172-187.

152. In its response, the applicant argued that “the client 102(1) of Cheline does not communicate with the DNS server 120 to obtain a virtual network address and private network address of another client, such as client 102(2), or to obtain a public network address of the service provider system 146, which is asserted by the Examiner to be a network traffic router as claimed.” *Id.* at 190-191.

153. On September 1, 2010, the examiner issued another office action, stating that claims 1-15, 17-24, 26-36, 41-43, and 93-109 were allowed, while claims 53-66 and 68-92 were rejected as anticipated by U.S. Patent No. 6,631,416 (Bendinelli). *Id.* at 613-618. The examiner’s reasons for allowance included the finding that “the prior art was not found to teach or suggest returning a public network address of a route director, a private network address for a destination device, and a virtual network address that corresponds to the destination device in response to a DNS request in the context of claims 1, 32, and 93,” and that “[s]pecifically, no examples were found in the prior art of record in which both a private address and a virtual address of a destination device would be returned.” *Id.* at 616-617. This determination by the examiner was critical to the claims of the ’785 patent being allowed. *Id.*

154. The applicant subsequently filed an amendment on October 21, 2010, revising some of the rejected claims (claims 53-55, 64, and 80) to include a similar DNS request and corresponding response with a triplet of addresses. *Id.* at 658-683.

155. Following another rejection by the examiner, this time rejecting claims 1-15, 17-24, 26-36, 41-43, and 80-92 as invalid under 35 U.S.C. §101 while finding that the remaining claims contained allowable subject matter, the applicant filed another amendment clarifying that the recited virtual network manager and/or route director are implemented using a memory and a processor of a computing device to traverse the examiner's §101 rejection. *Id.* at 687-691. In response, the applicant filed an amendment on December 2, 2010, to clarify that the recited virtual network manager and/or route director are implemented using a memory and processor of a computing device. *Id.* at 723-750. A notice of allowance was then issued on January 14, 2011. *Id.* at 760-766.

IX. THE CLAIMS

156. The table below includes identifiers for several representative independent claims and limitations of the '785 patent, which are referenced throughout this Declaration.

'785 Patent Claims
[1.pre] A virtual network system, comprising:
[1.1] a virtual network manager implemented with a first device memory and a first device processor of a first computing device,
[1.2] the virtual network manager configured to register devices in a virtual network that is defined by a domain name,

'785 Patent Claims

[1.3] each device in the virtual network being identified to the other devices by a virtual network address that is unique for each device and not directly routable via a public network, the virtual network manager further configured to distribute a virtual network address to a device when the device is registered in the virtual network;

[1.4] a route director implemented with a second device memory and a second device processor of a second computing device, the route director configured to communicate data between the devices that are registered in the virtual network, the data being communicated as encapsulated packets from a source device to a destination device, an encapsulated packet including a first virtual network address that corresponds to the source device and a second virtual network address that corresponds to the destination device; and

[1.5] the virtual network manager further configured to receive a DNS request from the source device, and return a public network address of the route director, a private network address for the destination device, and the second virtual network address that corresponds to the destination device.

[30.pre] A virtual network manager, comprising:

[30.1] a network interface configured for data communication via a virtual network that is defined by a domain name having an associated public network address;

[30.2] a memory and a processor to implement a register module configured to register devices in a virtual network, the register module further configured to:

[30.3] receive a registration request from an agent associated with a device; distribute a virtual network address to the device when the device is registered in the virtual network, the device being identified to other devices in the virtual network by the virtual network address; and

[30.4] a DNS server for the virtual network, the DNS server configured to receive a DNS request from a first device in the virtual network, and return a network address associated with a network route director, a private network

'785 Patent Claims
address associated with a second device in the virtual network, and a virtual network address associated with the second device.
[38.pre] A virtual network system, comprising:
[38.1] a computing device that includes at least a memory and a processor configured to implement a network manager of a virtual network that is defined by a public domain name,
[38.2] the network manager configured to distribute virtual network addresses to devices that register as members in the virtual network, each device in the virtual network being identified to the other devices by a virtual network address associated with the device;
[38.3] a first virtual network agent associated with a first device that is registered as a member in the virtual network; at least a second virtual network agent associated with at least a second device that is registered as a member in the virtual network;
[38.4] a route director configured to route communications between the first device and the at least second device in the virtual network via the respective first and second virtual network agents, the communications configured for routing as encapsulated packets that include a first virtual network address that is not directly routable corresponding to the first device and a second virtual network address that is not directly routable corresponding to the at least second device; and
[38.5] the network manager includes a DNS server configured to provide authoritative responses for DNS queries in the virtual network, the DNS server further configured to receive a DNS query from the first device and return a network address of the route director, a network address of the second device, and the virtual network address of the second device.
[48.pre] A computer-implemented method, comprising:

'785 Patent Claims
[48.1] receiving registration requests from devices that request to be registered as members of
[48.2] a virtual network that is defined by a domain name having an associated public network address in a public network,
[48.3] each of the devices having an associated private network address; distributing a virtual network address to a device to register the device as a member in the virtual network, each device in the virtual network being identified to the other devices by the virtual network address that is associated with the device;
[48.4] routing communications between the devices that are registered in the virtual network, the communications being routed as encapsulated packets from a source device to a destination device, an encapsulated packet including a first virtual network address that corresponds to the source device and a second virtual network address that corresponds to the destination device; and
[48.5] transmitting a response to a DNS request received from one of the devices that are the members in the virtual network, the response to the DNS request including a public network address of a route director that registers the devices, a public network address of the destination device, and the second virtual network address that corresponds to the destination device.
[62.pre] One or more processor readable storage media devices comprising processor readable code that, if executed by a computer device, implements a virtual network manager to:
[62.1] receive registration requests from devices that request to be registered as members of
[62.2] a virtual network that is defined by a domain name having an associated public network address in a public network,
[62.3] each of the devices having an associated private network address; distribute a virtual network address to a device to register the device as a member

'785 Patent Claims
in the virtual network, each device in the virtual network being identified to the other devices by the virtual network address that is associated with the device;
[62.4] manage communications routed between the devices that are registered in the virtual network, the communications routed as encapsulated packets from a source device to a destination device, an encapsulated packet including a first virtual network address that corresponds to the source device and a second virtual network address that corresponds to the destination device; and
[62.5] transmit a response to a DNS request received from one of the devices that are the members in the virtual network, the response to the DNS request including a public network address of the virtual network manager, a public network address of the destination device, and the second virtual network address that corresponds to the destination device.
[75.pre] A virtual network system, comprising:
[75.1] a computing device that includes at least a memory and a processor configured to implement a virtual network manager having a network interface coupled to a virtual network,
[75.2] the virtual network manager including at least one virtual community definition that is defined by a domain name having an associated public network address and
[75.3] a user set of one or more devices that are registered in the virtual network, each device in the virtual network being identified to the other devices by a virtual network address that is associated with the device, the virtual network manager configured to exchange virtual network information with the one or more devices of the user set, the virtual network being accessible by devices in the user set and devices outside of the user set,
[75.4] and the virtual network manager further configured to receive a DNS request from a source device, and return a public network address of a route director, a private network address for a destination device, and a virtual network address that corresponds to the destination device.

X. OVERVIEW OF THE PRIMARY PRIOR ART REFERENCES

157. I first note that the prior art references analyzed in the grounds below are analogous to the '785 patent because they are within the same field of endeavor and/or reasonably pertinent to one or more problems addressed by the '785 patent. For example, all references relate to network routing between multiple private/public IP addresses.

A. Caronni I

158. Caronni I is a US patent to Caronni *et al.*, entitled “System and Method for Separating Addresses from the Delivery Scheme in a Virtual Private Network.” EX1003 (Caronni I) at p. 1. Caronni I describes a system that establishes “a virtual network on top of current IP network delivery schemes,” where the “virtual network uses a separate layer to create a modification to the IP packet format that is used to separate network behavior from addressing.” *Id.* at 3:15-21. The Caronni I invention thus allows “any type of delivery scheme may be assigned to any address or group of addresses.” *Id.* at 3:21-23.

159. Caronni I describes a Supernet, “which is a private network that uses components from a public-network infrastructure.” *Id.* at 4:36-39. Beneficially, a Supernet “allows an organization to utilize a public-network infrastructure for its enterprise network so that the organization no longer has to maintain a private network infrastructure.” *Id.* at 4:40-44. A further Supernet advantage is its

portability capability, where “a Supernet is not geographically restrictive, so a user may plug their device into the Internet from virtually any portal in the world and still be able to use the resources of their private network in a secure and robust manner.” *Id.* at 4:48-51.

160. Figure 3 (reproduced below) illustrates a system using the inventive concepts of Caronni I where the Supernet public network infrastructure is the Internet. *See id.* at 4:62-66.

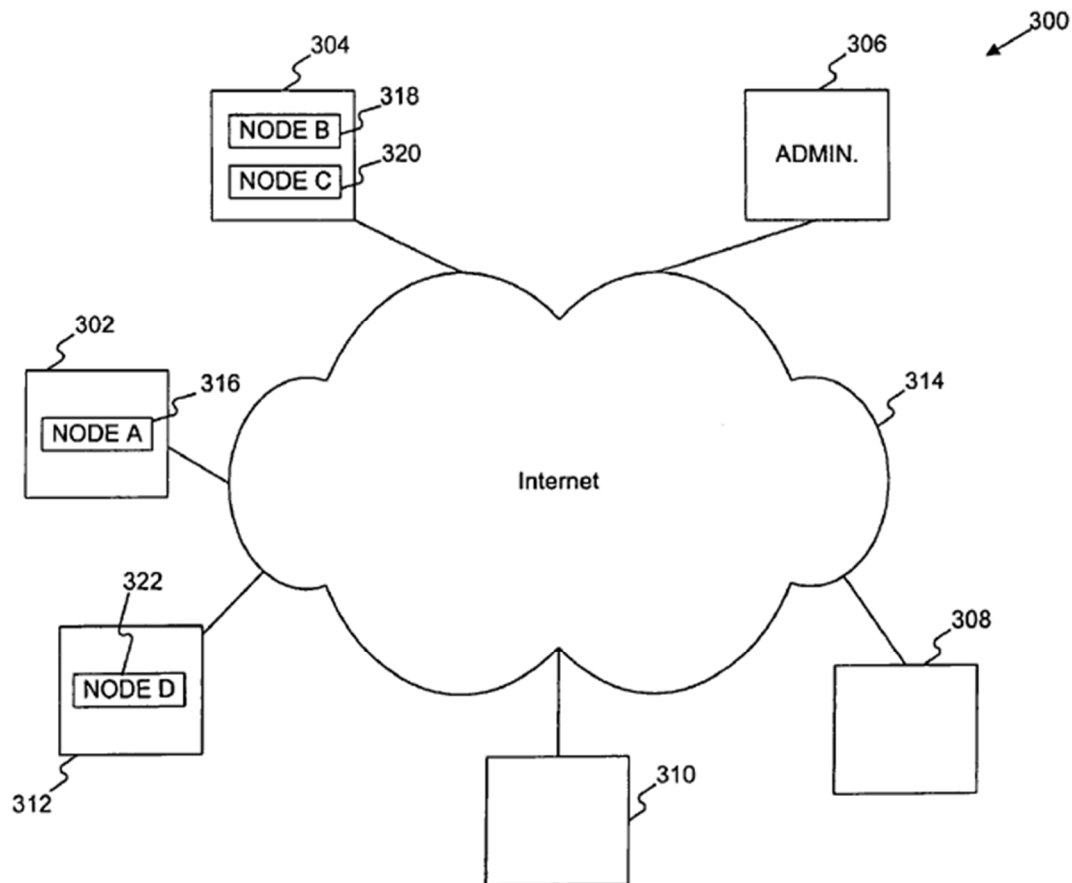


Fig. 3

Id. at Fig. 3.

161. Figure 3 further shows the Supernet includes various computers or devices (302-312) that are connected to the Internet. *Id.* at 4:62-66. Finally, Figure 3 shows that these computers or devices contain nodes (316, 318, 320, and 322).

162. Caronni I discloses that these nodes are endpoints that provide for communication over the Supernet:

A Supernet's infrastructure uses components from the Internet because devices **302**, **304**, and **312** contain nodes that together form a Supernet and that communicate by using the infrastructure of the Internet. These nodes **316**, **318**, **320**, and **322** are communicative entities (e.g., processes) running within a particular device and are able to communicate among themselves as well as access the resources of the Supernet in a secure manner. When communicating among themselves, the nodes **316**, **318**, **320**, and **322** serve as end points for the communications.

Id. at 4:66-5:9.

163. Caronni I also discloses an administrative node/machine 306, also shown in Figure 3. "The Supernet also includes an administrative node **306** to administer to the needs of the Supernet." *Id.* at 5:11-13. I note that Caronni I refers to item 306 as both an administrative node and an administrative machine interchangeably.

164. The Caronni devices (example 302) and administrative machine (306) are shown in greater detail in Figure 5 (reproduced below), where these devices communicate over the Internet (314).

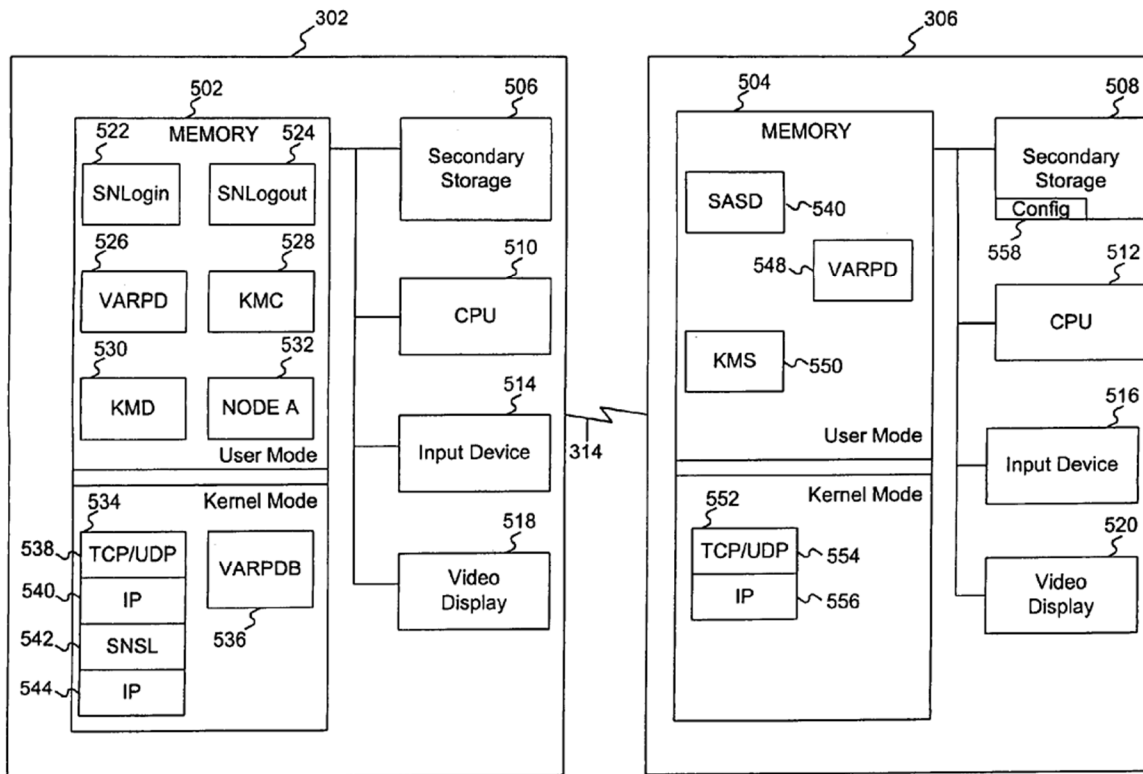


Fig. 5

165. Caronni I discloses the creation and use of a Supernet configuration file that specifies essential aspects of the Supernet. “To configure a Supernet, a system administrator creates a configuration file 558 that is used by SASD 540 when starting or reconfiguring a Supernet.” *Id.* at 8:1-3. “After the configuration file is created, it is used to start a Supernet.” *Id.* at 8:16-17. Central components of the administrative machine 306 include the SASD process 540, VARPD 548, and KMS 550.

166. The SASD process provides for “[s]tarting or reconfiguring a Supernet.” *Id.* at 8:1-3. Furthermore, the SASD process “represents a Supernet”

and “both authenticates nodes and authorizes nodes to join the Supernet.” *Id.* at 7:3-5.

167. Generally, a VARPD “maintains the mappings from real addresses to virtual addresses.” *Id.* at 7:3-5. I note that for a given Supernet, “one VARPD runs on each machine.” *Id.* at 7:21-22. Furthermore, a VARPD may play two roles:

First, a VARPD may act as a server by storing all address mappings for a particular Supernet into its associated VARPDB. Second, regardless of its role as a server or not, each VARPD assists in address translation for the nodes on its machine. In this role, the VARPD stores into its associated VARPDB the address mappings for its nodes, and if it needs a mapping that it does not have, it will contact the VARPD that acts as the server for the given Supernet to obtain it.

Id. at 7:22-30.

168. A POSITA would have understood that in either role, the address translation function of the VARPD is based on a query/response exchange:

When the VARPD is queried for the address translation (described below), the VARPD provides the real IP address associated with the e-mail address for delivery.

Id. at 7:59-62.

169. The KMS enables encryption for Supernet communications by “performing key management by generating a new key every time a node joins a channel and by generating a new key every time a node leaves a channel” where there is “one KMS per channel in a Supernet.” *Id.* at 7:64-67; *see* §VII.B.4 above.

170. I discuss each of these components and the role of the configuration file in more detail as they relate to the challenged claims.

B. Caronni II

171. Caronni II is a US patent to Caronni *et al.*, entitled “System and Method for Using Data Encapsulation in a Virtual Network.” EX1004 (Caronni II) at p. 1. Caronni II describes a system that extends the functionality of a virtual network to devices located behind packet filtering devices by encapsulating messages sent to a virtual destination address. “The illustrative embodiment of the present invention provides a method of encapsulating messages sent to a virtual destination address in higher level protocols.” *Id.* at 3:49-37. “The encapsulation of the messages in the specified higher level protocol enables the messages to reach their intended destination despite the presence of packet filtering devices on the edge of a network.” *Id.* at 3:54-57; *see* §VII.B.4 above.

172. Caronni II discloses a virtual network that is “a logical network overlaid on a physical network” that “allows a virtual IP (Internet Protocol) address to be assigned to individual processes or applications running on a computer other electronic device.” *Id.* at 1:13-17. Virtual addresses are used as destination addresses for messages sent over the virtual network. “Messages may be sent to a process at a virtual IP address separate from the real IP address used by the network device upon which the process is executed.” *Id.* at 1:21-23.

173. To facilitate the Caronni II virtual destination addressing feature, a virtual address resolution facility is disclosed, where the “virtual address resolution

facility is used to resolve a virtual IP address into a real IP address.” *Id.* at 2:54-55. Further, this address resolution facility employs a Virtual Address Resolution Protocol (VARP) lookup table. *Id.* at 4:11-14.

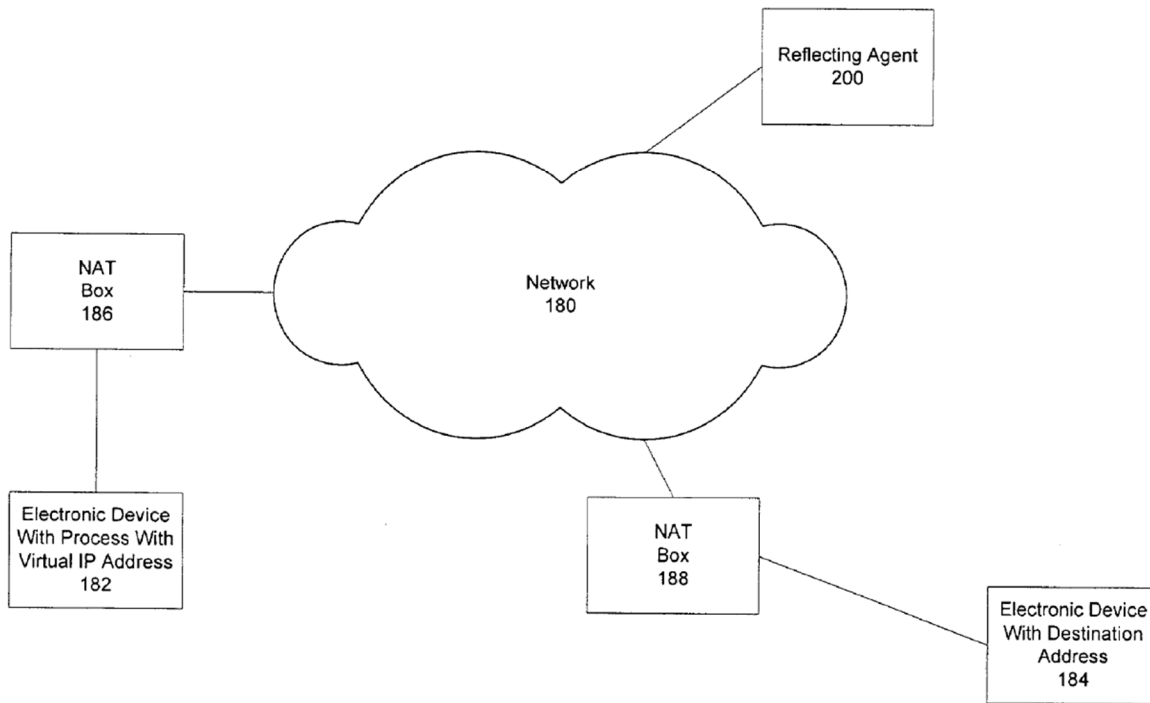
174. Caronni II describes edge devices (using NAT) that are problematic to use with virtual networks:

Similarly, if the edge device is a device performing Network Address Translation (a “NAT box”), the NAT box rewrites all outgoing packets from an end user in the interior of the network to make them look like they came directly from the NAT box, and remembers that any traffic coming back from the particular destination address must be mapped back to the originating internal device. Consequently, the responding devices think they are responding to the sending device when they are actually responding to an edge device. In such a case, the packets intended for the virtual destination address on the interior of the physical network may be dropped and not reach their intended destination.

Id. at 2:15-27.

175. In the case where a message is sent from a virtual address that is behind a NAT device to another virtual device that is also behind a NAT device, Caronni II describes a “reflecting agent” to facilitate the message transfer. This scenario is shown below in Figure 5 (reproduced below).

Figure 5



Id. at Fig. 5.

176. Here, with both the source device and destination device behind different NAT edge devices, the source device is unable to directly address the destination device because the destination device address may not be routable over the public network 180 because it may be a private address. A reflecting agent acts as an intermediary for this message traffic. To allow the use of a reflecting agent, Caronni II describes adding the address of the reflecting to the VARP table:

When the destination address is also behind a NAT box, the originator of the message is unable to directly address the destination (since the address of the destination may not be routable in the public Internet). In such a case, the present invention adds an entry to the VARP table for a third-party reflecting agent located at an address outside the NAT box.

Id. at 8:14-21.

A reflecting agent 200, the address of which has been added to a VARP table, acts as an intermediary to allow a connection between the originating process 182 and destination address 184.

Id. at 8:35-38.

C. Hipp

177. Hipp, is a US patent to Hipp *et al.*, entitled “Virtual Network Environment.” EX1005 (Hipp) at p. 1. Hipp describes “a system, method, and computer program product for grouping a set of applications into a virtual network environment and isolating the application from other applications in other virtual network environments,” where the “present invention provides isolation at the application level, rather than at the host level.” *Id.* at 2:7-12. This virtual environment thus effectively serves to “isolate groups of application such that they may be located as needed on a computer network without risk of interference with other applications.” *Id.* at 2:1-3.

178. Hipp discloses that the Virtual Network Environment (VNE) of the present invention is defined by a collection of IP addresses and that these IP addresses may be virtual:

An application IP address is an IP address associated with an application. It can be a virtual IP address, associated solely with an application, or it can be a real IP address, associated with an interface on the host where the application resides. The application IP address is the IP address the application uses as the local address for all communications. The present invention may be utilized in both virtual and interface IP addresses. The Virtual Network Environment (VNE)

is defined by a collection of IP addresses related to applications that are contained within the VNE or have the potential of being placed in the VNE.

Id. at 2:58-3:2.

179. Figure 1 (reproduced below) illustrates in high level block diagram form the overall structure of the present invention as used in connection with a global computer network such as the Internet.

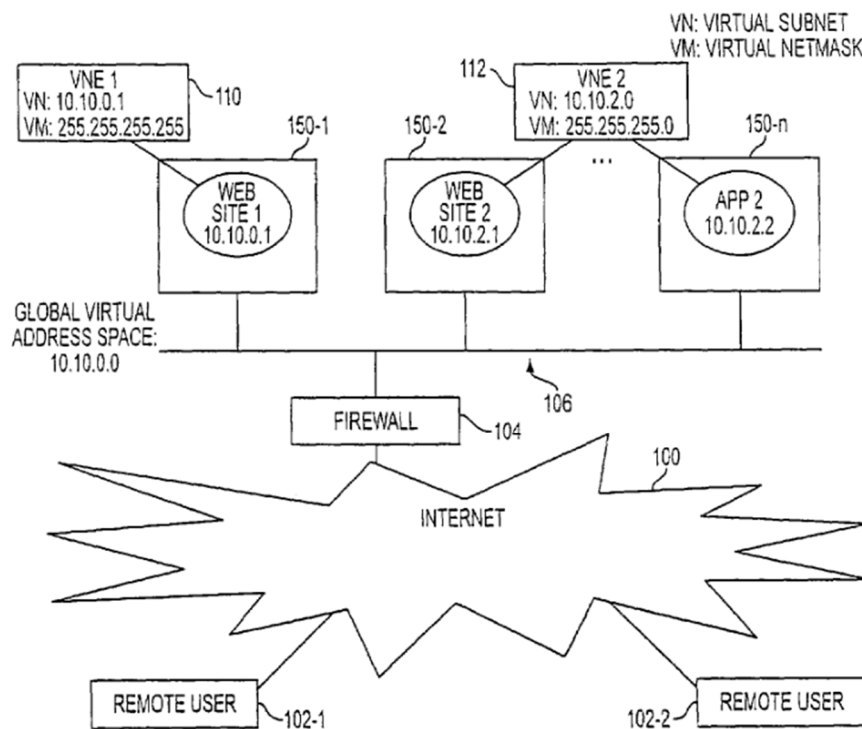


FIG. 1

Id. at Fig. 1.

180. Here, computer network 106, located behind a firewall 104, is a private network of computers 150-1, 150-2, etc., where the computers “run various

applications, as well as host web sites for access by remote users 102.” *Id.* at 2:60-61.

181. Aspects of the Hipp invention may be seen in the VNE 1 110 and VNE 2 112 virtual network environments implemented on the computer network 106. For example, a “VNE is defined by subnet of addresses contained within the VNE,” where “all applications within the subnet 10.10.2.0 comprise a VNE.” *Id.* at 3:26-28. As shown in Figure 1, the subnet 10.10.2.0 is designated as VNE 2. Further, VNE 2 “would include the addresses 10.10.2.0 through 10.10.2.255.” *Id.* at 3:30-31. Hipp explains the aforementioned isolation aspects of a VNE may be demonstrated by “an application with IP address 10.10.2.2 would be able to communicate with an application at address 10.10.2.60, but not at 10.10.0.1.” *Id.* at 3:31-34.

182. With respect to the virtual nature of a VNE, Hipp discloses that network identities may be virtualized:

Virtualization of network identity is achieved by assigning a unique virtual IP address and virtual hostname to a group of processes that make up the application instance which the instance keeps throughout its execution. This virtual network identity stays with the application instance regardless of which node the application is running on. The framework, in essence, provides a mechanism to create this virtual network identity (VNI) around the application using the virtual network parameters assigned to it. In one embodiment, the virtual network parameters include an IP address and hostname. The framework 200 ensures that the applications instance uses the virtual network parameters, transparently, so that it can be moved across machines, without modifications to the application.

Id. at 6:8-21.

183. Hipp further discloses that these virtual network identities may be included in a DNS resolution capability:

The virtual hostname resolves to the virtual IP address for both the applications registered with the VNI framework as well as those that are not registered. This may require configuration of a name service or OS host configuration files. For example, if an application instance used a virtual IP address of 10.10.0.1 and a virtual hostname of host 1055, the standard hostname to IP address resolution mechanisms (e.g. DNS or the /etc/hosts file) would have to be preconfigured to resolve a query of host 1055 to IP address 10.10.0.1.

Id. at 6:22-31.

D. RFC-1383

184. *Request For Comment (RFC) 1383: An Experiment in DNS-Based IP Routing* by Christian Huitema outlines a proposal for utilizing Domain Name System (DNS) records for IP routing. The approach described in this RFC seeks to help solve the issue of “routing explosion” on the Internet caused by the increasing number of networks, resulting in larger routing tables, more data in route exchanges, and slower responses to routing events. EX1006 (RFC-1383) at 1.

185. In turn, RFC-1383 offered “a scheme that allows for simple routing,” and is “complementary with the classic hierarchical routing approach,” while providing “an easy to implement and low cost solution for ‘multi-homed’ domains.” *Id.* at 2. Specifically, the RFC-1383 solution is described as “a generalization of the ‘MX record’ scheme used for mail routing.” *Id.*

186. Similar to the aforementioned mail routing model that uses MX (Mail eXchange) records, the protocol introduces “RX records” for routing IP packets by associating domain names with preferred gateways by means of “source routing” or “tunneling,” or another form of encapsulation protocol. *Id.* at 4. To implement this routing, a modified DNS lookup is suggested: the source or an early relay makes the routing decision by querying the DNS for RX records, which list gateways with preference levels. This approach avoids hop-by-hop routing updates across the network and is scalable. *Id.*; *see* §VII.B.4 above.

187. RFC-1383 discusses mechanisms to use its DNS records. “Source directed routing, as described above, can be implemented through one of two techniques: source routing, or a form of encapsulation protocol. For the sake of simplicity, we will use source routing, as defined in [1]: we don’t have to define a particular tunnelling protocol, and we don’t have to require hosts to implement a particular encapsulation protocol.” EX1006 (RFC-1383) at 6. A POSITA would have understood that when RFC-1383 returns one or more extra addresses in response to a DNS query, those could be easily used as the tunneling/encapsulation addresses to enable, for example, VPN and IPsec services. *See also* §VII.B above.

188. RFC-1383 describes a network interface integrated with a standard IP router, working alongside a DNS query manager. EX1006 (RFC-1383) at 8, 12. When an ICMP (Internet Control Message Protocol) message is received, the query

manager updates the local routing table to ensure that any new packets bound for the specified destination are routed through the real-time forwarder. *Id.* at 8. Simultaneously, the query manager sends a DNS request to retrieve the RX records for the destination. *Id.* at 11. Upon receiving the response, it selects a gateway and provides this information to the real-time forwarder. *Id.* at 12.

189. When the real-time forwarder receives a packet, it checks if a gateway for the destination is available. *Id.* at 12-13. If so, it inserts the necessary source routing information and forwards the packet—either through the general IP routing program or directly to the network interface linked to the intermediate gateway.

190. Notably, RFC-1383 discloses (a) a way to store and return multiple IP addresses in response to a single query, and (b) identify an address of the destination in the domain part of the DNS request and response.

191. RFC-1383 proposes using “TXT” DNS records to store that information. “This [TXT] record is designed for easy general purpose extensions in the DNS, and its content is a text string. Each RX record will contain three fields:

- A record identifier, ‘RX,’ to distinguish it from other experimental uses of the ‘TXT’ record.
- A cost indicator, encoded on up to 3 numerical digits.
- An IP address, encoded as a text string following the ‘dot’ notation.” *Id.* at 11.

192. RFC-1383 shows an example record (*id.*):

domain	type	record	value
*.27.32.192.in-addr.arpa	IP	TXT	RX, 10, 10.0.0.7

193. RFC-1383 explains the table shows: “that [] for all hosts whose IP address starts by the three octets ‘192.32.27’ the IP host ‘10.0.0.7’ can be used as a gateway, and that the preference value is 10.” *Id.* at 12. The TXT records described in RFC-1383 are simply a domain field containing a destination address, and a string value comprising comma-delimited fields, including an IP address.

XI. CLAIM CONSTRUCTION

194. For purposes of this declaration, I’ve been asked to assume that the time of the alleged invention is March 31, 2003.

A. Claim Construction Generally

195. I understand that, for purposes of my analysis in this inter partes review proceeding, the terms appearing in the patent claims should be interpreted according to their “ordinary and customary meaning” in view of the specification and prosecution history under the Phillips standard. *See Phillips v. AWH Corp.*, 415 F.3d 1303 (Fed. Cir. 2005); 37 C.F.R. §42.100. In determining the ordinary and customary meaning, I understand that the words of a claim are first given their plain meaning that those words would have had to a POSITA in the art at the time of the alleged invention. I also understand that the structure of the claims, the specification

and file history may be used to better construe a claim. Moreover, I understand that even treatises and dictionaries may be used, albeit under limited circumstances, to determine the meaning attributed by a POSITA in the art to a claim term.

B. Claim Construction Means Plus Function Terms

196. I understand that under the applicable version of 35 U.S.C. §112 ¶6, “an element in a claim for a combination may be expressed as a means or step for performing a specified function without the recital of structure, material, or acts in support thereof, and such claim shall be construed to cover the corresponding structure, material, or acts described in the specification and equivalents thereof.”

197. I understand that expressing an element in a claim in terms of a “means for ...” leads to a presumption that 35 U.S.C. §112 ¶6 applies. I also understand that even if an element is not expressed using the term “means,” it may still be considered a means-plus-function limitation, if the term fails to “recite sufficiently definite structure” or else recites “function without reciting sufficient structure for performing that function.” That a claim term lacks the word “means” does not create a “strong” presumption that the claim term is not subject to 35 U.S.C. §112 ¶6. The standard for determining whether the words of the claim are understood by POSITA in the art to have a sufficiently definite meaning as the name for structure.

198. I understand that construing a claim limitation that is a means-plus-function term involves two steps. The first step is to identify the claimed function.

The second step is to identify the structures that are described in the specification. I understand that the means-plus-function claim term is limited to those structures described in the specification that correspond to the claimed function, and their equivalents.

C. All Claims: “virtual network address”

199. All of the claims of the ’785 patent recite a “virtual network address” that is assigned to a device once it is registered in the virtual network. The ’785 patent explains a “virtual network address” is a name or address that can be used to identify and send communications to other devices in the virtual network. *See* EX1001 (’785 patent) at 6:1-7, 31:24-43.

D. Claim 30 (“register module”)

200. Claim 30 recites a “register module” for performing certain functions. “Module” is a nonce word that substitutes for “means” for purposes of §112, ¶6. *See Williamson v. Citrix Online, LLC*, 792 F.3d 1339, 1350 (Fed. Cir. 2015) (en banc). The following table highlights these terms in the relevant claims:

Claim	Claim Analysis
[30.2-30.4] a memory and a processor to implement] a register module configured to register devices in a virtual network, the register module further configured to: receive a registration request from an agent associated with a device; distribute a virtual network address to the device when the device is registered in the virtual network, the device being identified to other devices in the virtual network by the virtual network	Nonce word for “means” function without recitation of structure to perform it

201. I note that in the litigation, Patent Owner asserts Claims 30, 35, and 37, and reads the “registration module” fairly broadly against commonly and widely used open-source software known as Kubernetes. *See* EX1025 (’785 claim chart) at Claim 30. Patent Owner contends this term is not a means-plus-function limitation.

202. Means-plus-function claims are construed by identifying the claimed function and the corresponding structure in the specification for performing it. *See Williamson*, 792 F.3d at 1351. The table below provides these constructions.

Claims	Function	Structure in ’785 Specification
30	“receiving registration request from device’s agent and distributing a virtual network address to the device when it registers with the virtual network.”	“The Registration exchange between the member agent and the VCN Manager is shown in FIG. 12A. When a Member Agent 565 attempts to register at step 1200, Member Agent 565 will send a registration request packet 1202 to the VCN Manager 510 to start registration. The packet is included in an HTTP wrapper as illustrated at 1204. The registration packet will carry the client Fully Qualified Domain Name (FQDN) and a Diffie-Hellman Key Exchange

Claims	Function	Structure in '785 Specification
		request to the VCN Manager. The packet further includes packet version information, type and length information, a user authenticator, the length of FQDN in octets, a VCN name offset, the member's FQDN in DNS format, the length of Diffie-Hellman value in octets, and the members initial Diffie-Hellman value.” EX1001 ('785 patent), 18:18-29.

E. Claims 35-37, 77-78: “join module”

203. Claims 35-37, 77, and 78 recite a “join module” performing the below described functions. *See Williamson*, 792 F.3d at 1350 (“Module” is a nonce word that substitutes for “means” for purposes of §112, ¶(6)). Claim set 25-37, discussed in the table below, is illustrative.

Claims	Claim Analysis
[35] The virtual network manager of claim 30 further comprising a join module configured to receive a join request from the agent associated with the device to indicate that the device is connected for data communication within the virtual network, the join module further configured to receive a leave request from the agent associated with the device to indicate that the device will be disconnected from data communication within the virtual network.	Nonce word for “means” function without recitation of structure to perform it
[36] The virtual network manager of claim 35 wherein the join module is further configured to provide virtual network addresses to the devices that are registered in the virtual network.	
[37] The virtual network manager of claim 35 wherein the join module is further configured to maintain data to	

Claims	Claim Analysis
associate a virtual network address with a device in the virtual network.	
[77] The system of claim 75 wherein the virtual network manager includes a member join module.	
[78] The system of claim 77 wherein the member join module provides a virtual network address to a device that is registered as a member of the network.	

204. Means-plus-function claims are construed by identifying the claimed function and the corresponding structure in the specification for performing it. *See Williamson*, 792 F.3d at 1351. The table below provides these constructions.

Claims	Function	Structure in '785 Specification
35	“receiving join and leave requests from the device or its agent indicating that the device is connected or will be disconnected from data communication, respectively”	The Member Join process is illustrated in FIG. 13 and the leave process in FIG. 14.

Claims	Function	Structure in '785 Specification
		Fig. 13 Fig. 14 See also EX1001 ('785 patent), 20-21 (for description of figures).
36, 78	“providing virtual network addresses to	<i>Id.</i>

Claims	Function	Structure in '785 Specification
	the devices that are registered in the virtual network”	
37	“maintaining data to associate a virtual network address with a device in the virtual network.”	<i>Id.</i>
77	<i>N/A</i>	<i>N/A</i>

F. Other Claims

205. No express construction of the other challenged claims is necessary in this IPR because the prior art renders these claims obvious under any reasonable construction. Petitioners do not concede that no construction of the claims may ever be necessary or any other claim construction argument. Should any subsequent discussion of claim interpretation arise here or in the parallel district court proceeding, Petitioners reserve the right to propose constructions and respond to any construction advanced by Patent Owner, the Board, and/or the District Court.

206. For purposes of this declaration, I’ve been asked to assume that the time of the alleged invention is March 31, 2003.

XII. GROUND 1: Claims 1, 30, 35, 36, 37, 38, 48, 62, 75, 77, and 78 are obvious in view of Caronni I in view of Caronni II and in further view of Hipp

A. The Independent Claims

207. The six independent claims of the '785 patent (claims 1, 30, 38, 48, 62, and 75) recite substantially overlapping subject matter. For ease of discussion, I address the independent claims together. Furthermore, I group related limitations together.

208. These claims generally disclose a virtual network system or manager, comprising a virtual network with a public domain name, a registration feature or module to register devices on the network, a route director to route data traffic between the devices on the network, and a DNS server to exchange network addresses of the devices on the network upon request. These limitations, as shown further below, would have been obvious to a POSITA before the time of filing of the '785 patent, in view of the disclosures in Caronni I, Caronni II, and Hipp.

1. virtual network system or manager

209. I note that in the remainder of my declaration, grey text in tables indicates a limitation discussed in another section.

Claim	Limitations
[1.pre, 1.1]	A virtual network system, comprising: a virtual network manager implemented with a first device memory and a first device processor of a first computing device,

Claim	Limitations
[30.pre]	A virtual network manager, comprising:
[38.pre, 38.1]	A virtual network system, comprising: a computing device that includes at least a memory and a processor configured to implement a network manager of a virtual network that is defined by a public domain name
[48.pre]	A computer-implemented method, comprising:
[62.pre, 62.1]	One or more processor readable storage media devices comprising processor readable code that, if executed by a computer device, implements a virtual network manager to:
[75.pre, 75.1]	A virtual network system, comprising: a computing device that includes at least a memory and a processor configured to implement a virtual network manager having a network interface coupled to a virtual network,

210. The preambles of the independent claims, to the extent limiting, and additional claim limitations recite a virtual network system or virtual network manager. I note that the '785 patent explains that “an exemplary system for implementing a VCN in accordance with the invention comprises a VCN Manager 510, a Network Route Director 520 and/or Private Route Director 530, and a number of Member Agents 565.” EX1001 ('785 patent) at 10:24-27. Further, the '785 patent explains that “[t]he VCN Manager 510 is a central server or server cluster providing management, connection, and security services for the VCN.” EX1001 ('785 patent) at 10:66-11:1.

211. Caronni I discloses a virtual network system as a “virtual network on top of current IP network delivery schemes,” where the “virtual network uses a separate layer to create a modification to the IP packet format that is used to separate network behavior from addressing.” EX1003 (Caronni I) at 3:17-21. The system of Caronni I provides “addressing functionality that easily integrates and supports existing infrastructure services while at the same time allows for multiple delivery schemes.” *Id.* at 3:8-11. Caronni I refers to this system as a Supernet. “Methods and systems consistent with the present invention overcome the shortcomings of existing networks by establishing a “Supernet,” which is a private network that uses components from a public-network infrastructure.” *Id.* at 4:36-40.

212. Further, Caronni I discloses a virtual network manager as an administrative node. “The Supernet also includes an administrative node 306 to administer to the needs of the Supernet.” *Id.* at 5:11-13.

213. Caronni I discloses devices, including an administrative node, that contain a memory and a processor. “Each device contains similar components, including a memory 502, 504; secondary storage 506, 508; a central processing unit (CPU) 510, 512,” where “[m]emory 504 of administrative machine 306 includes the SASD process 540, VARPD 548, and KMS 550 all running in user mode.” *Id.*, 6:48-55. Further, “[m]emory 504 also contains a VARPDB 551 and a TCP/IP protocol stack 552 that are executed by CPU 512 running in kernel mode.” *Id.* at 6:62-64.

214. I note that “user mode” indicates the CPU prevents programs from “directly manipulating the hardware components,” while kernel mode indicates the CPU allows programs to “manipulate the hardware components.” *Id.* at 6:57-62. The administrative node memory and processor are shown below in annotated Figure 5.

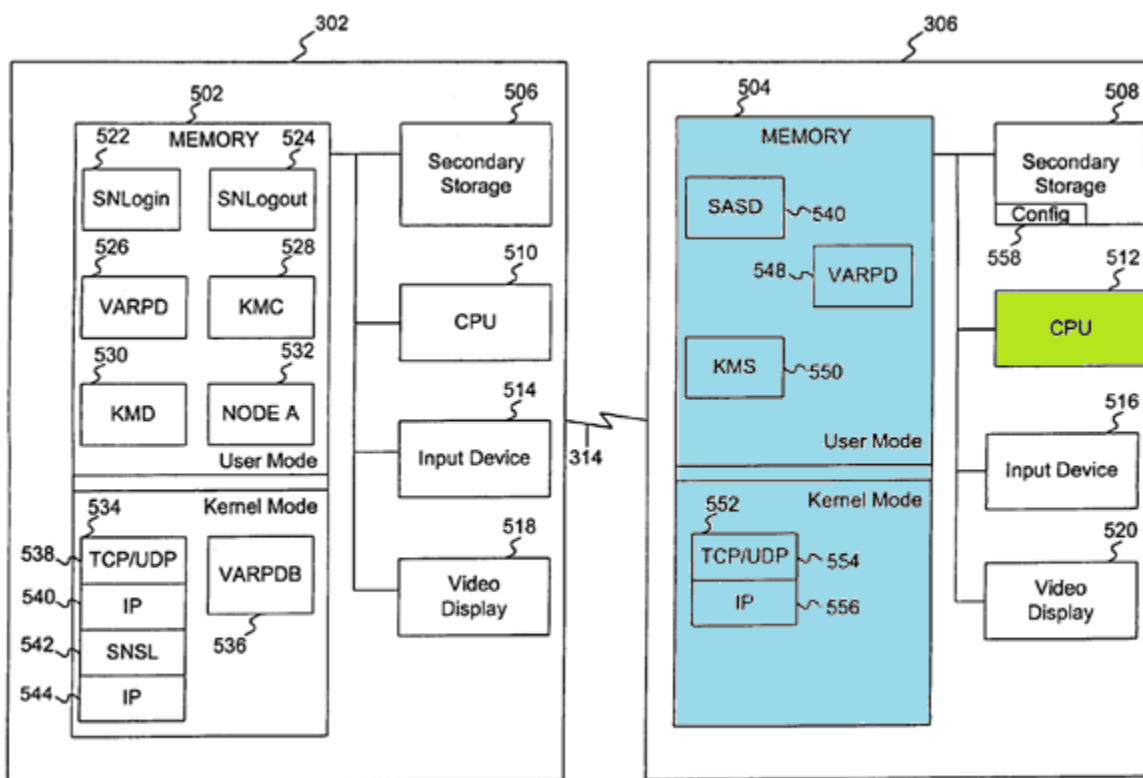


Fig. 5

Id. at Fig. 5 (annotated).

215. Caronni II discloses a virtual network system as an “embodiment of the present invention provides a method of extending the functionality of a virtual network,” where “[m]essages intended for a virtual destination address located on a network equipped with a device performing packet filtering, network address

translation or a similar function on the edge of the network (an ‘edge device’), are encapsulated in higher level protocols prior to being sent to the edge device.” EX1004 (Caronni II), 2:31-38; *see* VII.B.4 above.

216. Further, Caronni II discloses a virtual network manager as a virtual address resolution facility that “is software used to register, store and resolve virtual address information for processes and applications executing on the virtual network.” EX1004 (Caronni II) at 4:6-9.

217. A POSITA would understand that the combination of Caronni I, Caronni II, and Hipp discloses the above limitations. Namely, a virtual network system and a memory and a processor configured to implement a virtual network manager. A POSITA would have been motivated to combine Caronni I, Caronni II, and Hipp for the reasons discussed in §XII.B below.

2. a virtual network defined by a domain name having an associated public network address

Claim	Limitations
[1.2]	the virtual network manager configured to register devices in a virtual network that is defined by a domain name
[30.1]	a network interface configured for data communication via a virtual network that is defined by a domain name having an associated public network address
[38.1]	a computing device that includes at least a memory and a processor configured to implement a network manager of a virtual network that is defined by a public domain name

Claim	Limitations
[48.2]	a virtual network that is defined by a domain name having an associated public network address in a public network
[62.2]	a virtual network that is defined by a domain name having an associated public network address in a public network
[75.1, 75.2]	a computing device that includes at least a memory and a processor configured to implement a virtual network manager having a network interface coupled to a virtual network, the virtual network manager including at least one virtual community definition that is defined by a domain name having an associated public network address

218. Next, I discuss each element of the above claims in turn.

a. a network interface

219. As the '785 patent discloses: “[a]s shown in FIG. 4, A, B and X may all make direct connections to each other through communications within the virtual domain. In Figure 4, dashed lines represent direct communication paths seen to applications running on A, B and X, all “seeing the virtual addresses of each other and communicating with each other using the application’s IP interface.” EX1001 ('785 patent) at 9:36-41. Further, the '785 patent discloses “[t]he network interface can include a network card for connecting to an Ethernet or other type of LAN”. *Id.* at 9:56-58. A POSITA would understand a network interface includes a network card, such as an Ethernet or LAN card as well as an associated protocol stack.

220. Caronni I discloses communications using virtual network interfaces relating to virtual application IP addresses. Caronni I discloses physical layer transmissions:

Thus, the nodes of a Supernet may communicate over different transports, such as IP, IPX, X.25, or ATM, as well as different physical layers, such as RF communication, cellular communication, satellite links, or land-based links.

EX1003 (Caronni I) at 5:21-25.

221. Caronni I discloses protocol stacks as a component in the administrative node, where the protocol stack is “acting as the conduit for all Supernet communications.” *Id.* at 8:59.

Memory 504 also contains a VARPDB 551 and a TCP/IP protocol stack 552 that are executed by CPU 512 running in kernel mode. TCP/IP protocol stack 552 contains a TCP/UDP layer 554 and an IP layer 556, both of which are standard layers well known to those of ordinary skill in the art.

Id. at 6:62-68.

TCP/IP protocol stack 534 contains a standard TCP/UDP layer 538, two standard IP layers (an inner IP layer 540 and an outer IP layer 544), and a Supernet security layer (SNSL) 542, acting as the conduit for all Supernet communications. To conserve memory, both inner IP layer 540 and outer IP layer 544 may share the same instance of the code of an IP layer.

Id. at 8:56-62.

222. Further, “KMD 530 receives requests from SNSL 542 of the TCP/IP protocol stack 534 when a packet is received.” *Id.* at 8:51-52. Figure 5 below shows

a TCP/IP protocol stack for both a device 302 (annotated in green) and an administrative node (annotated in blue).

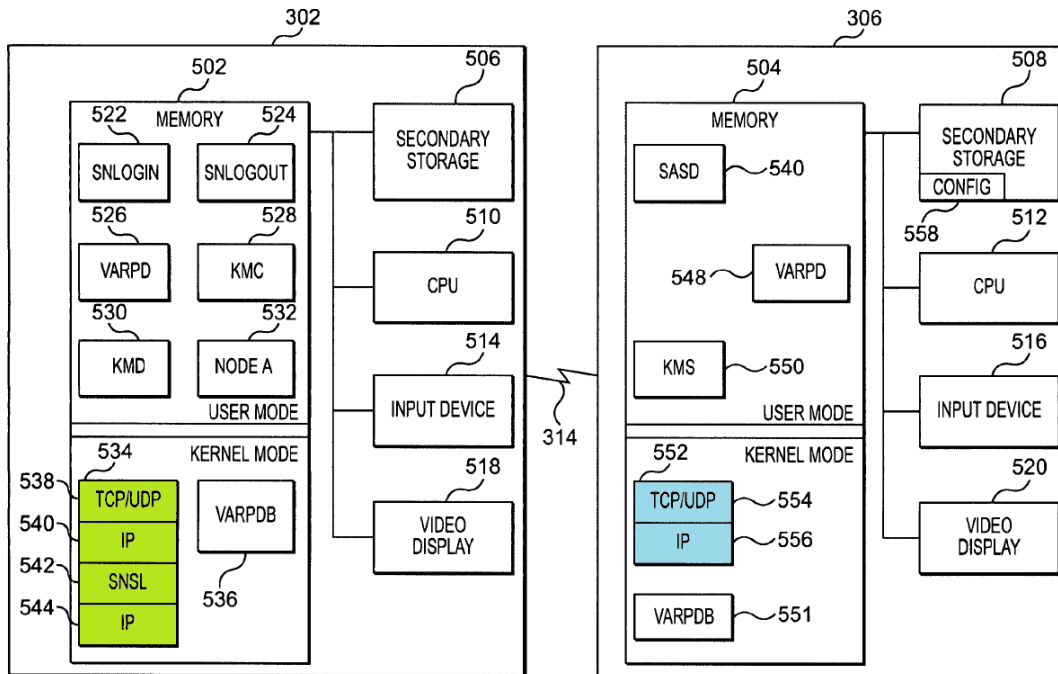


FIG. 5

Id. at Fig. 5 (annotated).

223. As Caronni I states, layers of a protocol model are “well known to those of ordinary skill in the art.” *Id.* at 6:66-67. I further note that a POSITA would understand the layers of a protocol stack and that the ISO/OSI model provides layer definitions:

The set of protocols that work together on different levels to enable communication on a network. For example, TCP/IP, the protocol stack on the Internet, incorporates more than 100 standards including FTP, IP, SMTP, TCP, and Telnet. *See also* ISO/OSI reference model.

EX1014 (Microsoft Computer Dictionary, protocol stack) at p. 387.

The ISO/OSI reference model separates computer-to-computer communications into seven protocol layers, or levels, each building—and relying—upon the standards contained in the levels below it. The lowest of the seven layers deals solely with hardware links; the highest deals with software interactions at the application-program level. It is a fundamental blueprint designed to help guide the creation of networking hardware and software.

EX1014 (Microsoft Computer Dictionary, ISO/OSI reference model) at pp. 266-267; *see also* §VII.B above.

224. In my opinion, the Caronni I protocol stacks disclose the claimed network interfaces because a POSITA would have understood that they provide (programmable) interfaces and work with actual hardware (physical or virtual) network interface cards.

225. Caronni II also discloses physical layer transmissions:

The Link layer 116 then passes the entire packet including headers to the Physical layer for transmission to the destination. The Physical Layer may use any of numerous methods to transmit the data including the use of fiber-optic cable, Ethernet cable, telephone wire, satellite relays, wireless transmissions, and other mechanisms.

EX1004 (Caronni II) at 6:66-7:5.

226. Further, Caronni II discloses network interfaces using the same protocol layering model as described in Caronni I:

The electronic device 2 includes **the five layers of the Internet protocol stack**, the application layer 4 which has a process associated with a virtual IP address, a Transport layer 8, a Network layer 10, a Link layer 12 and a Physical layer 14. The Application layer process communicates with the Transport layer through **an interface with a port address 6**.

Id. at 3:65-4:4 (emphases added).

The electronic device 30 also includes the **Internet protocol stack** with an Application layer 32, a Transport layer 36, a Network layer 38, a Link layer 40, and a Physical layer 42. The Application layer includes a process associated with the Virtual IP address of 10.0.0.1. Processes or applications on the Application layer and the Transport layer are connected by **an interface with a port address 34**.

Id. at 4:34-40 (emphases added).

227. Caronni II further discloses an interface to a local area physical network, which is a network interface:

The other layers of the **Internet protocol stack** are also present on the electronic device 52 including a Transport layer 58, a Network layer 60, a Link layer 62, and a Physical layer 64. The Application layer processes or applications and the Transport layer are connected by an interface with a port address 56. Additional electronic devices such as electronic device 70 may also be **interfaced with the local area physical network 50**.

Id. at 4:46-53 (emphases added).

228. Hipp discloses communications using virtual network interfaces relating to virtual application IP addresses. Specifically, an application may be associated with a real IP address, where the real address is associated with a host interface:

An application IP address is **an IP address associated with an application**. It can be a virtual IP address, associated solely with an application, or **it can be a real IP address**, associated with **an interface on the host** where the application resides. The application IP address is the IP address the application uses as the local address for all communications. The present invention may be utilized in both virtual and interface IP addresses.

EX1005 (Hipp) at 2:58-65 (emphases added).

229. In addition, Hipp discloses that an application is assigned an IP address that corresponds to a network interface:

Any application configuration of addresses and host names uses the virtual hostname and virtual IP address assigned to the instance. Virtualization of network identities is transparent to the application running within a VNI. From the perspective of the application, the application is running on a single node which has **an assigned IP address that corresponds to one of its network interfaces**.

Id. at 6:27-34 (emphasis added).

230. Thus, a POSITA would understand that the combination of Caronni I, Caronni II, and Hipp discloses a network interface. A POSITA would have been motivated to combine Caronni I, Caronni II, and Hipp for the reasons discussed in §XII.B below.

b. virtual network defined by a domain name having an associated public network address

231. Caronni I discloses this limitation as a virtual network or Supernet:

Methods and systems consistent with the present invention overcome the shortcomings of existing delivery schemes and addressing by establishing a virtual network on top of current IP network delivery schemes.

EX1003 (Caronni I) at 3:15-18.

A Supernet's infrastructure uses components from the Internet because devices 302, 304 and 312 contain nodes that together form a Supernet and that communicate by using the infrastructure of the Internet. These nodes 316, 318, 320, and 322 are communicative entities (e.g., processes) running within a particular device and are able to communicate among themselves as well as access the resources of the Supernet in a secure manner.

Id. at 4:66–5:7.

232. Caronni I also discloses this limitation in that a Supernet has a unique name that is associated with various public addresses, for example, a KMS address and/or a VARPd address:

To configure a Supernet, a System administrator creates a configuration file 558 that is used by SASD 540 when starting or reconfiguring a Supernet. This file may specify: (1) the Supernet name, (2) all of the channels in the Supernet, (3) the nodes that communicate over each channel, (4) the address of the KMS for each channel, (5) the address of the VARPd that acts as the server for the Supernet, (6) the user IDs of the users who are authorized to create Supernet nodes, (7) the authentication mechanism to use for each user of each channel, and (8) the encryption algorithm to use for each channel.

Id. at 8:1-11.

233. Caronni I discloses a Supernet ID, where “the Supernet IDs indicating the channels over which the process communicates.” *Id.* at 11:2-4. And further that a “Supernet ID (e.g., 0x123), reflecting a unique identifier of the Supernet.” *Id.* at 7:10-11. A POSITA would have understood that this Supernet ID is similar to a name because, just as the Supernet name uniquely identifies a Supernet, the Supernet ID defines a particular channel of the overall Supernet. In essence, a Supernet within a Supernet. Moreover, the example given of “0x123” (hexadecimal number “123”) is not formatted as a traditional IPv4 address (which uses a “dotted quad” style such as “192.168.1.2”); this further indicates to a POSITA that the Supernet ID is not an IP address and is more aptly considered a name. *See also* §VII.B.3 above.

234. Caronni I further discloses that a user may enter a Supernet name when joining a Supernet. The first step performed is that the user invokes the SNlogin script and enters the Supernet name, their user ID, their password.” EX1003 (Caronni I) at 9:67-10:2.

235. Additionally, a POSITA would understand that the SASD would have an associated public IP address, as the SASD communicates with various Supernet devices:

SASD 540 represents a Supernet: there is one instance of an SASD per Supernet, and it both authenticates nodes and authorizes nodes to join the Supernet.

Id. at 7:3-5.

[T]he SNlogin Script performs a handshaking with SASD to authenticate this information.

Id. at 10:5-7.

236. A POSITA would have understood this because every application that can communicate with other computers must rely on at least one IP address and port available for that host. *See* §§VII.AVII.B above.

237. These components are shown annotated in blue in Figure 5 below, located in the memory of administrative node 306. Here, a POSITA would understand that because the administrative node communicates over the Internet 314, public IP addresses would be used to communicate with administrative node

components. “Device 302 and administrative machine 306 communicate via Internet 314.” EX1003 (Caronni I) at 10:5-7.

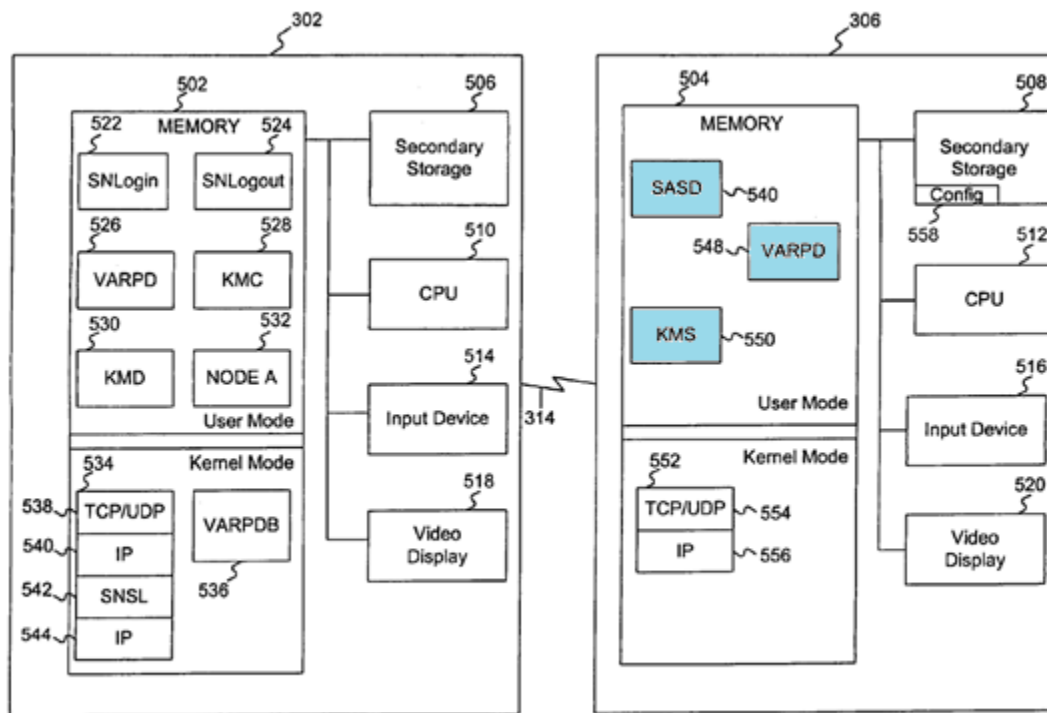


Fig. 5

Id. at Fig. 5 (annotated).

238. To the extent a domain name is a “human-readable address that identifies a specific location on the Internet, such as a website or online service,” a POSITA would recognize that the Supernet name disclosed by Caronni I is just such a domain name. EX1015 (’785 re-exam) at p. 568. For example, Caronni I discloses that a system administrator and a user both enter the Supernet name. First, a system administrator “creates a configuration file 558 that is used by SASD 540 when starting or reconfiguring a Supernet” where the “file may specify: (1) the Supernet

name.” EX1003 (Caronni I) at 8:1-5. Second a “user invokes the SNlogin script and enters the Supernet name, their user ID, their password.” *Id.* at 10:1-2.

239. Here, a POSITA would understand that a user or administrator entering a Supernet name would indicate that the Supernet name is human readable, in that the Supernet name would be easily remembered and/or transcribed. This is further exemplified by a user entering the Supernet name along with his or her username and password. The Supernet name, like a username and password, would be easily remembered and entered.

240. Further, the Supernet name identifies a specific location on the Internet. As Caronni I discloses, the administrator node (representing the Supernet) has several Internet accessible addresses as discussed immediately above. A POSITA would understand that a domain name, such as a Supernet name, would be resolvable via a DNS server to return an associated IP address.

241. In fact, DNS is a more versatile system than merely for translating hostnames to IP addresses (*i.e.*, “A” records). DNS can also translate IP addresses back to names (“PTR” records), support hostname aliases (CNAME records), mail exchange (MX) records, general-purpose (TXT) records, and even list other name server (NS) records, among others. Given this versatility, a POSITA would have understood that DNS servers can, and are most suitable, to translate a Supernet name or domain name into an IP address. *See also* §VII.B.3 above.

242. Caronni I thus discloses a virtual network defined by a domain name having an associated public network address.

243. Hipp discloses that applications and processes may constitute or be a member of a Virtual Network Environment (VNE). “The virtual network environment encapsulates a set of applications within a virtual network.” EX1005 (Hipp) at Abstract. Hipp further discloses that “[v]irtualization of network identity is achieved by assigning a unique virtual IP address and virtual hostname to a group of processes that make up the application instance which the instance keeps throughout its execution.” *Id.* at 6:1-4. A POSITA would understand that the assignment of hostnames and IP addresses to Hipp processes may be applied to the processes of Caronni I (SASD, VARPD, and KMD) to discloses this claim limitation. *See* §VII.B.4 above.

244. Thus, a POSITA would understand that the combination of Caronni I, Caronni II, and Hipp discloses a virtual network defined by a domain name having an associated public network address. A POSITA would have been motivated to combine Caronni I, Caronni II, and Hipp for the reasons discussed in §XII.B below.

3. registration/distribution of virtual network address to each device which uniquely identifies the device

Claim	Limitations
[1.2, 1.3]	the virtual network manager configured to register devices in a virtual network that is defined by a domain name,

Claim	Limitations
	each device in the virtual network being identified to the other devices by a virtual network address that is unique for each device and not directly routable via a public network, the virtual network manager further configured to distribute a virtual network address to a device when the device is registered in the virtual network;
[30.2, 30.3]	[a memory and a processor to implement] a register module configured to register devices in a virtual network, the register module further configured to: receive a registration request from an agent associated with a device; distribute a virtual network address to the device when the device is registered in the virtual network, the device being identified to other devices in the virtual network by the virtual network address
[38.2, 38.3]	the network manager configured to distribute virtual network addresses to devices that register as members in the virtual network, each device in the virtual network being identified to the other devices by a virtual network address associated with the device a first virtual network agent associated with a first device that is registered as a member in the virtual network; at least a second virtual network agent associated with at least a second device that is registered as a member in the virtual network
[48.1, 48.3]	receiving registration requests from devices that request to be registered as members of a virtual network that is defined by a domain name having an associated public network address in a public network, each of the devices having an associated private network address; distributing a virtual network address to a device to register the device as a member in the virtual network, each device in the virtual network being identified to the other devices by the virtual network address that is associated with the device

Claim	Limitations
[62.1, 62.3]	receive registration requests from devices that request to be registered as members of a virtual network that is defined by a domain name having an associated public network address in a public network, each of the devices having an associated private network address; distribute a virtual network address to a device to register the device as a member in the virtual network, each device in the virtual network being identified to the other devices by the virtual network address that is associated with the device
[75.3]	a user set of one or more devices that are registered in the virtual network, each device in the virtual network being identified to the other devices by a virtual network address that is associated with the device, the virtual network manager configured to exchange virtual network information with the one or more devices of the user set, the virtual network being accessible by devices in the user set and devices outside of the user set

245. Next, I discuss each element of the above claims in turn.

a. agent associated with a device

246. As an initial matter, a POSITA would have understood an “agent” to be one of at least “1. A program that performs a background task for a user and reports to the user when the task is done or some expected event has taken place. [] 3. In client/server applications, a process that mediates between the client and the server. 4. In Simple Network Management Protocol (SNMP), a program that monitors network traffic. *See also* SNMP.” EX1014 (Microsoft Computer Dictionary) at 19; *see also* §VII.B.1 above.

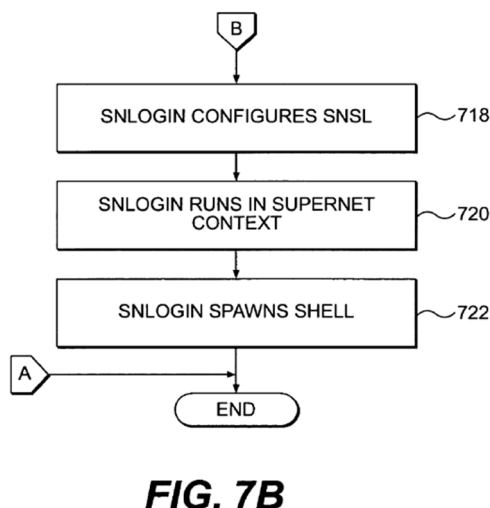
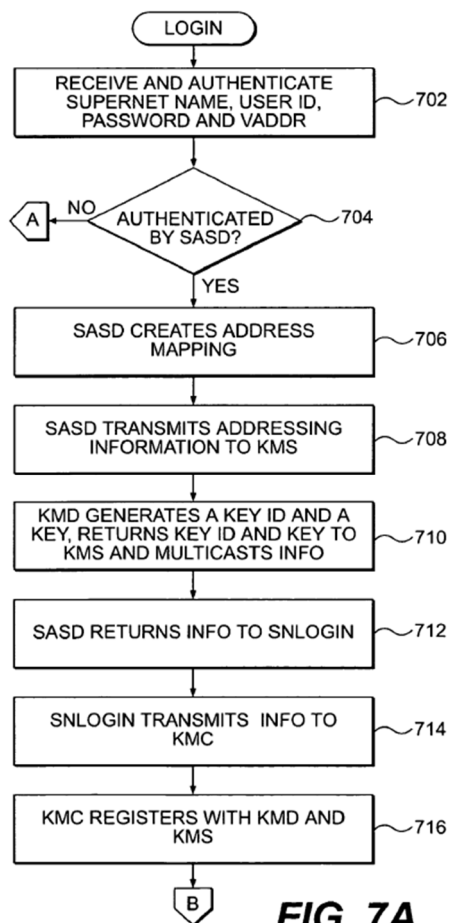
247. Caronni I discloses SNlogin and SNlogout scripts that provide access to a Supernet, where “SNlogin 522 is a Script used for logging into a Supernet” and “SNlogout 524 is a Script used for logging out of a Supernet.” EX1003 (Caronni I) at 8:31, 8:42. Further, Caronni I discloses the “SNlogin script configures SNSL” by indicating “which encryption algorithm to use for this channel and which authentication algorithm to use, both of which are received from the configuration file via SASD.” *Id.* at 10:45-49; *see* §VII.B.4 above.

248. Thus, a POSITA would understand that the Caronni I’s and Caronni II’s SNlogin and SNlogout scripts, in conjunction with the SNSL, are an agent associated with a device. A POSITA would further understand that the combination of Caronni I, Caronni II, and Hipp discloses an agent associated with a device. Also, Caronni I discloses at least a first and second agent associated with first and second devices in that “[a] Supernet’s infrastructure uses components from the Internet because devices 302, 304, and 312 contain nodes that together form a Supernet and that communicate by using the infrastructure of the Internet.” EX1003 (Caronni I) at 4:66-5:3. A POSITA would have been motivated to combine Caronni I, Caronni II, and Hipp for the reasons discussed in §XII.B below.

b. register module/registration request

249. Caronni I discloses a system for registering and distributing virtual network addresses in a Supernet that operates over a public network infrastructure.

Caronni I outlines a process for adding new nodes to the Supernet and assigning unique virtual addresses to them as shown below in Figures 7A and 7B. “FIGS. 7A and 7B depict a flow chart of the steps performed when a node joins a Supernet.” EX1003 (Caronni I) at 9:66–67.



Id. at Figs. 7A and 7B.

250. When a new node logs into the Supernet, it goes through an authentication process. “The first step performed is that the user invokes the SNlogin script and enters the Supernet name, their user ID, their password, and a requested virtual address.” *Id.* at 9:67–10:3. Then, “the SNlogin script performs a

handshaking with SASD to authenticate this information.” *Id.* at 10:5-7. Caronni I further discloses that “[i]n this step, the user may request a particular virtual address to be used, or alternatively, **the SASD may select one for them.**” *Id.* at 10:7-9 (emphasis added).

251. Then, after successful authentication, the system generates an address mapping for the new node:

Otherwise, upon successful authentication, SASD creates an address mapping between a node ID and the real address (step 706). In this step, SASD concatenates the Supernet ID with the virtual address to create the node ID, obtains the real address of the SNlogin script by querying network services in a well-known manner, and then **registers this information with the VARPD that acts as the server for this Supernet.**

Id. at 10:11-18 (emphasis added).

252. A POSITA would understand that this initial interaction of the SNlogin script and the SASD of the administrative node is the claimed registration request. Further, a POSITA would understand that the administrative node—containing a processor and memory to execute the SASD, VARPD, and KMS processes—is a claimed registration module that performs a registration process to register nodes/devices in a Supernet.

253. Caronni II discloses a registration process similar to Caronni I, including a registration request. As shown below and similar to the Caronni I VARPD address mapping, the Caronni II address-resolution facility stores a

received registration request that includes an association between a virtual address and a real IP address:

In an embodiment of the present invention a virtual network is supported by a physical network and includes a virtual address resolution facility. The virtual address resolution facility is used to resolve a virtual IP address into a real IP address. **A registration request is received at the virtual address resolution facility** which includes a real IP address, a port address, a transport protocol designation and an Application layer protocol designation associated with a virtual address destination. The associations are stored using the virtual address resolution facility.

EX1004 (Caronni II) at 3:11-20 (emphasis added).

254. Caronni II provides additional details. *See id.* at 5:10-29 (“[e]ach entry in the VARP lookup table 26 will list the registering virtual IP address 90 and associated information including a real IP address”), 7:35-39 (“a process associated with a virtual IP address registers with the virtual address resolution facility”); *see also id.* at 9:4-7 (Claim 9) (“registering a virtual address destination with said virtual address resolution facility, said registration indicating a real IP address”).

255. A POSITA would understand that the Caronni II virtual address resolution facility and registration process is the claimed register module/registration request.

256. Finally, Hipp also discloses a registration process for VNI applications. “The virtual hostname resolves to the virtual IP address for both the applications

registered with the VNI framework as well as those that are not registered.” EX1005 (Hipp) at 6:16-18.

257. Thus, a POSITA would understand that the Caronni I, Caronni II, and Hipp registration processes disclose the claimed registration limitations through the exchange of information. A POSITA would have been motivated to combine Caronni I, Caronni II, and Hipp for the reasons discussed in §XII.B below.

c. distribute a virtual address

258. Caronni I discloses that, after a device is registered, a virtual network address associated with the device is distributed to the device. As described in the registration process, the Node ID and real ID of a registered node is inserted into the administrative node’s VARPD. “SASD concatenates the Supernet ID with the virtual address to create the node ID, obtains the real address of the SNlogin script ... and **then registers this information with the VARPD that acts as the server for this Supernet.**” EX1003 (Caronni I) at 10:13-18 (emphasis added).

259. Further, Caronni I discloses that the first time a node sends a packet to another node, it must be supplied with its virtual network address from the administrative node’s VARPD:

The SNSL layer then accesses the VARPDB to obtain the address mapping between **virtual source node address** 642 and the **source real address** 614 as well as the virtual destination node address 644 and the destination real address 616 (step 808). If they are not contained in the VARPDB because **this is the first time a packet has been sent from this node** or sent to this destination, the VARPDB

accesses the local VARPd to obtain the mapping. When contacted, **the VARPd on the local machine contacts the VARPd that acts as the server for the Supernet to obtain the appropriate address mapping.**

Id. at 11:49-59 (emphases added).

260. Here, a POSITA would understand that the administrative node (SASD and/or VARPd) distributes a particular node's virtual network address the first time that particular node sends a packet. A POSITA would understand that this is an exchange of information.

261. Thus, a POSITA would understand that the combination of Caronni I, Caronni II, and Hipp discloses the distribution of a virtual address. A POSITA would have been motivated to combine Caronni I, Caronni II, and Hipp for the reasons discussed in §XII.B below.

d. identified by a unique virtual network address

262. As an initial matter, I note that most of the '785 patent's claims do not call for the network address to be of any specific form (*e.g.*, IPv4 or IPv6). "Virtual IP addresses are used to identify members in the community in the virtual packet and are not directly routable. Rather, they are unique identifiers of a given connection to the upper layer application. In one embodiment, virtual IP addresses are in IPV4 format." EX1001 ('785 patent) at 12:12-17. A POSITA would have understood this to mean that IP addresses can be formatted as 32-bit IPv4, but do not have to be (as long as they are unique).

263. “IPsec can (optionally) be applied to the Virtual IP Packet for end-to-end security.” *Id.* at 12:11-12. A POSITA would have known that IPsec encapsulates one address inside the other. But what is encapsulated can include an IPv4 address and more useful information as needed. It is only when the packets are decapsulated that the system has to pick the “inner” address and forward it along to its destination. *See* §VII.B.4 above. Moreover, a POSITA would have further understood that if one combined a unique IPv4 address with some other information, then the end result would still be unique. *Id.*

264. Further, a POSITA would understand Caronni I’s node ID, being a concatenation of a Supernet ID and a virtual network address, is itself a virtual network address. A POSITA would further understand both the virtual network address and the node ID are unique and used to identify a node/device to other nodes/devices in the virtual network.

265. The node ID serves as a unique identifier within the Supernet and consists of two parts. “The ‘node ID’ may include the following: a Supernet ID (e.g., 0x123), reflecting a unique identifier of the Supernet, and a virtual address, comprising an IP address (e.g., 10.0.0.1).” EX1003 (Caronni I) at 7:10-13.

266. Further, a POSITA would understand that the node virtual network addresses are unique “[s]ince the node ID includes a Supernet ID, a node will have more than one node ID when it communicates over more than one channel.” *Id.* at

7:16-18. Here, virtual network addresses are unique because within a particular channel or Supernet ID, individual nodes must be distinguishable—by their virtual address—from other nodes on that channel.

267. Hipp further discloses unique virtual addresses. EX1005 (Hipp) at 6:1-4 (“Virtualization of network identity is achieved by assigning a unique virtual IP address and virtual hostname to a group of processes that make up the application instance which the instance keeps throughout its execution.”).

268. Next, Caronni I discloses a node is identified to other nodes in a virtual network, where, as described below, a virtual packet is created by a source node using virtual network addresses for the source node and the destination node:

FIG. 8 depicts a flow chart of the steps performed when sending a packet, such as packet 600 depicted in FIG. 6, from node A. ... The first step performed is for inner IP layer 540 to receive a packet originating from node A via the TCP/UDP layer 538 (step 802). The packet contains virtual source node address 642, virtual destination node address 644, and data 654. The packet may be received from a process executing in node A connected to a socket.

EX1003 (Caronni I) at 11:18-32.

269. Here, A POSITA would understand that because node A uses a node’s virtual network address as a destination address, then that node is identified to other nodes—in this case node A—by its virtual network address.

270. Further, when transmitting data, the system uses modified socket structures containing both virtual and real addresses. *Id.* at 11:38–40 (“The socket

structure is modified so as to contain an extra data field for Supernet ID 626 and virtual source address 642.”). This allows the Supernet Socket Layer (SNSL) to “access[] the VARPDB to obtain the address mapping between virtual source node address 642 and the source real address 614 as well as the virtual destination node address 644 and the destination real address 616,” enabling secure cross-network communication without exposing physical infrastructure details. *Id.* at 11:49-53.

271. Thus, a POSITA would understand that the combination of Caronni I, Caronni II, and Hipp discloses identification by a unique virtual network address. A POSITA would have been motivated to combine Caronni I, Caronni II, and Hipp for the reasons discussed in §XII.B below.

e. not directly routable via a public network

272. Caronni I describes a virtual network address as “a virtual address, comprising an IP address (e.g., 10.0.0.1).” *Id.* at 7:12-13. Any POSITA would understand that the IP address of 10.0.0.1 is a reserved or private IP address and is not routable over a public network such as the Internet. For example:

[T]hree ranges of IP addresses have been declared as private. Companies may use them internally as they wish. The only rule is that no packets containing these addresses may appear on the Internet itself. The three reserved ranges are:

10.0.0.0	– 10.255.255.255/8	(16,777,216 hosts)
172.16.0.0	– 172.31.255.255/12	(1,048,576 hosts)
192.168.0.0	– 192.168.255.255/16	(65,536 hosts)

EX1028 (Tanenbaum2) at pp. 444-446.

273. Caronni I thus discloses a virtual network address not directly routable via a public network.

274. Caronni I also discloses that a Supernet is a private network with an addressing scheme separate from the underlying public network, such as the Internet:

Since the Supernet is a private network constructed from the infrastructure of another network, the Supernet has its own internal addressing scheme, **separate from the addressing scheme of the underlying public network**. Thus, when a packet from a Supernet node is sent to another Supernet node, it travels through the public network. To do so, the Supernet performs address translation from the internal addressing scheme to the public addressing scheme and vice versa. **By separating the addressing schemes**, the Supernet creates a flexible delivery scheme that is easily changeable by network software or a system administrator.

EX1003 (Caronni I) at 6:7-18 (emphases added).

275. A POSITA would understand that the virtual network addresses of the private Supernet addressing scheme that is separate from the underlying public network would not be routable over the underlying public network.

276. Also, Caronni I describes a virtual network address as “a virtual address comprising an IP address (e.g., 10.0.0.1)” and a real address comprising an “IP address (e.g., 10.0.0.2).” *Id.* at 7:12-13, 7:19-21. A POSITA would have recognized that these IP address, 10.0.0.1 and 10.0.0.2, are reserved or private IP addresses associated with the Supernet’s nodes/devices, and are not routable over a public network such as the Internet.

277. Further, Hipp discloses virtual addresses in the context of a virtual network environment (VNE) that include reserved or private IP addresses:

The VNE is defined by subnet of addresses contained within the VNE. For example, all applications within the subnet 10.10.2.0 comprise a VNE. The subnet/netmask specifying such a VNE would be 10.10.2.0/255.255.255.0 and would include the addresses 10.10.2.0 through 10.10.2.255. In this example, an application with IP address 10.10.2.2 would be able to communicate with an application at address 10.10.2.60, but not at 10.10.0.1.

EX1005 (Hipp) at 3:19-27; *see also id.* at Figs. 1, 3-7.

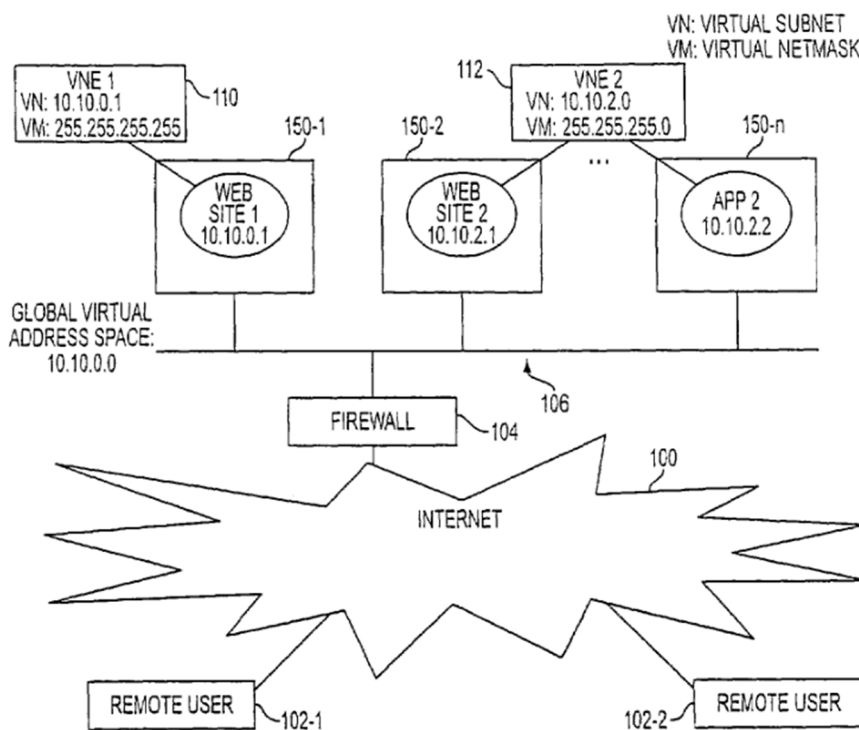


FIG. 1

Id. at Fig. 1 (showing virtual network addresses such as 10.10.0.x and 10.10.2.x).

278. Thus, a POSITA would understand that the combination of Caronni I, Caronni II, and Hipp discloses not being directly routable via a public network. A

POSITA would have been motivated to combine Caronni I, Caronni II, and Hipp for the reasons discussed in §XII.B below.

f. associated private network address

279. Caronni I also discloses that Supernet nodes/devices have an associated private network address:

VARPD 548 has an associated component, VARPD 551, into which it stores mappings of the internal Supernet addresses, known as a node IDS, to the network addresses recognized by the public-network infrastructure, **known as the real addresses**.

EX1003 (Caronni I) at 7:5-9 (emphasis added).

280. Caronni I provides examples of those real addresses:

The “real address” is an IP address (e.g., 10.0.0.2) that is globally unique and meaningful to the public-network infrastructure.

Id. at 7:18-21.

281. Caronni II discloses devices having an associated private address, called a “real” IP address:

Each entry in the VARP lookup table 26 will list the registering virtual IP address 90 and associated information including **a real IP address 91**, a transport protocol designation 92, a port number identification 93 and an Application layer protocol designation 94.

EX1004 (Caronni II) at 5:18-22 (emphasis added).

282. Further, Caronni II describes that a message is sent using a real IP address destination, where the destination may be behind an edge device (firewall, NAT box, proxy server, packet filtering device, and gateway):

[S]ending said formatted message, augmented by said determined MAC address destination, said **real IP address destination** and said transport protocol header, from a virtual IP address **to the real IP address destination** indicated in said virtual address destination registration.

Id. at 9:20-24 (emphases added).

The method of claim 1 wherein said virtual address destination is behind one of a **firewall, NAT box, proxy server, packet filtering device, and gateway.**

Id. at 9:31-32 (emphasis added).

283. Caronni II further explains inventive aspects, for example to allow messages to reach their intended destination when destinations are located in the interior of a network and have private addresses:

Unfortunately, when the virtual destination address is located on a physical device on the interior of a network which is running a proxy server, firewall, or other packet filtering mechanism, messages that have been sent to a virtual destination address have difficulty getting all the way to their target. **The term “interior of a network” refers to devices which are not able to directly access another network without first going through another device on their own network.** For example, most local area networks (LANs) access the Internet through a proxy server. **Devices other than the proxy server are said to be on the interior of the LAN.** The proxy server is referred to as an “edge device” because it is able to directly contact another network without using an intermediary device.

Id. at 1:60-2:6 (emphases added).

284. A POSITA would recognize that devices behind an edge device would be associated with the claimed private network address. Indeed, Caronni II discloses a LAN may be the only infrastructure network supporting a virtual network:

Those skilled in the art will recognize that a number of different types of networks may be utilized within the scope of the present invention. The networks may be the Internet, an intranet, wide area network (WAN), a local area network (LAN), a satellite network, a wireless network, or some other type of network capable of supporting a virtual network through the devices thereon.

Id. at 4:27-33.

285. Caronni II thus discloses this limitation.

286. Further, Hipp also discloses devices having an associated private network address. As shown below in Figure 1 (annotated with red arrows), computers 150-1, 150-2, ... 150-n are located on private network 106:

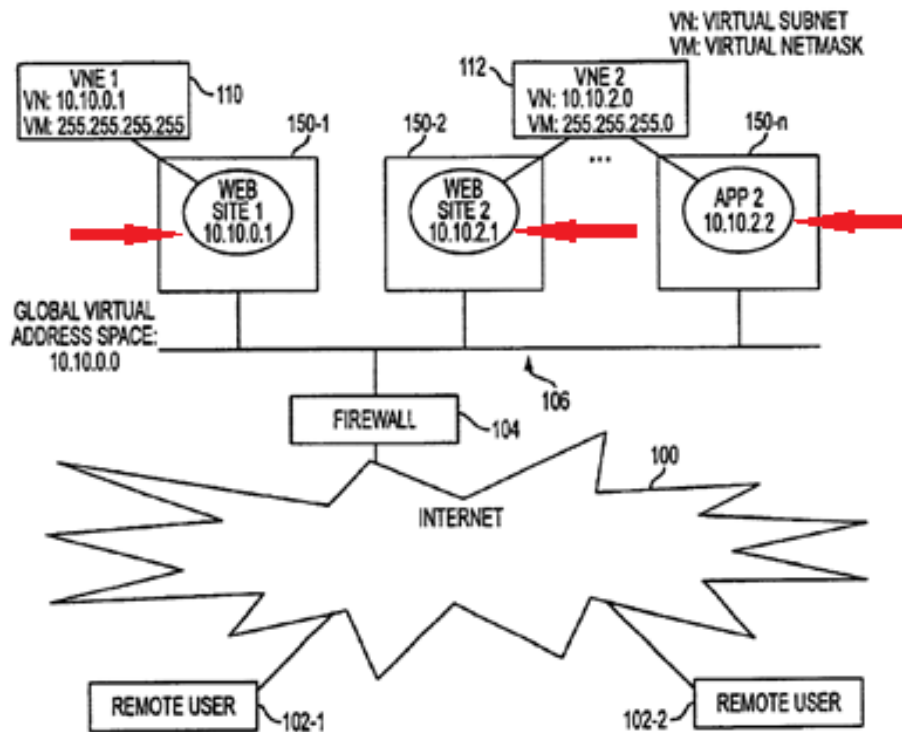


FIG. 1

EX1005 (Hipp) at Fig. 1 (annotated).

287. Hipp describes Figure 1 as having a private network address as follows:

FIG. 1 illustrates in high level block diagram form the overall Structure of the present invention as used in connection with a global computer network 100 such as the Internet.

Remote users 102-1 and 102-2 can connect through the computer network 100 to a private network of computers 106 protected by firewall 104. Computer network 106 is a network comprising computers 150-1, 150-2, through 150-n, where n is the total number of computers in network 106.

Id. at 2:43-52.

288. A POSITA would understand that devices, like computers 150, on a private network like network 106, would have private network addresses.

289. Thus, a POSITA would understand that the combination of Caronni I, Caronni II, and Hipp discloses an associated private network address. A POSITA would have been motivated to combine Caronni I, Caronni II, and Hipp for the reasons discussed in §XII.B below.

g. the virtual network being accessible by devices in the user set and devices outside of the user set

290. Caronni I discloses this limitation as shown below in Figure 4. Here, Node C is shared between Channel 1 and Channel 2, where a “‘channel’ refers to a collection of virtual links through the public-network infrastructure that connect the nodes on the channel such that only these nodes can communicate over it.” EX1003 (Caronni I) at 5:27-30. Further, “[e]ach Supernet has any number of preconfigured channels over which the nodes on that channel can communicate.” *Id.* at 5:36-38.

Thus, a POSITA would understand that channel 1 is one virtual network and channel 2 is a separate virtual network, where the nodes on channel 1 cannot communicate with the nodes on Channel 2.

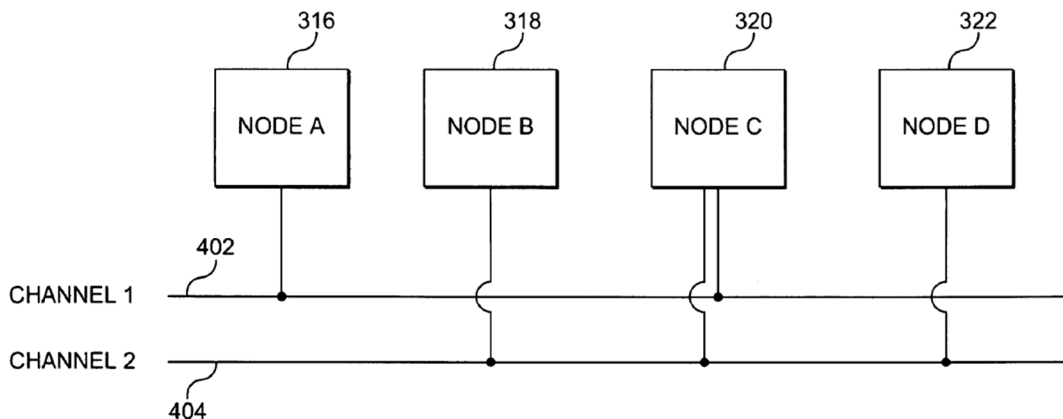


FIG. 4

Id. at Fig. 4.

291. Caronni further discloses that:

[N]ode C 320 serves as a file system manager by receiving file system requests (e.g., open, close, read, write, etc.) and by satisfying the requests by **manipulating a portion of the secondary storage** on its local machine. To maintain security, node C 320 **stores the data in an encrypted form so that it is unreadable by others.**

Id. at 5:44-49 (emphases added).

292. Here, a POSITA would understand that the portion of the node C secondary storage is accessible to the user set of channel 1 (nodes A and C) would also be accessible to the user set of channel 2 (nodes B, C and D). And further that the user set of nodes B and D are outside the user set of node A because, due to the encryption, the data is “unreadable by others.” *See* §VII.B.4 above.

293. Hipp discloses virtual network environments being accessible by devices in the user set (a collection of one or more computer applications in Hipp) and devices outside of the user set (remote applications and remote clients):

The purpose of the VNE is to isolate applications running in the shared resource environment. Therefore, although checks are made to ensure there is no communication between VNES, **communication with remote applications is still allowed**. For illustrative purposes, consider a first VNE containing a web server for company X and a second VNE containing a web server for company Z. The VNE framework 200 keeps the two VNES separate, but **both VNES can communicate with a remote client**.

EX1005 (Hipp) at 4:11-19 (emphases added).

294. Here, the virtual network of the first and second VNES contain applications (members of a user sets) that are available to communicate with members outside the user set (remote clients) as depicted in Figure 1.

295. Hipp discloses, as shown in Figure 1 below, a virtual network environment (VNE 2) contains a user set that includes computer 150-2 and 150-n but does not include remote users 102-1 or 102-2. As annotated below, blue indicates a user set of devices/members (VNE 2) and green indicates a different user set of devices/members (VNE 1). Further, orange indicates devices/member outside of the blue and green user sets.

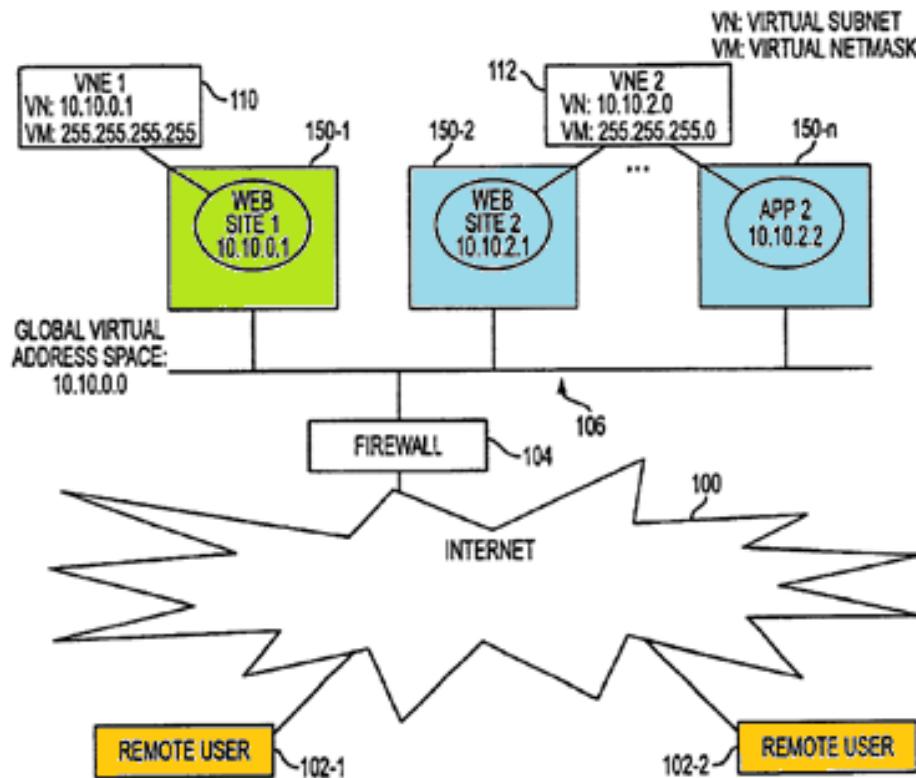


FIG. 1

Id. at Fig. 1 (annotated).

296. Here, computer network 106, located behind a firewall 104, is a private network of computers 150-1, 150-2, etc., where the computers “run various applications, as well as host web sites for access by remote users 102.” *Id.* at 2:60-61.

297. Further, as discussed above and shown in Hipp’s annotated Figure 1, VNE 1 is a user set and VNE 2 is a different user set. Hipp discloses that, although remote users are not part of the user set of either VNE 1 or VNE 2 remote users “can

connect through the computer network 100 to a private network of computers 106 protected by firewall 104.” *Id.* at 2:48-50. Also, remote users may access the VNE servers:

Computers 150 are used to run various applications, as well as host web sites for access by remote users 102. The present invention is implemented on computer network 106 in the form of virtual network environment (VNE) 110 and VNE 112.

Id. at 2:53-57.

298. Further, Hipp discloses that communication between applications in different VNEs may take place:

An application running within one VNE can communicate with another application in the same VNE. However, an application in one VNE can not communicate with an application in another VNE **unless expressly permitted**.

Id. at 3:5-9 (emphasis added).

299. Thus, a POSITA would understand that the combination of Caronni I, Caronni II, and Hipp discloses a virtual network being accessible by devices in the user set and devices outside of the user set. A POSITA would have been motivated to combine Caronni I, Caronni II, and Hipp for the reasons discussed in §XII.B below.

4. route director/routing encapsulated packets

Claim	Limitations
[1.4]	a route director implemented with a second device memory and a second device processor of a second computing device, the route director configured to communicate data between the devices that are registered in the virtual network, the data

Claim	Limitations
	being communicated as encapsulated packets from a source device to a destination device, an encapsulated packet including a first virtual network address that corresponds to the source device and a second virtual network address that corresponds to the destination device
[38.4]	a route director configured to route communications between the first device and the at least second device in the virtual network via the respective first and second virtual network agents, the communications configured for routing as encapsulated packets that include a first virtual network address that is not directly routable corresponding to the first device and a second virtual network address that is not directly routable corresponding to the at least second device;
[48.4]	routing communications between the devices that are registered in the virtual network, the communications being routed as encapsulated packets from a source device to a destination device, an encapsulated packet including a first virtual network address that corresponds to the source device and a second virtual network address that corresponds to the destination device
[62.4]	manage communications routed between the devices that are registered in the virtual network, the communications routed as encapsulated packets from a source device to a destination device, an encapsulated packet including a first virtual network address that corresponds to the source device and a second virtual network address that corresponds to the destination device

300. Next, I discuss each element of the above claims in turn.

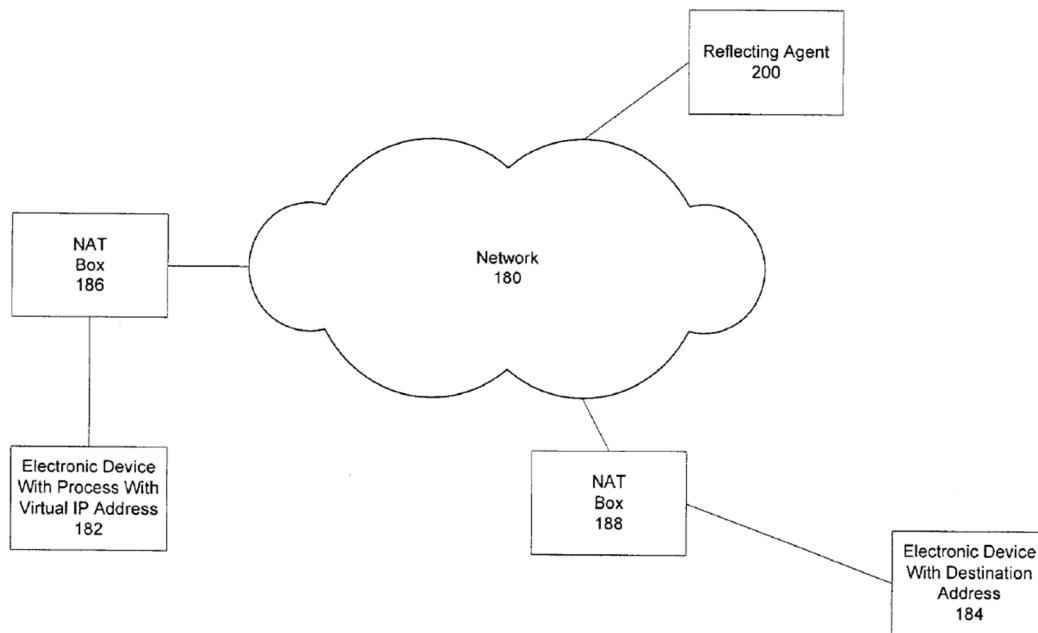
a. route director

301. Caronni II discloses the claimed route director as a reflecting agent as shown below in Figure 5:

Fig. 5 is a block diagram of an environment suitable for practicing the illustrative embodiment of the present invention which includes the use of a Network Address Translation (NAT) box and a reflecting agent.

EX1004 (Caronni II) at 3:42-45.

Figure 5



Id. at Fig. 5.

302. Caronni II discloses the reflecting agent is an intermediary that allows messages to be sent to and from devices behind a NAT box:

When the destination address is also behind a NAT box, the originator of the message is unable to directly address the destination (since the address of the destination may not be routable in the public Internet). In such a case, the present invention adds an entry to the VARP table for a third-party reflecting agent located at an address outside the NAT box. **Messages that are being sent from the originating virtual address behind a NAT box to a destination which is behind a different NAT box are sent via the reflecting agent intermediary outside the NAT box and reflected to the destination.**

Id. at 8:14-25 (emphasis added).

FIG. 5 depicts a block diagram of an environment holding a virtual network in which both the originator and destination of a message are located behind NAT boxes. An electronic device holding an originating process 182 which is part of a virtual network is interfaced with a network 180, such as the Internet, via a NAT box 186. An intended recipient for a message sent from the originating process 180 is an electronic device with a destination address 184 that is also behind a NAT box 188. **A reflecting agent 200, the address of which has been added to a VARP table, acts as an intermediary to allow a connection between the originating process 182 and destination address 184.**

Id. at 8:26-39 (emphasis added).

303. A POSITA would thus understand that adding the reflecting agent's address to the VARP table enables messages to be routed through this intermediary to reach their intended destination, even for devices behind a NAT box.

304. Thus, a POSITA would understand that the combination of Caronni I, Caronni II, and Hipp discloses a route director. Further, a POSITA would understand that the electronic devices disclosed in Caronni II, such as the reflecting agent, would be implemented with a memory and processor. *See* §VII.A above. A POSITA would have been motivated to combine Caronni I, Caronni II, and Hipp for the reasons discussed in §XII.B below.

b. encapsulated packets

305. A POSITA would have known that packet encapsulation was well understood and used by 2003. Virtual private networks (VPNs) existed, which utilize packet encapsulation (*e.g.*, embedding one packet inside another). IP Security

(IPSec) protocols existed to allow one to transmit secure, encrypted packets, embedded inside plain IP packets. *See* §VII.B above.

306. Caronni I discloses the claimed encapsulated packets that include a first virtual network address that corresponds to the source device and a second virtual network address that corresponds to the destination device. Generally, Caronni I discloses tunneling, where tunneling “refers to encapsulating one packet inside another when packets are transferred between two end points.” EX1003 (Caronni I) at 2:29-31.

307. Figure 6 of Caronni I, shown below, further shows packet encapsulation with virtual source and destination addresses, where a Supernet IP packet 600 is shown. For example, the Supernet IP packet 600 includes a section 660 (shown below annotated in blue) that Caronni I calls the addressing section, where “section 660 corresponds to inner IP layer.” *Id.* at 9:6-8.

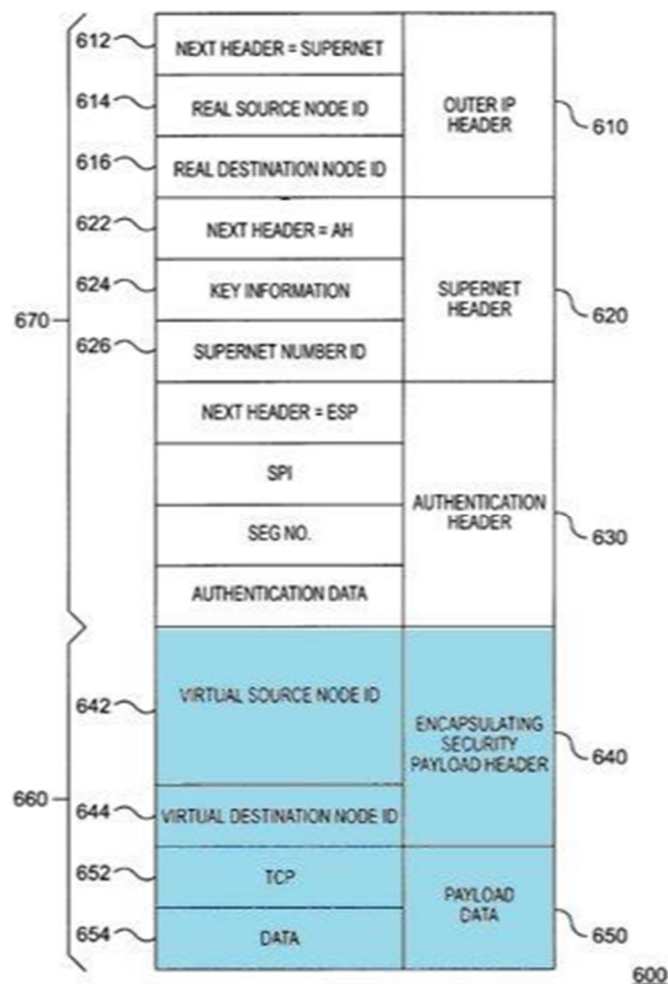


FIG. 6

Id. at Fig. 6 (annotated).

308. Caronni I further explains this inner layer receives a packet originating from a source node and includes a virtual source node address and a destination source node address, shown as 642 and 644 in the addressing section of Figure 6:

The first step performed is for inner IP layer 540 to receive a packet originating from node A via the TCP/UDP layer 538 (step 802). The packet contains virtual source node address 642, virtual destination node address 644, and data 654.

Id. at 11:26-30.

309. A POSITA would understand that these addresses are the claimed first virtual network address that corresponds to the source device and a second virtual network address that corresponds to the destination device, respectively.

310. Caronni I then further explains that “[t]he packet and Supernet ID are then transmitted to the SNSL layer.” *Id.* at 11:47-48. Referring to Figure 6, “[d]elivery scheme section 670 corresponds to the SNSL layer 542 and is meaningful only to the public-network infrastructure.” *Id.* at 9:8-10. Here, the delivery scheme section 670 includes real source and destination node addresses for routing the packet 600 over from destination node to source node. “Source real address 614 contains the real address of the originating node of Supernet packet 600. Destination real address 616 contains the real address of the destination node of Supernet packet 600.” *Id.* at 9:17-20.

311. Caronni I further explains the encapsulation process as follows:

ESP 640 contains virtual source node address 642 and virtual destination node address 644. Virtual addresses 642 and 644 contain the addresses known only to the members of the channel. For example, referring to FIG. 4, if packet 600 were sent from Node A316 to Node C 320, virtual source node address 642 would correspond to Node A316, and virtual destination node address 644 would correspond to Node C 320. Finally, payload data 650 contains the original IP packet sent from the source to the destination.

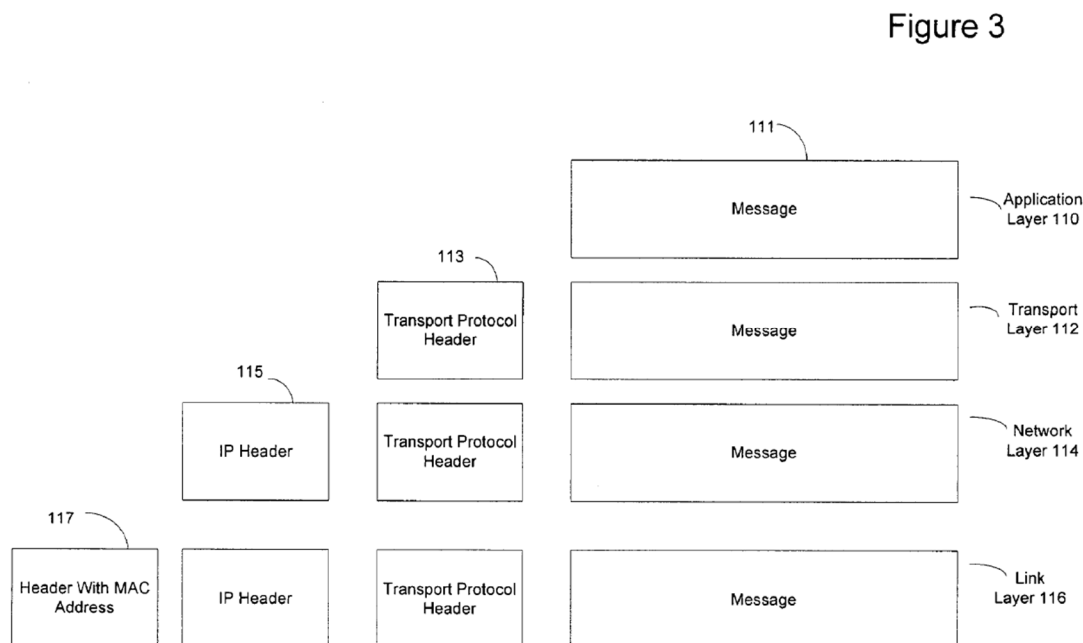
Id. at 9:28-37.

312. A POSITA would understand that the packet originating from a source node and including a virtual source node address and a destination source node address is thus encapsulated into the Supernet packet 600. *See* §VII.B.4 above.

313. Caronni II also discloses the encapsulation claim limitations. Figure 3 below shows “a block diagram of the encapsulation process used to send a message from a virtual IP address to another virtual IP address.” EX1004 (Caronni II) at 6:35-38; *see* §VII.B.4 above. For example:

FIG. 3 is a block diagram of the encapsulation of a message intended for a virtual destination address utilizing the registered higher level protocols specified in a Virtual Address Resolution Protocol lookup table.

EX1004 (Caronni II) at 3:32-35.



Id. at Fig. 3.

314. As shown in Figure 3 above, an initial message 111 is passed from the application layer to the transport layer. A POSITA would understand that this application layer message would contain a source and destination virtual IP address as the encapsulation process includes application layer. “Those skilled in the art will realize that the message may include an IPSEC header, inner IP header, UDP or TCP header and inner payload.” *Id.* at 6:51-53. Further, “[m]essages intended for a virtual destination address ... are encapsulated in higher level protocols prior to being sent to the edge device” where the “higher layers of the edge device, such as the Transport layer and the Application layer, may be accessible.” *Id.*, Abstract; *see* §VII.B.4 above.

315. Thus, a POSITA would understand that the combination of Caronni I, Caronni II, and Hipp discloses encapsulated packets. A POSITA would have been motivated to combine Caronni I, Caronni II, and Hipp for the reasons discussed in §XII.B below.

5. DNS server, request, and responses

Claim	Limitations
[1.5]	the virtual network manager further configured to receive a DNS request from the source device, and return a public network address of the route director, a private network address for the destination device, and the second virtual network address that corresponds to the destination device
[30.4]	a DNS server for the virtual network, the DNS server configured to receive a DNS request from a first device in the virtual network, and return a network address associated

Claim	Limitations
	with a network route director, a private network address associated with a second device in the virtual network, and a virtual network address associated with the second device
[38.5]	and the network manager includes a DNS server configured to provide authoritative responses for DNS queries in the virtual network, the DNS server further configured to receive a DNS query from the first device and return a network address of the route director, a network address of the second device, and the virtual network address of the second device
[48.5]	transmitting a response to a DNS request received from one of the devices that are the members in the virtual network, the response to the DNS request including a public network address of a route director that registers the devices, a public network address of the destination device, and the second virtual network address that corresponds to the destination device
[62.5]	transmit a response to a DNS request received from one of the devices that are the members in the virtual network, the response to the DNS request including a public network address of the virtual network manager, a public network address of the destination device, and the second virtual network address that corresponds to the destination device
[75.4]	the virtual network manager further configured to receive a DNS request from a source device, and return a public network address of a route director, a private network address for a destination device, and a virtual network address that corresponds to the destination device

316. Next, I discuss each element of the above claims in turn.

a. DNS server for the virtual network.

317. The '785 patent generally describes that the function of a DNS server is “to convert a domain name to an IP address.” EX1001 ('785 patent) at 1:48-50.

Also, “[i]f an application wishes to make a connection to ‘m3.abcdef.com’ and another to ‘ma.abcdef.com’, it resolves both DNS identifiers into IPv4 addresses.”

Id. at 12:32-36. The ’785 patent describes this function in terms of a DNS request or query and a DNS response:

The DNS query and response sequence is illustrated in FIG. 15. At step 1510, a member will request a DNS lookup by sending a DNS request 1520 to port 53 of the VCN Manager. The DNS request 1520 will result in a normal DNS response 1575 unless any of a number of failure conditions are met.

Id. at 22:13-18.

318. I provide additional technical details of DNS operations in §VII.B.3 above.

319. Caronni I contemplates the use of DNS as it describes that “SASD concatenates the Supernet ID with the virtual address to create the node ID, obtains the real address of the SNlogin script by querying network services in a well-known manner.” EX1003 (Caronni I), 10:13-16. Here, a POSITA would understand that the “querying network services in a well-known manner” is a DNS operation. *See also* §VII.B.3 above.

320. Further, Caronni I discloses that member virtual addresses are known to the member nodes of a particular channel or Supernet ID. “Virtual addresses 642 and 644 contain the addresses known only to the members of the channel.” EX1003 (Caronni I) at 9:30-31. Caronni exemplifies this “beforehand” knowledge of a

member node virtual address in its description of creating a packet for transmission from one member to another:

FIG. 8 depicts a flow chart of the steps performed when sending a packet, such as packet 600 depicted in FIG. 6, from node A. ... The first step performed is for inner IP layer 540 to receive a packet originating from node A via the TCP/UDP layer 538 (step 802). **The packet contains** virtual source node address 642, **virtual destination node address** 644, and data 654. The packet may be received from a **process executing in node A** connected to a socket.

Id. at 11:18-32 (emphases added).

321. However, Caronni I does not detail how this virtual destination node address is known, only that the packet containing the address was from a process executing in the sending node. A POSITA would understand that higher layer processes in a Caronni I sending node could use DNS services, such as those disclosed by Hipp or those generally known to a POSITA, to determine a virtual destination node address. The need for this functionality becomes apparent to a POSITA when considering the Supernet system disclosed by Caronni I in Figure 11.

322. As shown below in Figure 11, a Supernet system provides access to web servers for web clients, where “web clients 1102 communicate with web server 1104a by sending encrypted packets to the web server as described in FIG. 9.” *Id.* at 13:8-10. Here again, Caronni does not disclose how a virtual destination address is determined by a sending node. Nevertheless, a POSITA would understand that the web server/client configuration presents a classic and widely used application

for a DNS operation, where a client uses a DNS request to determine a web server IP (virtual) destination address. *See* VII.B.4 above.

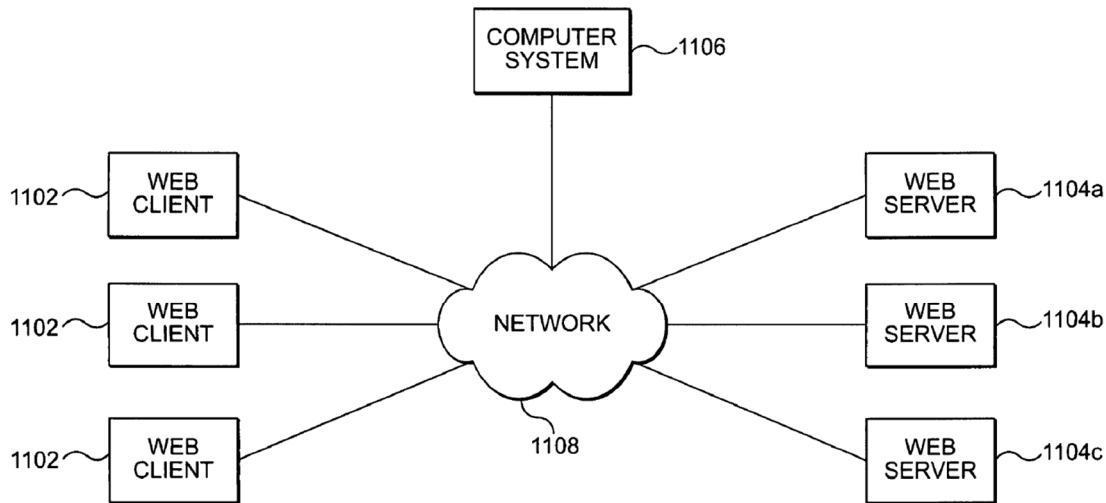


FIG. 11

EX1003 (Caronni I) at Fig. 11.

323. Furthermore, Hipp discloses a DNS server for use in a virtual network environment (VNE), where a unique virtual hostname is resolved or translated into a unique virtual IP address. “Virtualization of network identity is achieved by assigning a unique virtual IP address and virtual hostname to a group of processes that make up the application instance.” EX1005 (Hipp) at 6:1-4. Further, “[t]he virtual hostname resolves to the virtual IP address for both the applications registered with the VNI framework as well as those that are not registered.” *Id.* at 6:16-18.

324. Hipp indicates that this virtual network DNS server is a standard DNS mechanism that has been preconfigured to resolve virtual hostnames to virtual IP addresses:

For example, if an application instance used a virtual IP address of 10.10.0.1 and a virtual hostname of host1055, **the standard hostname to IP address resolution mechanisms** (e.g. DNS or the /etc/hosts file) would have to be preconfigured **to resolve a query of host1055 to IP address 10.10.0.1.**

Id. at 6:20-25 (emphases added).

325. Thus, a POSITA would understand that the combination of Caronni I, Caronni II, and Hipp discloses a DNS server for the virtual network. A POSITA would have been motivated to combine Caronni I, Caronni II, and Hipp for the reasons discussed in §XII.B below.

b. DNS request

326. Caronni I discloses DNS type requests through the operation of the VARP address translation or resolution process:

SNSL 542 utilizes VARPDB 536 to perform address translation. VARPDB stores all of the address mappings encountered thus far by SNSL542. If SNSL 542 requests a mapping that VARPDB 536 does not have, VARPDB communicates with the VARPD 526 on the local machine to obtain the mapping. VARPD 526 will then contact the VARPD that acts as the server for this particular Supernet to obtain it.

EX1003 (Caronni I) at 9:47-54.

[A] VARPD may act as a server by storing all address mappings for a particular Supernet into its associated VARPDB. Second, regardless of its role as a server or not, each VARPD assists in **address translation**

for the nodes on its machine. In this role, the VARPD stores into its associated VARPDB the address mappings for its nodes.

Id. at 7:22-28 (emphasis added).

327. Next, Caronni II also discloses DNS type requests through the operation of the VARP address translation process:

A **resolution request is received** referencing a virtual address destination and the resolution request is resolved using the virtual address resolution facility and the stored associations.

EX1004 (Caronni II) at 3:20-23 (emphasis added).

FIG. 2 depicts a block diagram of the VARP lookup table 26 used by the virtual address resolution facility 24. The **VARP lookup table 26** may be stored at any location accessible over the network. The VARP lookup table 26 includes registered virtual destination addresses located in a column 78 and corresponding associations located in a column 79. Entries 80, 82, 84, 86 and 88 include the registered virtual destination address and information associated with the registered virtual destination address. Each entry in the **VARP lookup table 26** will list the registering virtual IP address 90 and associated information including a real IP address 91, a transport protocol designation 92, a port number identification 93 and an Application layer protocol designation 94.

Id. at 5:10-22 (emphases added).

328. A POSITA would understand both the Caronni I (address translation) and Caronni II (VARP lookup table) are similar to DNS operations, in that they use a form of resolving hostnames to IP addresses. *See* §VII.B.3 above.

329. Hipp discloses DNS requests or queries:

The virtual hostname resolves to the virtual IP address for both the applications registered with the VNI framework as well as those that are not registered. This may require configuration of a name service or OS host configuration files. For example, if an application instance

used a virtual IP address of 10.10.0.1 and a virtual hostname of host1055, the standard hostname to IP address resolution mechanisms (e.g. DNS or the /etc/hosts file) would have to be preconfigured to **resolve a query of host1055 to IP address 10.10.0.1.**

EX1005 (Hipp) at 6:16-26 (emphasis added).

330. Thus, a POSITA would understand that the combination of Caronni I, Caronni II, and Hipp discloses a DNS request. A POSITA would have been motivated to combine Caronni I, Caronni II, and Hipp for the reasons discussed in §XII.B below.

c. DNS response

331. The '785 patent describes a modification to a standard DNS response to include additional information beyond the standard resolved virtual IP address. "The response includes the following information: ... the target member virtual IP address; ... the target member private IP address; ... the target Route Director public IP address; the source Route Director public IP address." EX1001 ('785 patent) at 22:38-50. Furthermore,

The DNS operation is conducted using the DNS protocol with the VCN Manager. The member software sends a DNS query to the VCN Manager, and the manager returns information about the target member. Besides the target member's virtual IP address, the DNS response also contains additional information about the target member.

Id. at 22:1-6.

The target member's virtual IP address will be presented using an A-type resource record in the answer section of the normal DNS response packet. All the remaining information will be transferred using a TXT-type resource record in an additional section of the DNS Response.

Id. at 22:13-18.

In order to route packets to a peer in a different addressing realm, the protocol stack knows the address of the appropriate Route Director that serves the peers realm, the private address of the peer or a NAT address. The Member Agent obtains the information by using the VCN Manager as a DNS authority for DNS lookup operations as described below (FIGS. 13 and 21). The DNS response from the VCN Manager will include the above-identified information in an optional text field of the response to the Member Agent.

Id. at 12:23-31.

332. A POSITA would understand this additional information disclosed by the '785 patent is well known in industry standards. As I discuss in §X.D above, RFC-1383 discloses a way to store and return multiple IP addresses in response to a single query using TXT fields or records. This precisely matches the DNS response format of the '785 patent.

333. Thus, a POSITA would understand that the combination of Caronni I, Caronni II, and Hipp discloses a DNS server, request, and responses. A POSITA would have been motivated to combine Caronni I, Caronni II, and Hipp for the reasons discussed in §XII.B below.

**(i) public network address of a route
director/virtual network manager**

334. Caronni II discloses the address of the reflecting agent is added to the VARP table. “In such a case, the present invention adds an entry to the VARP table for a third-party reflecting agent located at an address outside the NAT box.” EX1004 (Caronni II) at 8:19-21. Here, a POSITA would understand an address

outside of a NAT box would be a public network address. Further, a POSITA would have understood that the reflecting agent's address would be returned as part of a VARP address resolution response, because packets destined for the endpoint devices have to traverse the routing agent and be directed by it (else the endpoints, each behind their own NAT, won't be able to communicate.).

335. As a part of the VARP address resolution response, a POSITA would have understood that the reflecting agent's address would be returned. *Id.* at 3:20-23 ("A resolution request is received referring a virtual address destination and the resolution request is resolved using the virtual address resolution facility and the stored associations.").

336. A POSITA would understand that the address of the reflecting agent is returned because the singular purpose of the address resolution facility is "to resolve a virtual IP address into a real IP address." *Id.* at 2:54-55. Further, a POSITA would understand that when resolving a virtual address for a particular node into a real address for node, the real address of the associated reflecting agent must also be known so that (encapsulated) packets could be sent in the direction of the correct reflecting agent who can decapsulate the packets and transmit their inner payloads to the endpoint behind a NAT device. *See* §VII.B.4 above.

337. A POSITA would further recognize that this reflecting agent address would logically be returned with the real address of the resolved virtual node

address. Moreover, Caronni II discloses that the lookup tables associated with the address resolution facility are indexed by virtual addresses of nodes. Caronni II makes no disclosure that the reflecting agent is associated with a virtual address and thus could not be added to the lookup tables similar to the nodes. As a result, a POSITA would understand that the address of the reflecting agent is added to the existing records of nodes that are associated with that reflecting agent and returns the reflecting agent address with the resolution of associated node virtual addresses.

(ii) network address of the destination device

338. Caronni I discloses returning a network address of a destination device as a destination real address. “The SNSL layer then accesses the VARPDB to obtain the address mapping between virtual source node address 642 and the source real address 614 as well as the virtual destination node address 644 and the destination real address 616 (step 808). EX1003 (Caronni I) at 11:49-53.

339. As Caronni I further discloses, these real addresses may be public network addresses. “VARPD 548 has an associated component, VARPDB 551, into which it stores mappings of the internal Supernet addresses, known as a node IDS, to the network addresses recognized by the public-network infrastructure, known as the real addresses.” *Id.* at 7:5-9. Also, Caronni indicates that a real address may be a private address. “The ‘real address’ is an IP address (e.g., 10.0.0.2).” *Id.* at 7:18-19.

340. Caronni II discloses returning a network address of a destination device as a destination real address:

In one embodiment, a virtual network supported by a physical network includes a virtual address resolution facility. The virtual address resolution facility is used to resolve a virtual IP address into a real IP address. A virtual address is registered with the virtual address resolution facility and the registration includes a real IP address, a port number and a transport protocol designation associated with a virtual address. The virtual address destination is resolved using the virtual address resolution facility and a message is sent from a virtual address in the virtual network to the real IP address indicated in the virtual address destination registration.

EX1004 (Caronni II) at 2:52-62.

341. In addition, Caronni II discloses these real addresses may be public network addresses. “For example, the entry 80 for the virtual IP address 10.0.0.12 (90) includes the association of a real IP address 152.70.0.1.” *Id.* at 5:24-26. Additionally, Caronni II discloses that a LAN may be the only infrastructure network supporting a virtual network:

Those skilled in the art will recognize that a number of different types of networks may be utilized within the scope of the present invention. The networks may be the Internet, an intranet, wide area network (WAN), a local area network (LAN), a satellite network, a wireless network, or some other type of network capable of supporting a virtual network through the devices thereon.

Id. at 4:27-33.

342. A POSITA would understand that devices on a LAN would have real addresses that are private addresses. *See also* §VII.B.1 above.

(iii) virtual network address that corresponds to the destination device

343. Caronni I discloses returning a virtual network address that corresponds to the destination device. “Each time web client 1102 requests a packet from web server 1104a, the client requests the virtual address of the web server 1104a from computer system 1106.” EX1003 (Caronni I) at 13:14-17.

344. Similarly, Hipp discloses returning a virtual network address that corresponds to the destination device:

The virtual hostname resolves to the virtual IP address for both the applications registered with the VNI framework as well as those that are not registered. This may require configuration of a name service or OS host configuration files. For example, if an application instance used a virtual IP address of 10.10.0.1 and a virtual hostname of host1055, the standard hostname to IP address resolution mechanisms (e.g. DNS or the /etc/hosts file) would have to be preconfigured to resolve a query of host1055 to IP address 10.10.0.1.

EX1005 (Hipp) at 6:16-26.

345. Thus, a POSITA would understand that the combination of Caronni I, Caronni II, and Hipp discloses a DNS response. A POSITA would have been motivated to combine Caronni I, Caronni II, and Hipp for the reasons discussed in §XII.B below.

B. Motivation To Combine Caronni I, Caronni II, and Hipp

346. A POSITA would have been motivated to combine Caronni I, Caronni II, and Hipp for at least the following three reasons.

347. **First**, a POSITA would recognize that Caronni I, Caronni II, and Hipp are all directed to virtual networks using virtual addressing and virtual address translation (NAT) schemes built on existing network infrastructure:

Methods and systems consistent with the present invention overcome the shortcomings of existing delivery schemes and addressing by **establishing a virtual network on top of current IP network** delivery schemes. The virtual network uses a separate layer to create a modification to the IP packet format that is used to separate network behavior from addressing. As a result of the modification to the packet format, any type of delivery scheme may be **assigned to any address or group of addresses**.

EX1003 (Caronni I) at 3:15-23 (emphases added).

The illustrative embodiment of the present invention provides a method of extending the functionality of **a virtual network**. Messages intended for a **virtual destination address** located on a network equipped with a device performing packet filtering, network address translation or a similar function on the edge of the network (an ‘edge device’), are encapsulated in higher level protocols prior to being sent to the edge device. The **virtual destination address** may be associated with a process on the edge device or a process on another device in the interior of the network.

EX1004 (Caronni II) at 2:31-40 (emphases added).

A virtual network is a logical network overlaid on a physical network.
Id. at 1:13-14.

The **Virtual Network Environment (VNE)** of the present invention is defined by a collection of IP addresses. An application running within one VNE can communicate with another application in the same VNE.

EX1005 (Hipp) at 2:8-11 (emphasis added).

An application IP address is an IP address associated with an application. **It can be a virtual IP address**, associated solely with an

application, or it can be a real IP address, associated with an interface on the host where the application resides.

Id. at 2:58-62 (emphasis added).

The present invention provides **isolation at the application level**, rather than at the host level. As a result, applications residing on the same computer or network can be kept isolated from one another, allowing for secure shared resources.

Id. at 2:3-7 (emphasis added).

348. Further, all three references relate to address translation (virtual to physical) and/or NAT (Network Address Translation):

Fourth, the system provides **address translation** in a transparent manner. Since the Supernet is a **private network** constructed from the infrastructure of another network, the Supernet has its own **internal addressing scheme**, separate from the **addressing scheme of the underlying public network**.

EX1003 (Caronni I) at 6:6-12 (emphases added).

The registered addresses are stored in [sic] an searchable and accessible location on the network, such as by storing them in a **Virtual Address Resolution Protocol** (VARP) lookup table 26. The registrations are used to **resolve the virtual address into the real IP address** of a physical device to which messages may be sent.

EX1004 (Caronni II) at 4:11-16 (emphases added).

The physical network 20 is also interfaced with an electronic device 30 located at the **edge of a local area physical network** 50 which includes devices addressable using IP addresses in the 129.63.1.0/24 range. The electronic device 30 **may be a proxy server, gateway, NAT box, or other device** and may perform packet filtering.

Id. at 4:17-22 (emphases added).

The **virtual hostname resolves to the virtual IP address** for both the applications registered with the VNI framework as well as those that are not registered. This may require configuration of a name Service

or OS host configuration files. For example, if an application instance used a **virtual IP address of 10.10.0.1** and a virtual hostname of host1055, the standard hostname to IP address **resolution mechanisms** (e.g. DNS or the /etc/hosts file) would have to be preconfigured to **resolve a query of host1055 to IP address 10.10.0.1**.

EX1005 (Hipp) at 6:16-18 (emphases added).

349. **Second**, both Caronni I and Caronni II disclose similar systems that enable node to node virtual network coverage using a (i) Virtual Address Resolution Protocol (VARP) scheme, (ii) a virtual address device registration process, and (iii) packet encapsulation. I detail each of these three points next.

1. Virtual network

350. Caronni I and Caronni II describe a Virtual Address Resolution Protocol (VARP) scheme:

These nodes 316, 318, 320, and 322 are communicative entities (e.g., processes) running within a particular device and are able to communicate among themselves as well as access the resources of the Supernet in a secure manner. **When communicating among themselves, the nodes 316, 318, 320, and 322 serve as end points for the communications**, and no other processes or devices that are not part of the Supernet are able to communicate with the Supernet's nodes or utilize the Supernet's resources.

EX1003 (Caronni I) at 5:3-11 (emphasis added).

The illustrative embodiment of the present invention provides a method of extending the functionality of a virtual network. **Messages intended for a virtual destination address** located on a network equipped with a device performing packet filtering, network address translation or a similar function on the edge of the network (an 'edge device'), are encapsulated in higher level protocols prior to being sent to the edge device. The virtual destination address may be associated with a

process on the edge device or **a process on another device in the interior of the network.**

EX1004 (Caronni II) at 2:31-39 (emphases added).

351. Here, a POSITA would have recognized that Caronni I provides foundational architecture for scalable virtual networks, where Caronni I's Supernet framework establishes a logical overlay network that provides secure communications between member nodes. "The virtual network also maintains secure communications between nodes." EX1003 (Caronni I) at 3:23-24.

352. Having the same named lead inventor (and assignee), Caronni II naturally extends and improves this architecture by introducing a versatile reflecting agent to manage communications between private networks having edge devices, such as NAT boxes. A POSITA would recognize the benefit to incorporating the reflecting agent capability of Caronni II into Caronni I's Supernet to achieve communications where "[m]essages that are being sent from the originating virtual address behind a NAT box to a destination which is behind a different NAT box are sent via the reflecting agent intermediary outside the NAT box and reflected to the destination." EX1004 (Caronni II) at 8:19-23.

2. Registration process

353. Both Caronni I and Caronni II disclose registrations processes to register virtual addresses into the respective VARP structures:

In this step, SASD concatenates the Supernet ID with the virtual address to create the node ID, obtains the real address of the SNlogin

script by querying network services in a well-known manner, and then **registers this information with the VARPD** that acts as the server for this Supernet.

EX-1003 (Caronni I) at 10:13-18 (emphasis added).

The virtual address resolution facility is **software used to register**, store and resolve virtual address information for processes and applications executing on the virtual network. The virtual address resolution facility 24 works with stored virtual address registrations for the virtual network. **The registered addresses are stored** in [sic] an searchable and accessible location on the network, such as by storing them in a Virtual Address Resolution Protocol (VARP) lookup table 26.

EX-1004 (Caronni II) at 4:6-14 (emphases added).

3. Packet encapsulation

354. Both Caronni I and Caronni II disclose packet encapsulation when an inner packet having **virtual** source and destination addresses is encapsulated in an outer packet having **real** source and destination addresses:

The Supernet contains a **modification to the IP packet format** that can be used to separate network behavior from addressing. As a result of the modification, any delivery scheme may be assigned to any address, or group of addresses.

EX1003 (Caronni I) at 4:55-59 (emphasis added).

The first step performed is for **inner IP layer** 540 to receive a packet originating from node A via the TCP/UDP layer 538 (step 802). **The packet contains virtual source node address 642, virtual destination node address 644**, and data 654.

Id. at 11:26-30 (emphases added).

Outer IP header 610 contains a next header field 612, IP source node address 614, and IP destination node address 616. Next header field 612 identifies the type of the next payload after outer IP header 610,

such as Supernet header 620. Source real address 614 contains the **real address of the originating node** of Supernet packet 600. Destination real address 616 contains the **real address of the destination node** of Supernet packet 600.

Id. at 9:13-20 (emphases added).

SNlogin 522 is a script used for logging into a Supernet. Successfully executing this Script results in a Unix shell from which programs (e.g., node A522) can be started to run within the Supernet context, such that address translation and security **encapsulation is performed**.

Id. at 8:31-35 (emphasis added).

FIG. 3 depicts a block diagram of **the encapsulation process** used to send a message from a virtual IP address to another virtual IP address located at the interior of a network at which packet filtering and/or network address translation is occurring.

EX1004 (Caronni II) at 6:35-39 (emphasis added).

Those skilled in the art will realize that the message may include an **IPSEC** header, inner IP header, UDP or TCP header and inner payload.

Id. at 6:51-53 (emphasis added).

The Transport layer 112 takes the original message 111 and adds a transport protocol header ... At the Network layer 114, the resolved real IP address which has been retrieved from the VARP lookup table 26 is added to the transport protocol header 113 and the original message 111. Following the addition of the IP header 115, the Network layer 114 passes the IP header transport protocol header 113 and message 111 down to the Link layer 116.

Id. at 6:45-62.

355. Here, a POSITA would understand that the packet encapsulation processes of both Caronni I and Caronni II are performed with the same intent. *See* §VII.B.4 above. Namely, to achieve a separation of network behavior from addressing:

It should also be noted that since a Supernet is layered on top of an existing network, it operates independently of the transport layer. Thus, the nodes of a Supernet may communicate over different transports, such as IP, IPX, X.25, or ATM, as well as different physical layers, such as RF communication, cellular communication, satellite links, or land-based links.

EX1003 (Caronni I) at 5:19-25.

The transport protocol header 113 may be a UDP header, TCP header, an X.25 header, XTP header, AppleTalk header, or other similar transport protocol header. Those skilled in the art will realize that the message may include an IPSEC header, inner IP header, UDP or TCP header and inner payload.

EX1004 (Caronni II) at 6:48-53.

356. Although Caronni II operates at a higher abstraction level than Caronni I, its disclosure of analogous components—virtual addressing, encapsulation, and node registration—provides overlapping technical objectives. A POSITA would recognize that integrating Caronni II's reflecting agent with Caronni I's Supernet structure directly addresses NAT-to-NAT communication challenges. The reflecting agent's ability to map public-facing IP addresses to private virtual addresses aligns with Caronni I's VARP system, enabling bidirectional connectivity across NAT boundaries. *Id.*, Fig. 5. The combination of Caronni I and Caronni II results in the inclusion of the reflecting agent's IP address in Caronni I's VARP system. *Id.*, 8:35–38. This combination enhances Caronni I's virtual addressing scheme by adding NAT transparency through Caronni II's reflecting agent, fulfilling a well-documented industry need for scalable hybrid networks.

357. **Third**, all three references include similar and complimentary DNS functionality. As an initial matter, the Domain Name System (DNS) is a foundational Internet technology, so much so that it was one of the very first IETF standards. Work on standardizing domain names began in 1981, with the publication of RFC-799, titled “Internet Name Domains”. *See* <https://datatracker.ietf.org/doc/html/rfc799>.

358. As I detail in §VII.B.3 above, several more RFCs ensued and provide extensive detail on DNS. Even the ’785 patent recognized the importance and ubiquity of DNS that it refers the reader for “[m]ore information about Dynamic DNS can be found in RFC-2136 incorporated herein by reference.” EX1001 (’785 patent) at 5:12-14. Indeed, without being able to name computer hosts and address them uniquely, the Internet would not even exist.

359. Similarly, Hipp describes DNS as a standard mechanism. “For example, if an application instance used a virtual IP address of 10.10.0.1 and a virtual hostname of host1055, the **standard hostname to IP address resolution mechanisms (e.g. DNS or the /etc/hosts file)** would have to be preconfigured to resolve a query of host1055 to IP address 10.10.0.1.” EX1005 (Hipp) at 6:20-25 (emphasis added).

360. Caronni I also refers to DNS: “In this step, SASD concatenates the Supernet ID with the virtual address to create the node ID, **obtains the real address**

of the SNlogin script by querying network services in a well-known manner and then registers this information with the VARPd that acts as the server for this Supernet.” EX1003 (Caronni I) at 10:13-18 (emphasis added). A POSITA would understand that DNS is a well-known network service used to query for real addresses.

361. A POSITA would recognize that DNS has been in use and well known for decades prior to 2003, and would have easily sought out additional DNS details in the various RFCs and textbooks written on the subject. *See* §VII.B.3 above.

362. Caronni I further describes the VARPd mechanism as “accessing” to “obtain” the “destination real address”:

The SNSL layer then **accesses** the VARPDB **to obtain** the address mapping between virtual source node address 642 and the source real address 614 as well as the virtual destination node address 644 and the **destination real address** 616 (step 808).

EX1003 (Caronni I) at 4:27-33 (emphases added).

363. Here, Caronni I further discloses the virtual source node address may be an email address. “Although the virtual address is described in an IP address scheme, one skilled in the art will appreciate that the virtual address may be any other type addressing scheme, **such as an e-mail address**, IPX, or IPv6.” *Id.* at 7:13-16 (emphasis added).

364. An email address has three main parts, a username, the “@” symbol, and the domain name. Using the example `jdoe@mycompany.com`, `jdoe` is the

username and mycompany.com is the domain name. Thus, a POSITA would understand that Caronni I's VARPD mechanism, using an email address as virtual source node address, is a query using a domain name (mycompany.com) resolved to a virtual IP address. This is equivalent to a typical DNS query/response using a host name that is resolved to an IP address. *See* §VII.B.3 above.

365. Furthermore, Caronni II discloses the same VARP mechanism:

The virtual address resolution facility is used to resolve a virtual IP address into a real IP address. A virtual address is registered with the virtual address resolution facility and the registration includes a real IP address, a port number and a transport protocol designation associated with a virtual address. The virtual address destination is resolved using the virtual address resolution facility

EX1004 (Caronni II) at 2:54-60.

FIG. 2 depicts a block diagram of the VARP lookup table 26 used by the virtual address resolution facility 24. The VARP lookup table 26 may be stored at any location accessible over the network. The VARP lookup table 26 includes registered virtual destination addresses located in a column 78 and corresponding associations located in a column 79.

Id. at 5:10-15.

366. However, Caronni I does not detail how this virtual destination node address is known, only that the packet containing the address came from a process executing in the sending node. *See* §a above above. A POSITA would have understood that higher layer processes in a Caronni I sending node could use DNS services, such as those disclosed by Hipp or those generally known to a POSITA, to determine a virtual destination node address.

367. Hipp's DNS hostname resolution complements these frameworks by introducing DNS functionality to virtual hostnames. "The virtual hostname resolves to the virtual IP address for both the applications registered with the VNI framework as well as those that are not registered." EX1005 (Hipp) at 6:15-24. By incorporating Hipp's hostname resolution capability into Caronni I's VARPD operations and Caronni II's VARP reflecting agent entries, a POSITA would appreciate a DNS capability to resolve a virtual hostname into a virtual IP address, a real IP address and an IP address of a reflecting agent. This integration would leverage Hipp's DNS capability for virtual hostnames to resolve to multiple address records from a single query, providing the capability to route virtual IP packets over an underlying network infrastructure. A POSITA would have understood that the Hipp DNS hostname resolution capability fills a gap in both the Caronni I and Caronni II systems, where a virtual IP address of a destination node or device is not known.

368. Finally, this implementation would have been simple and would not have required extensive experimentation or impose undue burden, because DNS is well known, standardized, and widely used. For the same reasons, the combination would have yielded predictable results.

369. Furthermore, the reasonable expectation of success of this combination stems from the complementary nature of the three patents: (1) Caronni I's structural

blueprint, (2) Caronni II's NAT traversal solution, and (3) Hipp's DNS resolution mechanics. A POSITA would recognize that DNS enhancements (Hipp) logically apply to virtual addressing systems (Caronni I/II) given industry-standard practices of coupling name resolution with routing, while reflecting agents (Caronni II) provide a well-understood method for bridging NAT-dependent environments.

C. The Dependent Claims

1. Claims 35, 36, 37, 77, and 78 are obvious and unpatentable over the combination of Caronni I, Caronni II, and Hipp

370. Claims 35, 36, 37, 77, and 78 are dependents of independent Claims 30 and 75 and are identical or substantially similar. For ease of discussion, I discuss them together as follows.

- a. Claim 35: The virtual network manager of claim 30 further comprising a join module configured to receive a join request from the agent associated with the device to indicate that the device is connected for data communication within the virtual network, the join module further configured to receive a leave request from the agent associated with the device to indicate that the device will be disconnected from data communication within the virtual network.**
- b. Claim 77: The system of claim 75 wherein the virtual network manager includes a member join module.**

371. Caronni I discloses an administrative node that is responsive to SNlogin and SNlogout scripts that provide access to a Supernet, where "SNlogin 522 is a script used for logging into a Supernet" and "SNlogout 524 is a Script used for

logging out of a Supernet.” EX1003 (Caronni I) at 8:31; 8:42. A POSITA would understand that these scripts are agents associated with a device.

372. Further, Caronni I discloses the first time a node accesses or is accessed over a Supernet, the local VARPDB (part of the device agent) sends a message to the administrative node to obtain its address mappings. A POSITA would have understood that this message (during login) is an indication that the device is connected (joined) for data communication within the virtual network:

The SNSL layer then accesses the VARPDB to obtain the address mapping between virtual source node address 642 and the source real address 614 as well as the virtual destination node address 644 and the destination real address 616 (step 808). If they are not contained in the VARPDB because **this is the first time a packet has been sent from this node or sent to this destination**, the VARPDB accesses the local VARPD to obtain the mapping. When contacted, the VARPD on **the local machine contacts the VARPD that acts as the server for the Supernet** to obtain the appropriate address mapping. Since the VARPDB maintains all real IP addresses, a remote node may securely communicate with another remote node without reverification.

Id. at 11:49-61 (emphases added).

373. Similarly, Caronni I discloses devices logging out of a Supernet:

FIG. 10 depicts a flow chart of the steps performed when logging a node out of a Supernet. The first step performed is for the user to run the SNlogout script and to enter a node ID (step 1002). Next, the SNlogout script requests a log out from SASD (step 1004). Upon receiving this request, SASD removes the mapping for this node from the VARPD that acts as the server for the Supernet (step 1006). SASD then informs KMS to cancel the registration of the node, and KMS terminates this KMC (step 1008).

Id. at 12:65-13:2.

374. Here, a POSITA would understand that the log out message from SNlogout is a leave request from the device agent to indicate that the device will be disconnected from further data communications.

375. Thus, a POSITA would understand that the Caronni I, Caronni II, and Hipp registration processes disclose the claimed join limitations through the exchange of information by the initial local VARPDB message and the SNlogout message. A POSITA would have been motivated to combine Caronni I, Caronni II, and Hipp for the reasons discussed in §XII.B above.

- c. **Claim 36: The virtual network manager of claim 35 wherein the join module is further configured to provide virtual network addresses to the devices that are registered in the virtual network.**
- d. **Claim 78: The system of claim 77 wherein the member join module provides a virtual network address to a device that is registered as a member of the network.**

376. *See* Claim 35 in §XII.C.1.a. As I discuss in claim 35, “the local machine contacts the VARPDB that acts as the server for the Supernet to obtain the appropriate address mapping.” EX1003 (Caronni I) at 11:56-59. Here, a POSITA would understand that the administrator node thus provides the a virtual network address to a registered device. A POSITA would further understand that only registered devices have addresses in the server VARPDB as I discuss in Claim 35 with regard to SNlogout.

377. Thus, a POSITA would understand that the Caronni I, Caronni II, and Hipp registration processes disclose the claimed join limitations that provide virtual network addresses to the devices that are registered. A POSITA would have been motivated to combine Caronni I, Caronni II, and Hipp for the reasons discussed in §XII.B above.

- e. **Claim 37: The virtual network manager of claim 35 wherein the join module is further configured to maintain data to associate a virtual network address with a device in the virtual network.**

378. See Claim 35 in §XII.C.1.a. Further, Caronni I's admin node contains a server VARPDDB (join module and part of administrative node) that stores and maintains device information, including addresses in mapping tables. EX1003 (Caronni I) at 7:5–9 (“VARPD 548 has an associated component, VARPDDB 551, into which it stores mappings of the internal Supernet addresses, known as a node IDs, to the network addresses recognized by the public-network infrastructure, known as real addresses.”).

379. Thus, a POSITA would understand that the Caronni I, Caronni II, and Hipp registration processes disclose the claimed join limitations that maintain data to associate a virtual network address with a device in the virtual network. A POSITA would have been motivated to combine Caronni I, Caronni II, and Hipp for the reasons discussed in §XII.B above.

XIII. GROUND 2: Claims 1, 30, 35, 36, 37, 38, 48, 62, 75, 77, and 78 are obvious in view of Caronni I in view of Caronni II and in further view of RFC-1383

A. The Independent Claims

380. The same independent and dependent claims as discussed in Ground 1 are discussed in the same manner below with respect to the Ground 2 references. These limitations would have been obvious to a POSITA before the time of the filing of the '785 Patent, in light of Caronni I combined with Caronni II and RFC-1383.

1. preamble: virtual network system or manager

381. Caronni I and Caronni II disclose these limitations as discussed in §XII.A.1 above.

382. Thus, a POSITA would understand that the combination of Caronni I, Caronni II, and RFC-1383 discloses a virtual network system or manager. A POSITA would have been motivated to combine Caronni I, Caronni II, and RFC-1383 for the reasons discussed in §XII.B above.

2. a virtual network defined by a domain name having an associated public network address

383. Caronni I and Caronni II disclose this limitation as discussed in §XII.A.2.a and §XII.A.2.b above.

384. Thus, a POSITA would understand that the combination of Caronni I, Caronni II, and RFC-1383 discloses a virtual network defined by a domain name having an associated public network address. A POSITA would have been

motivated to combine Caronni I, Caronni II, and RFC-1383 for the reasons discussed in §XII.B above.

3. registration/distribution of virtual network address to each device which uniquely identifies the device

385. Caronni I and Caronni II disclose this limitation as discussed in §XII.A.3.a through §XII.A.3.g above.

386. Thus, a POSITA would understand that the combination of Caronni I, Caronni II, and RFC-1383 discloses a domain name having an associated public network address. A POSITA would have been motivated to combine Caronni I, Caronni II, and RFC-1383 for the reasons discussed in §XII.B above.

4. route director/routing encapsulated packets

387. Caronni I and Caronni II disclose this limitation as discussed in §XII.A.4.a and §XII.A.4.b above.

388. Thus, a POSITA would understand that the Caronni I and Caronni II registration processes disclose the route director/routing encapsulated packet limitations through the exchange of information. A POSITA would have been motivated to combine Caronni I, Caronni II, and RFC-1383 for the reasons discussed in §XII.B above.

5. DNS server, request, and responses

389. Caronni I and Caronni II disclose this limitation as discussed in §XII.A.5.a through XII.A.5.c above.

390. Further, RFC-1383 discloses this limitation as follows:

a. DNS server for the virtual network.

391. RFC-1383 discloses a generalized scheme for packet routing using a standard DNS with a defined DNS record. “In the scheme that we propose, the DNS is only accessed once” and “will use the general purpose ‘TXT’ record.” EX1006 (RFC-1383) at pp. 5, 11; *see* §XII.B above.

In a definitive scheme, it would be necessary to define a DNS record type and the corresponding “RX” format. In order to deploy this scheme, we would then have to teach this new format to the **domain name service software**. While not very difficult to do, this would probably take a couple of month, and will not be used in the early experimentations, which will **use the general purpose “TXT” record**. EX1006 (RFC-1383) at p. 11 (emphases added).

392. Thus, a POSITA would understand that the Caronni I, Caronni II, and RFC-1383 combination discloses a DNS server. A POSITA would have been motivated to combine Caronni I, Caronni II, and RFC-1383 for the reasons discussed in §XII.B above.

b. DNS request

393. RFC-1383 discloses DNS records, which a POSITA would understand is returned from a DNS in response to a DNS request:

In a definitive scheme, it would be necessary to **define a DNS record type and the corresponding “RX” format**. In order to deploy this scheme, we would then have to teach this new format to the domain name service software. While not very difficult to do, this would probably take a couple of month, and will not be used in the early experimentations, which will **use the general purpose “TXT” record**.

EX1006 (RFC-1383) at p. 11 (emphases added).

[T]he query manager will send a **DNS request**, in order to read the **RX records** corresponding to the destination. After reception of the **response**, it will select a gateway, and pass the information to the real time forwarder.

EX1006 (RFC-1383) at p. 12 (emphases added).

c. DNS response

394. The '785 patent describes a modification to a standard DNS response to include additional information beyond the standard resolved virtual IP address. "The response includes the following information: ... the target member virtual IP address; ... the target member private IP address; ... the target Route Director public IP address; the source Route Director public IP address." EX1001 ('785 patent) at 22:38-50. Furthermore,

The DNS operation is conducted using the DNS protocol with the VCN Manager. The member software sends a DNS query to the VCN Manager, and the manager returns information about the target member. Besides the target member's virtual IP address, the DNS response also contains additional information about the target member.

Id. at 22:1-6.

The target member's virtual IP address will be presented using an A-type resource record in the answer section of the normal DNS response packet. All the remaining information will be transferred using a TXT-type resource record in an additional section of the DNS Response.

Id. at 22:53-57.

In order to route packets to a peer in a different addressing realm, the protocol stack knows the address of the appropriate Route Director that serves the peers realm, the private address of the peer or a NAT address. The Member Agent obtains the information by using the VCN Manager

as a DNS authority for DNS lookup operations as described below (FIGS. 13 and 21). The DNS response from the VCN Manager will include the above-identified information in an optional text field of the response to the Member Agent.

Id. at 12:23-31.

395. A POSITA would understand this additional information disclosed by the '785 patent is well known in industry standards. As I discuss in §X.D above, RFC-1383 discloses a way to store and return multiple IP addresses in response to a single query using TXT fields or records. This precisely matches the DNS response format of the '785 patent.

396. Thus, a POSITA would understand that the combination of Caronni I, Caronni II, and RFC-1383 discloses a DNS server, request, and responses. A POSITA would have been motivated to combine Caronni I, Caronni II, and RFC-1383 for the reasons discussed in §XIII.B below.

B. Motivation To Combine Caronni I, Caronni II, and RFC-1383

397. A POSITA would have been motivated to combine Caronni I, Caronni II, and RFC-1383 for at least the following three reasons.

398. **First**, a POSITA would recognize that Caronni I and Caronni II are both directed to virtual networks using virtual addressing and virtual address translation schemes built on existing network infrastructure:

Methods and systems consistent with the present invention overcome the shortcomings of existing delivery schemes and addressing by **establishing a virtual network on top of current IP network** delivery schemes. The virtual network uses a separate layer to create a

modification to the IP packet format that is used to separate network behavior from addressing. As a result of the modification to the packet format, any type of delivery scheme may be **assigned to any address or group of addresses**.

EX1003 (Caronni I) at 3:15-23 (emphases added).

The illustrative embodiment of the present invention provides a method of extending the functionality of **a virtual network**. Messages intended for a **virtual destination address** located on a network equipped with a device performing packet filtering, network address translation or a similar function on the edge of the network (an 'edge device'), are encapsulated in higher level protocols prior to being sent to the edge device. The **virtual destination address** may be associated with a process on the edge device or a process on another device in the interior of the network.

EX1004 (Caronni II) at 2:31-40 (emphases added).

A virtual network is a logical network overlaid on a physical network. *Id.* at 1:13-14.

399. Furthermore, Caroni I and Caroni II relate to address translation (virtual to physical) and/or NAT (Network Address Translation):

Fourth, the system provides **address translation** in a transparent manner. Since the Supernet is a **private network** constructed from the infrastructure of another network, the Supernet has its own **internal addressing scheme**, separate from the **addressing scheme of the underlying public network**.

EX1003 (Caronni I) at 6:6-12 (emphases added).

The registered addresses are stored in [sic] an searchable and accessible location on the network, such as by storing them in a **Virtual Address Resolution Protocol** (VARP) lookup table 26. The registrations are used to **resolve the virtual address into the real IP address** of a physical device to which messages may be sent.

EX1004 (Caronni II) at 4:11-16 (emphases added).

The physical network 20 is also interfaced with an electronic device 30 located at the **edge of a local area physical network 50** which includes devices addressable using IP addresses in the 129.63.1.0/24 range. The electronic device 30 **may be a proxy server, gateway, NAT box, or other device** and may perform packet filtering.

Id. at 4:17-22 (emphases added).

400. **Second**, both Caronni I and Caronni II disclose similar systems that enable node to node virtual network coverage using a (i) Virtual Address Resolution Protocol (VARP) scheme, (ii) a virtual address device registration process, and (iii) packet encapsulation. I detail each of these three points next.

1. Virtual network

401. Caronni I and Caronni II describe a Virtual Address Resolution Protocol (VARP) scheme:

These nodes 316, 318, 320, and 322 are communicative entities (e.g., processes) running within a particular device and are able to communicate among themselves as well as access the resources of the Supernet in a secure manner. **When communicating among themselves, the nodes 316, 318, 320, and 322 serve as end points for the communications**, and no other processes or devices that are not part of the Supernet are able to communicate with the Supernet's nodes or utilize the Supernet's resources.

EX1003 (Caronni I) at 5:3-11 (emphasis added).

The illustrative embodiment of the present invention provides a method of extending the functionality of a virtual network. **Messages intended for a virtual destination address** located on a network equipped with a device performing packet filtering, network address translation or a similar function on the edge of the network (an 'edge device'), are encapsulated in higher level protocols prior to being sent to the edge device. The virtual destination address may be associated with a

process on the edge device or **a process on another device in the interior of the network.**

EX1004 (Caronni II) at 2:31-39 (emphases added).

402. Here, a POSITA would have recognized that Caronni I provides foundational architecture for scalable virtual networks, where Caronni I's Supernet framework establishes a logical overlay network that provides secure communications between member nodes. "The virtual network also maintains secure communications between nodes." EX1003 (Caronni I) at 3:23-24.

403. Having the same named lead inventor (and assignee), Caronni II naturally extends and improves this architecture by introducing a versatile reflecting agent to manage communications between private networks having edge devices, such as NAT boxes. A POSITA would recognize the benefit to incorporating the reflecting agent capability of Caronni II into Caronni I's Supernet to achieve communications where "[m]essages that are being sent from the originating virtual address behind a NAT box to a destination which is behind a different NAT box are sent via the reflecting agent intermediary outside the NAT box and reflected to the destination." EX1004 (Caronni II) at 8:19-23.

2. Registration process

404. Both Caronni I and Caronni II disclose registrations processes to register virtual addresses into the respective VARP structures:

In this step, SASD concatenates the Supernet ID with the virtual address to create the node ID, obtains the real address of the SNlogin

script by querying network services in a well-known manner, and then **registers this information with the VARPD** that acts as the server for this Supernet.

EX-1003 (Caronni I) at 10:13-18 (emphasis added).

The virtual address resolution facility is **software used to register**, store and resolve virtual address information for processes and applications executing on the virtual network. The virtual address resolution facility 24 works with stored virtual address registrations for the virtual network. **The registered addresses are stored** in [sic] an searchable and accessible location on the network, such as by storing them in a Virtual Address Resolution Protocol (VARP) lookup table 26.

EX-1004 (Caronni II) at 4:6-14 (emphases added).

3. Packet encapsulation

405. Both Caronni I and Caronni II disclose packet encapsulation when an inner packet having **virtual** source and destination addresses is encapsulated in an outer packet having **real** source and destination addresses:

The Supernet contains a **modification to the IP packet format** that can be used to separate network behavior from addressing. As a result of the modification, any delivery scheme may be assigned to any address, or group of addresses.

EX1003 (Caronni I) at 4:55-59 (emphasis added).

The first step performed is for **inner IP layer** 540 to receive a packet originating from node A via the TCP/UDP layer 538 (step 802). **The packet contains virtual source node address 642, virtual destination node address 644**, and data 654.

Id. at 11:26-30 (emphases added).

Outer IP header 610 contains a next header field 612, IP source node address 614, and IP destination node address 616. Next header field 612 identifies the type of the next payload after outer IP header 610,

such as Supernet header 620. Source real address 614 contains the **real address of the originating node** of Supernet packet 600. Destination real address 616 contains the **real address of the destination node** of Supernet packet 600.

Id. at 9:13-20 (emphases added).

FIG. 3 depicts a block diagram of **the encapsulation process** used to send a message from a virtual IP address to another virtual IP address located at the interior of a network at which packet filtering and/or network address translation is occurring.

EX1004 (Caronni II) at 6:35-39 (emphasis added).

Those skilled in the art will realize that the message may include an **IPSEC** header, inner IP header, UDP or TCP header and inner payload.

Id. at 6:51-53 (emphasis added).

The Transport layer 112 takes the original message 111 and adds a transport protocol header ... At the Network layer 114, the resolved real IP address which has been retrieved from the VARP lookup table 26 is added to the transport protocol header 113 and the original message 111. Following the addition of the IP header 115, the Network layer 114 passes the IP header transport protocol header 113 and message 111 down to the Link layer 116.

Id. at 6:45-62.

SNlogin 522 is a script used for logging into a Supernet. Successfully executing this Script results in a Unix shell from which programs (e.g., node A522) can be started to run within the Supernet context, such that address translation and security **encapsulation is performed**.

EX1003 (Caronni I) at 8:31-35 (emphasis added).

406. Here, a POSITA would understand that the packet encapsulation processes of both Caronni I and Caronni II are performed with the same intent. Namely, to achieve a separation of network behavior from addressing:

It should also be noted that since a Supernet is layered on top of an existing network, it operates independently of the transport layer. Thus, the nodes of a Supernet may communicate over different transports, such as IP, IPX, X.25, or ATM, as well as different physical layers, such as RF communication, cellular communication, satellite links, or land-based links.

EX1003 (Caronni I) at 5:19-25.

The transport protocol header 113 may be a UDP header, TCP header, an X.25 header, XTP header, AppleTalk header, or other similar transport protocol header. Those skilled in the art will realize that the message may include an IPSEC header, inner IP header, UDP or TCP header and inner payload.

EX1004 (Caronni II) at 6:48-53.

407. Although Caronni II operates at a higher abstraction level than Caronni I, its disclosure of analogous components—virtual addressing, encapsulation, and node registration—provides overlapping technical objectives. A POSITA would recognize that integrating Caronni II's reflecting agent with Caronni I's Supernet structure directly addresses NAT-to-NAT communication challenges. The reflecting agent's ability to map public-facing IP addresses to private virtual addresses aligns with Caronni I's VARP system, enabling bidirectional connectivity across NAT boundaries. *Id.* at Fig. 5. The combination of Caronni I and Caronni II results in the inclusion of the reflecting agent's IP address in Caronni I's VARP system. *Id.* at 8:35–38. This combination enhances Caronni I's virtual addressing scheme by adding NAT transparency through Caronni II's reflecting agent, fulfilling a well-documented industry need for scalable hybrid networks. *See* §VII.B.4 above.

408. **Third**, all three references include similar and complimentary DNS functionality. As an initial matter, the Domain Name System (DNS) is a foundational Internet technology, so much so that it was one of the very first IETF standards. Work on standardizing domain names began in 1981, with the publication of RFC-799, titled “Internet Name Domains”. *See* <https://datatracker.ietf.org/doc/html/rfc799>.

409. As I detail in §VII.B.3 above, several more RFCs ensued and provide extensive detail on DNS. Even the ’785 patent recognized the importance and ubiquity of DNS that it refers the reader for “[m]ore information about Dynamic DNS can be found in RFC-2136 incorporated herein by reference.” EX1001 (’785 patent) at 5:12-14. Indeed, without being able to name computer hosts and address them uniquely, the Internet would not even exist.

410. Caronni I also refers to DNS: “In this step, SASD concatenates the Supernet ID with the virtual address to create the node ID, **obtains the real address of the SNlogin script by querying network services in a well-known manner** and then registers this information with the VARPD that acts as the server for this Supernet.” EX1003 (Caronni I) at 10:13-18 (emphasis added). A POSITA would understand that DNS is a well-known network service used to query for real addresses.

411. A POSITA would recognize that DNS has been in use and well known for decades prior to 2003, and would have easily sought out additional DNS details in the various RFCs and textbooks written on the subject. *See* §VII.B.3 above.

412. Caronni I further describes the VARPD mechanism as “accessing” to “obtain” the “destination real address”:

The SNSL layer then **accesses** the VARPDB **to obtain** the address mapping between virtual source node address 642 and the source real address 614 as well as the virtual destination node address 644 and the **destination real address** 616 (step 808).

EX1003 (Caronni I) at 4:27-33 (emphases added).

413. Here, Caronni I further discloses the virtual source node address may be an email address. “Although the virtual address is described in an IP address scheme, one skilled in the art will appreciate that the virtual address may be any other type addressing scheme, **such as an e-mail address**, IPX, or IPv6.” *Id.* at 7:13-16 (emphasis added).

414. An email address has three main parts, a username, the “@” symbol, and the domain name. Using the example `jdoe@mycompany.com`, `jdoe` is the username and `mycompany.com` is the domain name. Thus, a POSITA would understand that Caronni I’s VARPD mechanism, using an email address as virtual source node address, is a query using a domain name (`mycompany.com`) resolved to a virtual IP address. This is equivalent to a typical DNS query/response using a host name that is resolved to an IP address.

415. Furthermore, Caronni II discloses the same VARP mechanism:

The virtual address resolution facility is used to resolve a virtual IP address into a real IP address. A virtual address is registered with the virtual address resolution facility and the registration includes a real IP address, a port number and a transport protocol designation associated with a virtual address. The virtual address destination is resolved using the virtual address resolution facility

EX1004 (Caronni II) at 2:54-60.

FIG. 2 depicts a block diagram of the VARP lookup table 26 used by the virtual address resolution facility 24. The VARP lookup table 26 may be stored at any location accessible over the network. The VARP lookup table 26 includes registered virtual destination addresses located in a column 78 and corresponding associations located in a column 79.

Id. at 5:10-15.

416. However, Caronni I does not detail how this virtual destination node address is known, only that the packet containing the address was from a process executing in the sending node. *See* §a above above. A POSITA would have understood that higher layer processes in a Caronni I sending node could use DNS services, such as those disclosed by RFC-1383 or those generally known to a POSITA, to determine a virtual destination node address.

417. Given the Caronni I and Caronni II combination, where a POSITA would understand the use of standard DNS practices, a POSITA would further look to RFC-1383 to complement these frameworks by introducing a modified DNS functionality to virtual hostnames and multiple real IP addresses returned by a DNS service. A POSITA would have recognized that RFC-1383 goes into more detail

about the modified DNS server by disclosing a way to identify an address of the destination in the domain part of the DNS request and response, and to store and return multiple IP addresses in response to a single query.

418. RFC-1383 proposes using “TXT” DNS records to store that information. A POSITA would have understood that TXT records are a well understood way to store any arbitrary information. *See* §VII.B.3 above.

419. “This [TXT] record is designed for easy general purpose extensions in the DNS, and its content is a text string. The RX record will contain three fields: - A record identifier composed of the two characters ‘RX’. This is used to disambiguate from other experimental uses of the ‘TXT’ record. - A cost indicator, encoded on up to 3 numerical digits. The corresponding positive integer value should be less than 256, in order to preserve future evolutions. - An IP address, encoded as a text string following the ‘dot’ notation.” EX1006 (RFC-1383) at p. 11.

420. RFC-1383 shows an example TXT record:

domain	type	record	value
*.27.32.192.in-addr.arpa	IP	TXT	RX, 10, 10.0.0.7

Id.

421. RFC-1383 explains the table: “that means for all hosts whose IP address starts by the three octets ‘192.32.27’ the IP host ‘10.0.0.7’ can be used as a gateway, and that the preference value is 10.” *Id.* at p. 12.

422. A POSITA would have appreciated that TXT records are just a domain field containing a destination address, and a string value comprising comma-delimited fields, including an IP address. A POSITA would have found it trivial to use such TXT records to return any number of comma-delimited IP addresses, values, and any information disclosed in the combination of Caronni I and Caronni II.

423. I note that even the '785 patent contemplates using TXT records in very similar ways as disclosed by RFC-1383: a resource record with auxiliary information stored in TXT records. “The target member’s virtual IP address will be presented using an A-type resource record in the answer section of the normal DNS response packet. All the remaining information will be transferred using a TXT-type resource record in an additional section of the DNS Response.” EX1001 ('785 Patent) at 22:53-57.

424. Parsing data strings using delimiters such as commas is a well-known technique often taught to first year computer science students. This would be well known to any user who used a spreadsheet file formatted as a comma-separated vector (CSV file). Indeed a “comma delimited file” is “[a] data file consisting of fields and records, stored as text, in which the fields are separated from each other by commas. Use of comma-delimited files allows communication between database systems that use different formats. If the data in a field contains a comma, the field

is further surrounded with quotation marks.” EX1014 (Microsoft Computer Dictionary) at pp. 100-101.

425. In sum, given how well-known DNS is, a POSA would have found it easy and desirable to modify the combination of Caronni I and Caronni II with RFC-1383’s TXT record disclosures for routing IP packets using DNS because RFC-1383 provides additional detail about modifications to a DNS server of the Caronni I / Caronni II combination that would enable multiple addresses to be returned as part of the DNS request and response utilizing standard “TXT” DNS records.

426. This change would have been simple and would not have required extensive experimentation or impose undue burden, because DNS is well known, standardized, and widely used. For the same reasons, the combination would have yielded predictable results. Lastly, a POSITA would have had a high expectation of success because RFC-1383 merely provides details of one form of returning IP addresses out of several that Caronni I and Caronni II describe.

C. The Dependent Claims

1. Claims 35, 36, 37, 77, and 78 are obvious and unpatentable over the combination of Caronni I, Caronni II, and RFC-1383

427. Claims 35, 36, 37, 77, and 78 are dependents of independent Claims 30 and 75 and are identical or substantially similar. For ease of discussion, I discuss them together as follows:

- a. **Claim 35: The virtual network manager of claim 30 further comprising a join module configured to receive a join request from the agent associated with the device to indicate that the device is connected for data communication within the virtual network, the join module further configured to receive a leave request from the agent associated with the device to indicate that the device will be disconnected from data communication within the virtual network.**
- b. **Claim 77: The system of claim 75 wherein the virtual network manager includes a member join module.**

428. Caronni I discloses an administrative node that is responsive to SNlogin and SNlogout scripts that provide access to a Supernet, where “SNlogin 522 is a script used for logging into a Supernet” and “SNlogout 524 is a Script used for logging out of a Supernet.” EX1003 (Caronni I) at 8:31; 8:42. A POSITA would understand that these scripts are agents associated with a device.

429. Furthermore, Caronni I discloses the first time a node accesses or is accessed over a Supernet, the local VARPDB (part of the device agent) sends a message to the administrative node to obtain its address mappings. A POSITA would have understood that this message (during login) is an indication that the device is connected (joined) for data communication within the virtual network:

The SNSL layer then accesses the VARPDB to obtain the address mapping between virtual source node address 642 and the source real address 614 as well as the virtual destination node address 644 and the destination real address 616 (step 808). If they are not contained in the VARPDB because **this is the first time a packet has been sent from this node or sent to this destination**, the VARPDB accesses the local

VARPD to obtain the mapping. When contacted, the VARPD on **the local machine contacts the VARPD that acts as the server for the Supernet** to obtain the appropriate address mapping. Since the VARPDB maintains all real IP addresses, a remote node may securely communicate with another remote node without reverification.

Id. at 11:49-61 (emphases added).

430. Similarly, Caronni I discloses devices logging out of a Supernet:

FIG. 10 depicts a flow chart of the steps performed when logging a node out of a Supernet. The first step performed is for the user to run the SNlogout script and to enter a node ID (step 1002). Next, the SNlogout script requests a log out from SASD (step 1004). Upon receiving this request, SASD removes the mapping for this node from the VARPD that acts as the server for the Supernet (step 1006). SASD then informs KMS to cancel the registration of the node, and KMS terminates this KMC (step 1008).

Id. at 12:65-13:2.

431. Here, a POSITA would understand that the log out message from SNlogout is a leave request from the device agent to indicate that the device will be disconnected from further data communications.

432. Thus, a POSITA would understand that the Caronni I and Caronni II registration processes disclose the claimed join limitations through the exchange of information by the initial local VARPDB message and the SNlogout message. A POSITA would have been motivated to combine Caronni I, Caronni II, and RFC-1383 for the reasons discussed in §XIII.B above.

- c. **Claim 36: The virtual network manager of claim 35 wherein the join module is further configured to provide virtual network addresses to the devices that are registered in the virtual network.**
- d. **Claim 78: The system of claim 77 wherein the member join module provides a virtual network address to a device that is registered as a member of the network.**

433. *See* Claim 35 in §XIII.C.1.a above. As I discuss in claim 35, “the local machine contacts the VARPd that acts as the server for the Supernet to obtain the appropriate address mapping.” EX1003 (Caronni I) at 11:56-59. Here, a POSITA would understand that the administrator node thus provides a virtual network address to a registered device. A POSITA would further understand that only registered devices have addresses in the server VARPd as I discuss in Claim 35 with regard to SNlogout.

434. Thus, a POSITA would understand that the Caronni I and Caronni II registration processes disclose the claimed join limitations that provide virtual network addresses to the devices that are registered. A POSITA would have been motivated to combine Caronni I, Caronni II, and RFC-1383 for the reasons discussed in §XIII.B above.

- e. **Claim 37: The virtual network manager of claim 35 wherein the join module is further configured to maintain data to associate a virtual network address with a device in the virtual network.**

435. See Claim 35 in §XIII.C.1.a above. Furthermore, Caronni I's admin node contains a server VARPDB (join module and part of administrative node) that stores and maintains device information, including addresses in mapping tables. EX1003 (Caronni I) at 7:5–9 (“VARPD 548 has an associated component, VARPDB 551, into which it stores mappings of the internal Supernet addresses, known as a node IDs, to the network addresses recognized by the public-network infrastructure, known as real addresses.”).

436. Thus, a POSITA would understand that the Caronni I and Caronni II registration processes disclose the claimed join limitations that maintain data to associate a virtual network address with a device in the virtual network. A POSITA would have been motivated to combine Caronni I, Caronni II, and RFC-1383 for the reasons discussed in §XIII.B above.

XIV. SECONDARY CONSIDERATIONS

437. I have seen no evidence in either the '785 patent's prosecution history or elsewhere supporting any secondary considerations arguments, or evidence of nexus of such alleged evidence to the challenged claims. To the extent patent owner or its expert asserts the existence of any secondary considerations in its responses, I reserve the right to address any such evidence.

XV. CONCLUSION

438. The findings and opinions set forth in this declaration are based on my work and examinations to date.

439. I may continue my examinations. I may also receive additional documentation and other factual evidence over the course of this IPR that will allow me to supplement and/or refine my opinions. I reserve the right to add to, alter, or delete my opinions and my declaration upon discovery of any additional information. I reserve the right to make such changes as may be deemed necessary.

440. In signing this declaration, I recognize that the declaration will be filed as evidence in an IPR before the PTAB. I also recognize that I may be subject to cross-examination in the case and that cross-examination will take place within the United States. If cross-examination is required of me, I will appear for cross-examination within the United States during the time allotted for cross-examination.

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

Date: 4/10/2025

A handwritten signature in black ink, appearing to read "Erez Zadok", written in a cursive style.

Erez Zadok, Ph.D.