

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

<i>In re</i> Patent of Mohamed Boucadair <i>et al.</i>	§	Attorney Docket No. U022-0010RE
	§	
U.S. Patent 8,451,845	§	
	§	
Issue Date: May 28, 2013	§	Customer No.: 29150
	§	
Filing Date: June 26, 2009	§	
	§	
For: METHOD OF RECEIVING A DATA	§	
PACKET IN AN IPv6 DOMAIN, AN	§	
ASSOCIATED DEVICE AND AN	§	
ASSOCIATED HOME GATEWAY	§	

Mail Stop “*Ex Parte* Reexam”
Attn: Central Reexamination Unit
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

**REQUEST FOR *EX PARTE* REEXAMINATION OF
U.S. PATENT 8,451,845**

Dear Commissioner:

Unified Patents, LLC (“Requester”) requests an *ex parte* reexamination of claims 1 and 5-8 (the “Challenged Claims”) of U.S. Patent 8,451,845 (the “’845 Patent,” Ex. 1001) under 35 U.S.C. §§ 301-307. The ’845 Patent issued on May 28, 2013, to France Telecom, listing Mohamed Boucadair et al. as inventors. The current owner of the ’845 Patent is Monarch Networking Solutions LLC according to an assignment recorded in the USPTO at reel/frame 051238/0718 on December 10, 2019. The ’845 Patent issued from U.S. Patent Application No. 13/001,907 (the “’907 Application”), filed on June 26, 2009. The ’845 Patent received 275 days of patent term adjustment, resulting in an expiration date of March 28, 2030.

Requester submits that this Request presents prior art references and analyses not considered, and not cumulative of, prior art and arguments considered by the Examiner during the original prosecution of the ’845 Patent, and/or presented in a new light or a different way that escaped review during earlier examination. The Challenged Claims are invalid over these references under 35 U.S.C. §§ 102(a), (b), and/or (e) (pre-AIA) or under 35 U.S.C. §§ 103(a) (pre-

AIA). Requester, therefore, requests issuance of an Order for Reexamination and an Office Action rejecting claims 1 and 5-8.

Ex Parte Patent Reexamination Filing Requirements

Section IV of this Request contains statements pointing out substantial new questions of patentability based on material, non-cumulative reference patents and printed publications for the Challenged Claims of the '845 Patent, in compliance with 37 C.F.R. § 1.510(b)(1).

Reexamination of the Challenged Claims of the '845 Patent is requested, and Section V of this Request contains a detailed explanation of the pertinence and manner of applying the cited references to the Challenged Claims, in compliance with 37 C.F.R. § 1.510(b)(2).

Exhibits 1005, 1008, 1009, 1011, and 1020 of this Request contain copies of every patent or printed publication relied upon or referred to in the statement pointing out each substantial new question of patentability (Section IV) or in the detailed explanation of the pertinence and manner of applying the cited references (Section V), in compliance with 37 C.F.R. § 1.510(b)(3).

A copy of the '845 Patent is provided as Exhibit 1001 of this Request, along with a copy of any disclaimer, certificate of correction, and reexamination certificate issued corresponding to the patent, in compliance with 37 C.F.R. § 1.510(b)(4). Additionally, Requester provides an Information Disclosure Statement listing the Exhibits on the Exhibit List below.

The attached Certificate of Service indicates that a copy of this Request, in its entirety including the accompanying Exhibits, has been served on Patent Owner at the following address of record for Patent Owner, in compliance with 37 C.F.R. §§ 1.510(b)(5) and 1.33(c):

NOBLITT & NEWSON, PLLC (Acacia)
Daniel J. Noblitt
8777 NORTH GAINEY CENTER DRIVE, STE.175
Scottsdale, AZ 85258
UNITED STATES

Requester also submits the fee set forth in 37 C.F.R. § 1.20(c)(2).

Requester hereby certifies that the statutory estoppel provisions of 35 U.S.C. § 315(e)(1) and 35 U.S.C. § 325(e)(1) do not prohibit Requester from filing this Request, in compliance with 37 C.F.R. § 1.510(b)(6).

Table of Contents

I.	U.S. Patent 8,451,845	9
A.	Overview.....	9
B.	Prosecution History.....	12
1.	Original Prosecution History	12
2.	<i>Cisco Systems, Inc. v. Monarch Networking Solutions LLC</i> , IPR2020-01679 Challenging the '845 Patent	13
C.	Challenged Claims	13
D.	Level of Ordinary Skill in the Art.....	15
E.	Claim Construction	15
1.	“domain” (Claims 1, 5, 7, and 8).....	15
2.	“regularizing at least one of the IPv6 source address and the IPv6 destination address” (claims 1 and 8).....	16
3.	“Means for” Elements	17
II.	Listing of Prior Art Patents and Printed Publications.....	20
A.	U.S. Patent 7,245,622 to Huitema (“Huitema,” Ex. 1011).....	20
B.	RFC 4380 by C. Huitema (“RFC 4380,” Ex. 1009).....	20
C.	Kong et al., “Design and Implementation of IPv6-IPv4 Protocol Translation System Using Dynamic IP Address” (“Kong,” Ex. 1020).....	21
D.	U.S. Patent Application Publication No. 2004/0107287 to Ananda et al. (“Ananda,” Ex. 1007)	22
E.	U.S. Patent 7,639,686 to Wetterwald et al. (“Wetterwald,” Ex. 1008).....	22
III.	Overview of the Prior Art	22
A.	Analogous Art.....	22
B.	Huitema.....	22
C.	RFC 4380.....	28

D.	Kong.....	29
E.	Ananda	30
F.	Wetterwald.....	33
IV.	Substantial New Questions of Patentability.....	33
A.	Grounds 1 and 2: Substantial New Questions Based on Huitema.....	33
B.	Grounds 3 and 4: Substantial New Questions Based on Kong.....	34
C.	Ground 5: Substantial New Questions Based on Ananda, Wetterwald, and RFC 4380	35
V.	Detailed Application of Challenged Claims to Prior Art.....	36
A.	Grounds 1 and 2: Huitema Anticipates and/or Would Have Rendered Obvious Claims 1 and 5-8.....	36
1.	Independent Claim 1	37
2.	Dependent Claim 5	56
3.	Dependent Claim 6	58
4.	Dependent Claim 7	61
5.	Independent Claim 8	62
B.	Grounds 3 and 4: Kong Anticipates and/or Would Have Rendered Obvious Claims 1 and 5-8	67
1.	Independent Claim 1	67
2.	Dependent Claim 5	80
3.	Dependent Claim 6	84
4.	Dependent Claim 7	84
5.	Independent Claim 8	86
C.	Ground 5: Ananda in View of Wetterwald and RFC 4380 Would Have Rendered Obvious Claims 1 and 5-8	90
1.	Claim 1.....	90
2.	Claim 5.....	118

3. Claim 6..... 122

4. Claim 7..... 122

5. Claim 8..... 123

VI. Secondary Considerations..... 129

VII. Disclosure of Concurrent Litigation, Reexamination, and Related Proceedings..... 129

VIII. Conclusion 129

List of Exhibits

Exhibit	Description
Ex. 1001	U.S. Patent No. 8,451,845 to Mohamed Boucadair et al. (“845 patent”)
Ex. 1002	Prosecution History of U.S. Patent No. 8,451,845
Ex. 1003	Declaration of Dr. Paul Min, Ph. D. from IPR2020-01679
Ex. 1004	<i>Curriculum Vitae</i> of Dr. Min
Ex. 1005	U.S. Patent Application Publication No. 2004/0107287 to Ananda et al. (“Ananda”)
Ex. 1006	Definition of “Domain,” Newton, Harry, “Newton’s Telecom Dictionary,” Sixteenth Expanded Edition, July 2000 (“Newton”)
Ex. 1007	Y. Rekhter et al., “Address Allocation for Private Internets,” Network Working Group, Request for Comments: 1918 (Feb. 1996), available at https://tools.ietf.org/html/rfc1918 (“RFC 1918”)
Ex. 1008	U.S. Patent No. 7,639,686 to Wetterwald et al. (“Wetterwald”)
Ex. 1009	C. Huitema, “Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs),” Internet Engineering Task Force Network Working Group, Request for Comments: 4380 (February 2006), available at https://tools.ietf.org/html/rfc4380 (“RFC4380” or “Teredo”)
Ex. 1010	U.S. Patent Application Publication No. 2005/0286553 to Wetterwald et al. (“553 Publication”)
Ex. 1011	U.S. Patent 7,245,622 to Huitema (“Huitema”)
Ex. 1012	“Internet Protocol – DARPA Internet Program Protocol Specification,” IETF Network Working Group, Request for Comments: 791 (J. Postel ed., Sep. 1981), available at https://tools.ietf.org/html/rfc791 (“RFC 791”)
Ex. 1013	R. Hinden et al., “IP Version 6 Addressing Architecture,” Network Working Group Request for Comments: 1884 (Dec. 1995), available at https://tools.ietf.org/html/rfc1884 (“RFC 1884”)
Ex. 1014	O. Jacobsen et al., “A Glossary of Networking Terms,” Network Working Group, Request for Comments: 1208 (Mar. 1991), available at https://tools.ietf.org/html/rfc1208 (“RFC 1208”)
Ex. 1015	“Internet Users’ Glossary,” Network Working Group Request for Comments: 1983 (G. Malkin ed., Aug. 1996), available at https://tools.ietf.org/html/rfc1983 (“RFC 1983”)
Ex. 1016	U.S. Patent Application Publication No. 2006/0212572 to Afek et al.
Ex. 1017	European Patent No. EP1295454 to Soppera
Ex. 1018	U.S. Patent No. 6,792,546 to Shanklin et al.
Ex. 1019	Monarch Networking Solution LLC’s and Defendant Cisco Systems, Inc.’s Joint Claim Constructions and Prehearing Statement (P.R. 4-3), <i>Monarch Networking Sols. LLC v. Cisco Sys., Inc. et al.</i> , 2:20-cv-00015 (E.D. Tex. filed September 4, 2020)
Ex. 1020	Kong et al., “Design and Implementation of IPv6-IPv4 Protocol Translation System Using Dynamic IP Address” (2003)
Ex. 1021	A. Conta et al., “Generic Packet Tunneling in IPv6 Specification,” Network Working Group Request for Comments: 2473 (Dec. 1998),

	available at https://tools.ietf.org/html/rfc2473 ("RFC 2473")
Ex. 1022	R. Hinden et al., "IP Version 6 Addressing Architecture," Network Working Group Request for Comments: 2373 (July 1998), available at https://tools.ietf.org/html/rfc2373 ("RFC 2373")
Ex. 1023	R. Hinden et al., "Internet Protocol Version 6 (IPv6) Addressing Architecture," Request for Comments: 3513 (Apr. 2003), available at https://tools.ietf.org/html/rfc3513 ("RFC 3513")
Ex. 1024	R. Hinden et al., "IP Version 6 Addressing Architecture," Network Working Group, Request for Comments: 4291 (Feb. 2006), available at https://tools.ietf.org/html/rfc4291 ("RFC 4291")
Ex. 1025	U.S. Patent Application Publication No. 2006/0227792 to Wetterwald et al.
Ex. 1026	Declaration of Dr. Paul Min in Support of Request for <i>Ex Parte</i> Reexamination
Ex. 1027	Declaration of Dr. Ingrid Hsieh-Yee
Ex. 1028	<i>Curriculum Vitae</i> of Dr. Hsieh-Yee
Ex. 1029	Deering, "Internet Protocol, Version 6 (IPv6) Specification," Network Working Group, Request for Comments: 2460 (Dec. 1998), available at https://tools.ietf.org/html/rfc2460 ("RFC 2460")
Ex. 1030-1035	Reserved
Ex. 1036	R. Droms, "Dynamic Host Configuration Protocol," Network Working Group, Request for Comments: 2131 (March 1997), available at https://tools.ietf.org/html/rfc2131 ("RFC 2131")
Exs. 1037-1040	Reserved
Ex. 1041	RFC Editor Webpage, www.rfc-editor.org , as captured by the Internet Archive Wayback Machine on Apr. 30, 2008, available at https://web.archive.org/web/20080430211008/http://www.rfc-editor.org/
Ex. 1042	S. Bradner, "The Internet Standards Process – Revision 3," Request for Comments: 2026 (Oct. 1996), available at https://tools.ietf.org/html/rfc2026 ("RFC 2026").
Ex. 1043	Declaration of Dr. Sylvia Hall-Ellis under 37 C.F.R. § 1.68 in IPR2020-01679
Ex. 1044	<i>Curriculum Vitae</i> of Dr. Hall-Ellis
Exs. 1045-1046	<i>Reserved</i>
Ex. 1047	U.S. Patent No. 7,403,995 to Mace et al.
Ex. 1048	U.S. Patent No. 7,277,453 to Chin et al.
Ex. 1049	U.S. Patent No. 7,764,691 to Sehgal et al.
Ex. 1050	U.S. Patent No. 7,720,976 to Chu et al.
Ex. 1051	U.S. Patent No. 7,729,347 to Rosberg
Ex. 1052	U.S. Patent No. 7,133,405 to Graham et al.
Ex. 1053	U.S. Patent No. 6,917,618 to Thubert et al.
Ex. 1054	Canadian Patent Application Publication No. CA2446579A1
Ex. 1055	PCT Publication No. WO 01/27766
Ex. 1056	U.S. Patent No. 7,995,523 to Chen

Request for *Ex Parte* Reexamination
U.S. Patent 8,451,845

Ex. 1057	U.S. Patent No. 6,957,346 to Kivinen et al.
Ex. 1058	U.S. Patent No. 7,080,151 to Borella et al.
Ex. 1059	U.S. Patent No. 7,095,738 to Desanti
Ex. 1060	U.S. Patent No. 8,045,558 to Droms et al.
Ex. 1061	U.S. Patent No. 7,729,312 to Suzuki et al.
Ex. 1062	U.S. Patent No. 7,961,725 to Nagarajan et al.
Ex. 1063	U.S. Patent No. 7,813,843 to Patel et al.
Ex. 1064	U.S. Patent No. 8,688,834 to Fujimoto et al.
Ex. 1065	U.S. Patent No. 7,568,040 to Townsley et al.
Ex. 1066	U.S. Patent No. 7,738,882 to Dutta et al.
Ex. 1067	U.S. Patent No. 7,543,065 to Barkley et al.
Ex. 1068	U.S. Patent No. 7,640,338 to Huitema et al.
Ex. 1069	U.S. Patent No. 7,502,474 to Kaniz et al.
Ex. 1070	U.S. Patent No. 8,245,304 to Chen et al.
Ex. 1071	U.S. Patent No. 7,804,848 to Mirtorabi et al.
Ex. 1072	Definition of “Concatenate,” Alan Freedman, “Computer Desktop Encyclopedia,” 9 th Edition, McGraw-Hill (2001)
Ex. 1073	Complaint for Patent Infringement in <i>Monarch Networking Solutions LLC v. Juniper Networks, Inc.</i> , Case No. 1:23-cv-670 (E.D. Va. May 23, 2023)

I. U.S. Patent 8,451,845

A. Overview

The '845 patent is directed to an Internet Protocol (IP) telecommunications network for transmitting data packets from a source terminal identified by a source address to a destination terminal identified by a destination address. Ex. 1001, 1:17-21. The '845 patent explains that a “home gateway” interconnects a home network (e.g., a private network such as a home network or a business network) and a public network. *Id.*, 1:40-48, 4:46-47.

The conventional version of the IP address system, known as IPv4 networking,¹ was widely used. It was commonly accepted that public IPv4 addresses would run out. Ex. 1001, 1:61-62. The Internet Engineering Task Force (IETF) then developed a new IP addressing protocol referred to as IPv6.² Ex. 1001, 1:61-2:6.

IPv6 allows for 128-bit addresses, which are substantially longer than IPv4's 32-bit addresses, and therefore provides a greater number of possible addresses than IPv4. Ex. 1001, 7:26-29; Ex. 1012, 7 (IPv4); Ex. 1013, 3 (IPv6). Example IPv4 and IPv6 addresses are shown below. Ex. 1013, § 2.2, 4-5.

Example Addresses	
IPv4 Address	192.159.252.76
IPv6 Address	3FFE:F200:0234:AB00:0123:4567:8901:ABCD

IPv4 and IPv6 addresses have two parts: (1) a prefix identifying the components of a subnetwork (“subnet”) through which a data packet is to be routed to reach its destination (e.g., router, gateway, etc.); and (2) an address of a machine connected to the subnetwork (e.g., a gateway, or the destination). Ex. 1001, 7:30- 34; Ex. 1013, 3-4 (describing a subnet prefix). In the

¹ The IPv4 protocol was standardized by the IETF in RFC791 in September 1981. Ex. 1012; *see* Ex. 1043, ¶¶ 49-52 (citing Exs.1051-1052).

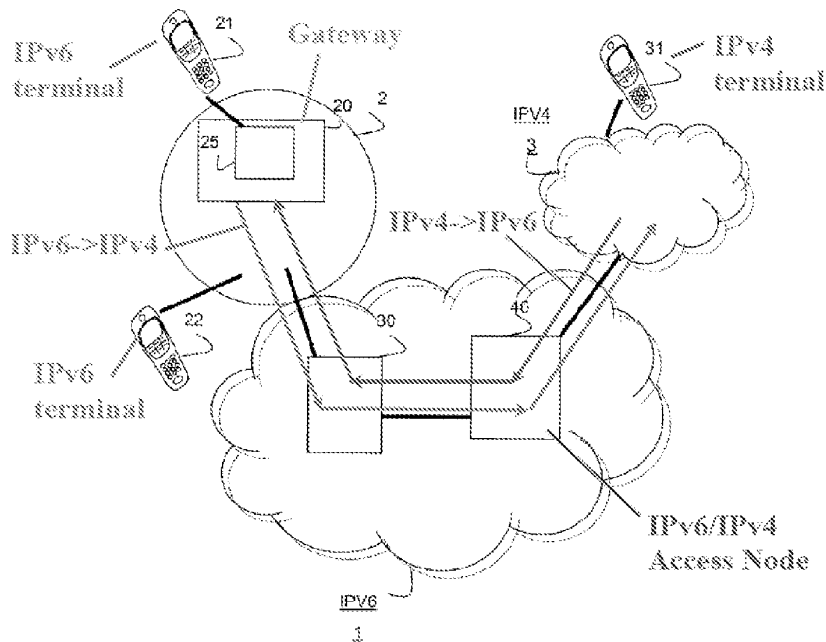
² The IPv6 protocol was standardized by the IETF in RFC1884 in December 1995 and revised in later years. Ex. 1013; *see* Ex. 1043, ¶¶ 53-56 (citing Exs.1048, 1053); *see also* Ex. 1022-1024; Ex. 1043, ¶¶ 69-80 (citing Exs.1058-1063).

above example IPv6 address, the first 64 bits may be an **IPv6 prefix**, and the last 64 bits may be a **device address**, as shown below.

3FFE:F200:0234:AB00:0123:4567:8901:ABCD

The '845 patent explains that if an IPv6-configured device in an IPv6 domain transmits an IPv6 packet to an IPv4-configured terminal in an IPv4 domain, the IPv6 packet must be translated to an IPv4 packet, and vice versa. Ex. 1001, 3:18-48.

To facilitate translation from IPv4 to IPv6 addresses, the '845 patent discloses that an IPv6 address is constructed by concatenating an IPv6 prefix (operator prefix), an IPv4 address, and a port number (e.g., port number of a gateway, router or access node connected to the destination terminal). Ex. 1001, 3:7-9, 3:30-48, 7:18-20. As shown by the **IPv6-to-IPv4** translation in annotated Figure 1 below, the constructed IPv6 packet allows for the IPv6 packet from an **IPv6 terminal** packet to be transmitted across an IPv6 network via a **gateway** to an **IPv6/IPv4 access node**, which translates the IPv6 packet into an IPv4 packet and forwards the IPv4 packet to an **IPv4 destination terminal**. Ex. 1001, 3:34-48, 11:45-61.



Ex. 1003, ¶ 38; '845 Patent, FIG. 1*³

Conversely, the '845 patent discloses that an IPv4 packet from an **IPv4 terminal** is translated into an IPv6 packet⁴ by the **IPv6/IPv4 access node** and transmitted across the IPv6 network 1 to an IPv6 terminal via the **gateway**. Ex. 1001, 11:38-61. This represents an **IPv4-to-IPv6** translation, as shown in annotated Figure 1 of the '845 Patent above.

The '845 patent explains that source and destination IPv6 addresses of an IPv6 packet can be either a native IPv6 address or a constructed IPv6 address, i.e., an address that is constructed as described above, and the gateway can replace the native IPv6 address or constructed IPv6 address in translating from IPv6 to IPv4. Ex. 1001, 3:10-15, 11:6-12.

³ Requester uses “*” to indicate annotated Figures.

⁴ This operation is known as “4in6” tunneling or “IPv4-in-IPv6” tunneling. *See* RFC2473 (Ex. 1021, 2-7); *see also* Ex. 1043, ¶¶ 65-68 (citing Exs.1056-1057).

B. Prosecution History

1. Original Prosecution History

The '845 patent was filed as U.S. Application No. 13/501,907 (“’907 Application”) as a U.S. National Stage Application of PCT/FR2009/051228, which claims priority to French Patent Application No. 0854405, filed on June 30, 2008. Ex. 1001, 1. Requester does not concede that the '845 Patent claims are entitled to a priority date of June 30, 2008, but the references in Grounds 1-5 qualify as prior art regardless.

Only one Office Action issued during prosecution of the '845 patent.⁵ Ex. 1002, 55-64. The Examiner did not reject the claims based on prior art. *Id.* The applicant presented clarifying amendments to overcome enablement and indefiniteness rejections relating to the construction of the IPv6 destination address, and a clarity objection relating to the “regularizing” element. Ex. 1002, 43-52.

The Examiner then allowed the application. Ex. 1002, 29-36. The Examiner identified elements [1.1]/[8.1], [1.2.1]/[8.2.1], [1.2.2]/[8.2.2], and [1.2.3]/[8.2.3] as the allegedly allowable features of claims 1 and 8 but without discussing these features in relation to any prior art of record. Ex. 1002, 35. As discussed in Section V below, these features are rendered obvious by at least: Grounds 1 and 2—Huitema (Ex. 1011); Grounds 3 and 4—Kong (Ex. 1020); and Ground 5—Ananda (Ex. 1005), Wetterwald (Ex. 1008), and RFC 4380 (Ex. 1009).

There is no record in the prosecution history that the Examiner considered Kong, Ananda, Wetterwald, or RFC 4380 in relation to the challenged claims. And although the Examiner cited Huitema on a Form PTO-892 accompanying the Office Action, neither the Examiner nor the applicant commented on Huitema on the record. Ex. 1002, 60. Given the strong merits of Grounds 1 and 2 described below, the Examiner did not fully appreciate the relevance of Huitema to the challenged claims.

⁵ In the Office Action, the Examiner cited U.S. 2005/0286553 to Wetterwald et al. (Ex. 1010), which is a different reference, with a different disclosure, than Wetterwald (Ex. 1008) cited in Ground 4 of unpatentability. Ex. 1002, 60 (“Notice of References Cited”).

2. *Cisco Systems, Inc. v. Monarch Networking Solutions LLC*, IPR2020-01679 Challenging the '845 Patent

On October 1, 2020, Cisco filed an *inter partes* review (IPR) petition challenging claims 1 and 5-8 of the '845 Patent in IPR2022-01679. Ex. 1038. The petition only raised one ground—obviousness over Ananda, Wetterwald, and RFC 4380—which Requester includes here as Ground 5. However, the parties settled the IPR before Patent Owner's preliminary response was due and before the PTAB rendered an institution decision on the merits of the petition. Ex. 1039. Thus, the U.S. Patent and Trademark Office never fully considered the substantial new question(s) of patentability raised by Cisco in its petition (Ground 5 of this Request).

C. Challenged Claims

As described above, Requester challenges claims 1 and 5-8. Requester reproduces all of the Challenged Claims below, with claim element reference numbers in brackets used in the Detailed Application of Challenged Claims to Prior Art Section:

Challenged Claims

1. [1.pre.1] A method for receiving an IPv6 data packet in an IPv6 domain connected to an IPv4 domain, [1.pre.2] said packet comprising an IPv6 destination address and an IPv6 source address, [1.pre.3] said method being executed in a home gateway adapted to connect a user terminal to said IPv6 domain, and comprising the following steps:

[1.1] identifying an IPv6 destination address constructed by concatenating an IPv6 prefix, an IPv4 destination address and a destination port number;

[1.2.1] if necessary, regularizing at least one of the IPv6 source address and the IPv6 destination address of the data packet by replacing said at least one IPv6 address, [1.2.2] the replacement process belonging to a group comprising replacement of a constructed address with a native address and replacement of a native address with a constructed address, and [1.2.3] modifying the data packet using the regularized address; and

[1.3] routing the modified data packet to its destination.

5. [5.pre] The method according to claim 1, wherein when the data packet comes from the IPv4 domain and goes to said user terminal,

the step of regularizing the destination address comprises the following steps:

[5.1] searching an address translation table of the gateway for an entry associating with said constructed destination address a native destination address;

[5.2] if an entry is found, replacing the constructed destination address of the data packet with the native destination address from that entry.

6. [6.1] The method according to claim 5, further comprising a step of verifying that the destination port number of the data packet and the destination port number contained in the constructed destination address match.

7. [7.1] A non-transitory computer program product that is at least one of downloadable from a communications network and stored on a computer-readable medium and executable by a microprocessor, comprising program code instructions for executing the method according to claim 1 of receiving a data packet in an IPv6 domain.

8. [8.pre] A device for receiving an IPv6 data packet in an IPv6 domain connected to an IPv4 domain, [8.pre.1] said packet comprising an IPv6 destination address and an IPv6 source address, [8.pre.2] said device for use in a home gateway adapted to connect a user terminal to said IPv6 domain, comprising:

[8.1] means for identifying an IPv6 destination address constructed by concatenating an IPv6 prefix, an IPv4 destination address, and a destination port number;

[8.2] means for [8.2.1] regularizing, if necessary, at least one of the IPv6 source address and the IPv6 destination address of the data packet by replacing said at least one IPv6 address, [8.2.2] the replacement process belonging to a group comprising replacement of a constructed address with a native address and replacement of a native address with a constructed address, and [8.2.3] modifying the data packet using the regularized address; and

[8.3] means for routing the modified data packet to its destination.

D. Level of Ordinary Skill in the Art

A person of ordinary skill in the art (“POSITA”) in June 2008 would have had a working knowledge of the packet switching art that is pertinent to the ’845 patent, including computer networking, IPv4, IPv6, packet encapsulation, and packet routing. A POSITA would have had a bachelor’s degree in electrical engineering, computer engineering, computer science, or a related subject, and three years of work experience in computer networking, including experience with the IPv4 and IPv6 protocols, or a related field. Lack of professional experience can be remedied by additional education, and vice versa. Ex. 1003, ¶¶ 21-22.

E. Claim Construction

The ’845 Patent has not yet expired, so the Challenged Claims should be given their “broadest reasonable interpretation consistent with the specification.” 37 CFR §1.75(d)(1); *see also* MPEP § 2111; *Phillips v. AWH Corp.*, 415 F.3d 1303, 1316 (Fed. Cir. 2005). Under this standard, the Office “determines the scope of claims in patent applications not solely on the basis of the claim language, but upon giving claims their broadest reasonable construction in light of the specification as it would be interpreted by one of ordinary skill in the art.” *Phillips*, 415 F.3d 1303, 1316 (Fed. Cir. 2005) (quotation and citation omitted). “Because applicant has the opportunity to amend the claims during prosecution, giving a claim its broadest reasonable interpretation will reduce the possibility that the claim, once issued, will be interpreted more broadly than is justified.” MPEP § 2111 (citing *In re Yamamoto*, 740 F.2d 1569, 1571 (Fed. Cir. 1984)).

1. “domain” (Claims 1, 5, 7, and 8)

The term “domain” does not require construction. But to the extent that Patent Owner argues, or the Examiner determines, that construction is required, Requester submits that an appropriate construction is “network.” Ex. 1003, ¶¶ 54-55. However, Requester’s detailed claim analysis below shows how the claims are unpatentable under multiple constructions that Patent Owner might advance.

The ’845 patent states that “[a] telecommunications network administered by an operator is also referred to as a domain.” Ex. 1001, 1:30-31. The ’845 patent repeatedly suggests that “domain” is essentially synonymous with “network.” For example, the ’845 patent explains that a “home gateway [is provided] between a home network and the public network or IP domain of the

operator,” and that the home gateway “generally assigns private IP addresses to the terminals of its home network.” Ex. 1001, 1:40-44; Ex. 1003, ¶ 56.

Newton’s Telecom Dictionary defines “*domain*” as “the part of a computer network in which the data processing resources are under common control,” as “[s]ome people think of a domain as a group of computers on a network that share something in common, such as the job they do, the organization type they belong to, or their location.” Ex. 1006, 196. The “Glossary of Networking Terms” in RFC 1208 defines “*domain*” as “an administrative partition of a complex distributed system.” Ex. 1014, 6. Relatedly, RFC 1208 defines a “subnetwork” as “[a] collection of OSI [Open Systems Interconnection] end systems and intermediate systems under the control of a single administrative domain and utilizing a single network access protocol.” Ex. 1014, 16. The “Internet Users’ Glossary” in RFC 1983 notes that “*domain*” is “a heavily overused term,” indicating that it may take on different meanings depending on context, including referring to “collection of hosts and routers, and the interconnecting network(s), managed by a single administrative authority” or to the “routers exchanging routing information within” such a collection. Ex. 1015, 17, 4, 48; Ex. 1003, ¶ 57; *see also* Ex. 1043, ¶¶ 61-64 (citing Exs. 1054-1055).

Particularly under the BRI standard, the term “domain” should not be construed to require “common control” or a “single administrative authority” because the claims themselves identify the point of commonality for each “*domain*” by reciting “*IPv4 domain*” or “*IPv6 domain*.” Thus, terminals within an “*IPv4 domain*” communicate using the IPv4 protocol, while terminals within an “*IPv6 domain*” communicate using the IPv6 protocol. Ex. 1003, ¶ 58.

In the event Patent Owner may argue otherwise, this Request shows that features such as “common control” would have been obvious. Ex. 1003, ¶ 59.

In summary, no construction is believed necessary for “*domain*,” but to the extent a construction is adopted, the Office should construe “*domain*” to mean a “network.”

2. “regularizing at least one of the IPv6 source address and the IPv6 destination address” (claims 1 and 8)

By using the phrase “at least one of,” elements [1.2.1] and [8.2.1] require either (1) regularizing the IPv6 source address by replacing the IPv6 source address, or (2) regularizing the IPv6 destination address by replacing the IPv6 destination address. Ex. 1003, ¶ 60.

This interpretation is consistent with the applicant's claim amendments and remarks during prosecution. In response to a clarity objection, the applicant amended elements [1.2.1] and [8.2.1] to recite "regularizing at least one *of the IPv6 source address and the IPv6 destination* address of the data packet by replacing said *at least one IPv6* address." Ex. 1002, 46, 47.⁶ The applicant explained that "[t]he words 'at least' make clear that it can be either the IPv6 destination address or the IPv6 source address. The claims have been amended to spell that out explicitly." Ex. 1002, 49; Ex. 1003, ¶ 61.

3. "Means for" Elements

Claim 8 recites several "means for" elements. Requester submits that the following "means for" terms should be interpreted as means-plus-function elements under 35 U.S.C. § 112 ¶ 6.⁷ Requester identifies the function and structure for each of the "means" terms below, in compliance with 37 C.F.R. § 42.104(b)(3).

Claim 8 recites that the claimed "device [is] for use in a home gateway adapted to connect a user terminal to said IPv6 domain." Ex. 1001, 19:20-21. With reference to annotated Figure 9 below, the '845 patent describes "a device 25 [] for receiving a data packet in an IPv6 domain." Ex. 1001, 16:56-58. The '845 patent explains that the device 25 includes "hardware elements conventionally found in a standard computer or a computer router," including a **processor 251**, a "**telecommunications means 254** for communicating with the network 1," and a **read-only memory 255**. Ex. 1001, 16:55-17:7; Ex. 1003; ¶ 64.

⁶ Emphasis added unless otherwise specified.

⁷ In a related district court litigation, Patent Owner argued that the "means" terms identified in this Section should be construed under 35 U.S.C. § 112 ¶6. Ex. 1019, 14-16, 19-20.

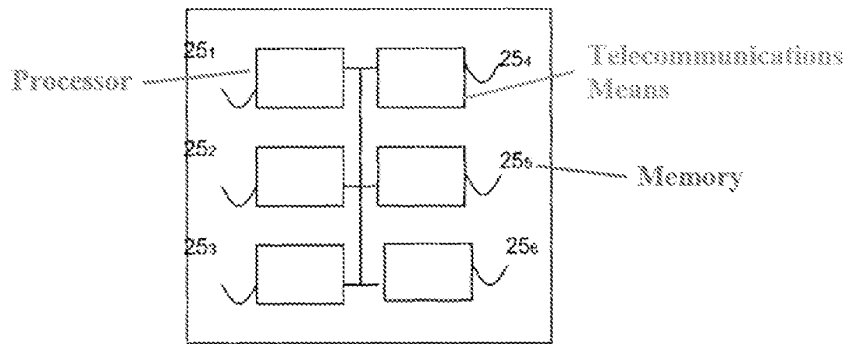


FIG. 9

Ex. 1003, ¶ 64; Ex. 1001, FIG. 9*

- a. *“means for identifying an IPv6 destination address constructed by concatenating an IPv6 prefix, an IPv4 destination address, and a destination port number;” (Claim 8)*

Function: identifying an IPv6 destination address constructed by concatenating an IPv6 prefix, an IPv4 destination address, and a destination port number.

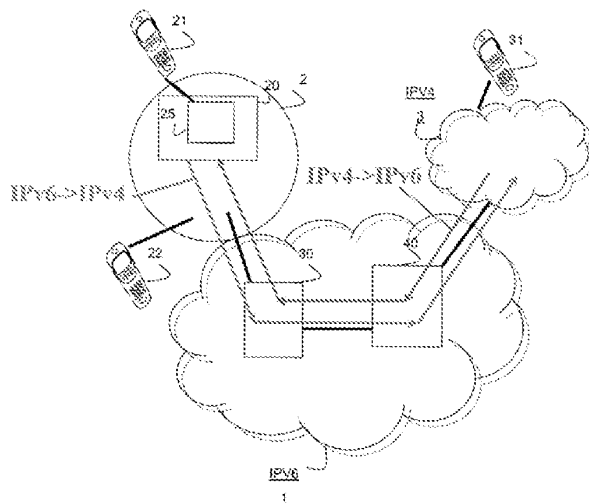
Structure: a receiver device (e.g., device 25) of the home gateway 20 having a processor 25₁ and memory 25₅, and equivalents. Ex. 1001, 10:61-11:4, 16:55-17:7; Ex. 1003, ¶ 67.

Explanation: The '845 patent explains that the disclosed method of receiving an IPv6 data packet is “executed by a receiver device included in a home gateway connected to the IPv6 domain, such as the gateway 20,” and includes a step R1 (Fig. 3) of identifying an IPv6 destination address. Ex. 1001, 10:61-11:4. The '845 patent explains that the receiver device of the home gateway 20 determines whether the IPv6 destination address is a constructed IPv6 address based on whether the IPv6 destination address contains a certain IPv6 prefix. Ex. 1001, 13:11-16, 10:10-22, 12:40-43, 13:10-14; Ex. 1003, ¶¶ 68-69.

- b. *“means for regularizing, if necessary, at least one of the IPv6 source address and the IPv6 destination address of the data packet*

Function: (1) regularizing the IPv6 source address by replacing the IPv6 source address, or (2) regularizing the IPv6 destination address by replacing the IPv6 destination address. Further, the “means for regularizing” includes the function of modifying the data packet using the regularized address.

Explanation: With reference to annotated Figure 1 below, the '845 patent discloses that when a packet is transmitted from an IPv4 domain 3 to the home gateway 20 in the IPv6 domain (IPv4->IPv6 in annotated Figure 1), the IPv6 source address of the packet is regularized (i.e., replaced) using an address translation table, whereas when a packet is transmitted from an IPv6 terminal in the IPv6 domain 1 via the gateway 20 to the IPv4 network (IPv6->IPv4 in annotated Figure 1), the IPv6 destination address is regularized (i.e., replaced) using an address translation table. Ex. 1001, 3:10-15, 11:7-11, 16:18-44, 16:55-17:7; Ex. 1003, ¶¶ 75-76. Ex. 1003, ¶¶ 75-76;



'845 Patent, FIG. 1*

c. ***“means for routing the IPv6 data packet to its destination” (Claim 8)***

Function: routing the IPv6 data packet to its destination.

Structure: telecommunications means 25₄ of the device 25 of the home gateway 20, and equivalents. Ex. 1001, 16:58-63; Ex. 1003, ¶ 78.

Explanation: The '845 patent explains that the device 25 of the home gateway 20 includes “the hardware elements conventionally found in a standard computer or a dedicated router, namely a... *telecommunications means 25₄* for communicating with the [IPv6] network 1.” Ex. 1001, 16:58-63; Ex. 1003, ¶¶ 79-80.

II. Listing of Prior Art Patents and Printed Publications

Requester requests reexamination of the Challenged Claims in view of the following prior art patents and printed publications. Requester has also filed concurrently an Information Disclosure Statement listing these prior art references and the other Exhibits on the Exhibit List.

Requester submits a Declaration of Dr. Hall-Ellis (Ex. 1043) from IPR2020-01679 regarding the public availability of the RFC references cited in this request, including RFC 4380 (Ex. 1009), before the claimed priority date of the '845 Patent. Requester also submits a declaration of Dr. Hsieh-Yee (Ex. 1027) regarding the public availability of Kong (Ex. 1020) before the claimed priority date of the '845 Patent.

A. U.S. Patent 7,245,622 to Huitema (“Huitema,” Ex. 1011)

Huitema is a U.S. Patent entitled “Allowing IPv4 clients to communicate over an IPv6 network when behind a network address translator with reduced server workload,” filed on March 27, 2003, and issued on July 17, 2007. Ex. 1011, 1. The '845 Patent was filed on June 26, 2009, Ex. 1001, 1, so Huitema qualifies as prior art under at least 35 U.S.C. §§ 102(a), (b).

B. RFC 4380 by C. Huitema (“RFC 4380,” Ex. 1009)

RFC 4380 is an Internet Engineering Task Force (IETF) publications by C. Huitema, “Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs),” Internet Engineering Task Force Network Working Group, Request for Comments: 4380 (February 2006), available at <https://tools.ietf.org/html/RFC4380> (“RFC 4380”). Ex. 1009, 1.

Dr. Min's Declaration (Ex. 1003), Dr. Min's Supplemental Declaration (Ex. 1026), and this Request cite RFC 4380, as well as additional documents published by the Internet Engineering Task Force (IETF) publications, as further evidence for various facts. These documents qualify as prior art "printed publications" for the purposes for which they are cited. Ex.1003, ¶ 24 (citing Exs.1007, 1012-1015, 1021-1024, 1036, 1042); Ex.1043, ¶¶ 41-44 (citing Exs.1047-1048), 49-88 (citing Exs.1051-1067). The '845 patent itself acknowledges that IETF documents—including draft documents—were accessible, available, and known to the public. *See, e.g.*, Ex.1001, 7:46-49 (citing RFC 2462), 7:53-58 (citing RFC 3315), 7:67-8:5 (citing RFC 3633), 9:58-61 (citing RFC 3633), Code (56), References Cited (citing RFC2663). The PTAB has repeatedly found that IETF documents, including RFCs, are "printed publications." *See, e.g.*, IPR2017-00337, Paper 31 at 46-47 (Requests for Comments are "precisely the type of documents whose main purpose is for public disclosure"); IPR2018-00130, Paper 11 at 30-33 (finding multiple RFCs to be printed publications).

RFC 4380 qualifies as a prior art "printed publication" to the '845 Patent. Ex. 1043, ¶¶ 45-48 (public accessibility of RFC 4380) (citing Exs.1049-1050); Ex. 1003, ¶ 24 (citing Exs.1007, 1012-1015, 1021-1024, 1036, 1042); Ex. 1042, 6, 8, 26. For example, Dr. Hall-Ellis testifies that RFC 4380 was publicly available on the IETF.com website by at least February 28, 2006. Ex. 1043, ¶ 46. Additionally, RFC 4380 was archived by the Wayback Machine many times from 2000-2008. Ex. 1043, ¶ 46; *see also* MPEP § 2128.II.E ("Prior art obtained via the Wayback Machine® sets forth a *prima facie* case that the art was publicly accessible at the date and time provided in the time stamp.") (citation omitted). Accordingly, RFC 4380 "has been disseminated or otherwise made available to the extent that persons interested and ordinarily skilled in the subject matter or art, exercising reasonable diligence, can locate it." MPEP § 2128.III (citation omitted). Thus, RFC 4380 qualifies as prior art under at least 35 U.S.C. §§ 102(a), (b).

C. Kong et al., "Design and Implementation of IPv6-IPv4 Protocol Translation System Using Dynamic IP Address" ("Kong," Ex. 1020)

Kong is a 2003 article by In-Yeup Kong et al. entitled, "Design and Implementation of IPv6-IPv4 Protocol Translation System Using Dynamic IP Address." Ex. 1020, 1. As explained by Dr. Hsieh-Yee, Kong was publicly accessible at the Library of Congress since at least May 2003 and at the Stanford University Library since September 2003. Ex. 1027, ¶¶ 26-45. Thus, Kong qualifies as prior art under at least 35 U.S.C. §§ 102(a), (b).

D. U.S. Patent Application Publication No. 2004/0107287 to Ananda et al. (“Ananda,” Ex. 1007)

Ananda is U.S. Patent Application Publication No. 2004/0107287 to Ananda et al., entitled “Method and apparatus for communicating on a communication network” and published on June 3, 2004. Ananda is prior art under 35 U.S.C. §§ 102(a), (b), and (e).

E. U.S. Patent 7,639,686 to Wetterwald et al. (“Wetterwald,” Ex. 1008)

Wetterwald is a U.S. Patent entitled, “Access network clusterhead for providing local mobility management of a roaming IPv4 node,” filed on April 7, 2005, and issued on December 29, 2009. Ex. 1008, 1. Wetterwald is prior art under 35 U.S.C. § 102 (e). Wetterwald was also the subject of a pre-grant publication, U.S. Pub. No. 2006/0227792 on October 12, 2006, and is therefore prior art under 35 U.S.C. § 102 (a) and (b). *See* Ex. 1025.

III. Overview of the Prior Art

The references discussed below relate to providing interoperability and communication between IPv4 nodes and IPv6 nodes across an IPv6 network. To do so, each reference discloses a manner of constructing an IPv6 address based on IPv4 address information by combining or concatenating an IPv6 prefix with IPv4 address and/or port information in the manner claimed.

A. Analogous Art

All the references identified below relate to analogous art. Ex. 1026, ¶ 31. “A reference is analogous art to the claimed invention if: (1) the reference is from the same field of endeavor as the claimed invention (even if it addresses a different problem); or (2) the reference is reasonably pertinent to the problem faced by the inventor (even if it is not in the same field of endeavor as the claimed invention).” MPEP § 2141.01(a) (*citing In re Bigio*, 381 F.3d 1320, 1325 (Fed. Cir. 2004)). Both are true here: the challenged ’845 Patent claims—as well as Huitema, RFC 4380, Kong, Ananda, and Wetterwald—all relate to the field of IP networking and seek to solve the same problem of how to communicate between IPv4 and IPv6 networks. Ex. 1026, ¶ 26.

B. Huitema

Huitema describes embodiments of methods in which an IPv4 client in an IPv4 network and an IPv6 client in an IPv6 network establish communication, including the methods shown in

Figure 2, Figures 3A-3B, and Figure 4.⁸ Ex. 1011, 5:54-6:2; Ex. 1026, ¶ 32. As seen in Figure 3A of Huitema reproduced below, IPv4 client 101 resides in a private network of an IPv4 network, IPv6 client 202 resides in an IPv6 network, and relay 202 and server 103 connect the IPv6 network to the IPv4 network. *Id.*, 3:3-5:25, 5:57-65, 6:15-18, 11:62-14:52, FIG. 3A.

⁸ While Requester focuses on the embodiment of Figures 3A-3B, the three embodiments overlap of Figures 2, 3A-3B, and 4 overlap substantially and independently disclose or suggest each element of '845 Patent claims 1 and 5-8. Ex. 1026, ¶ 32, n. 1.

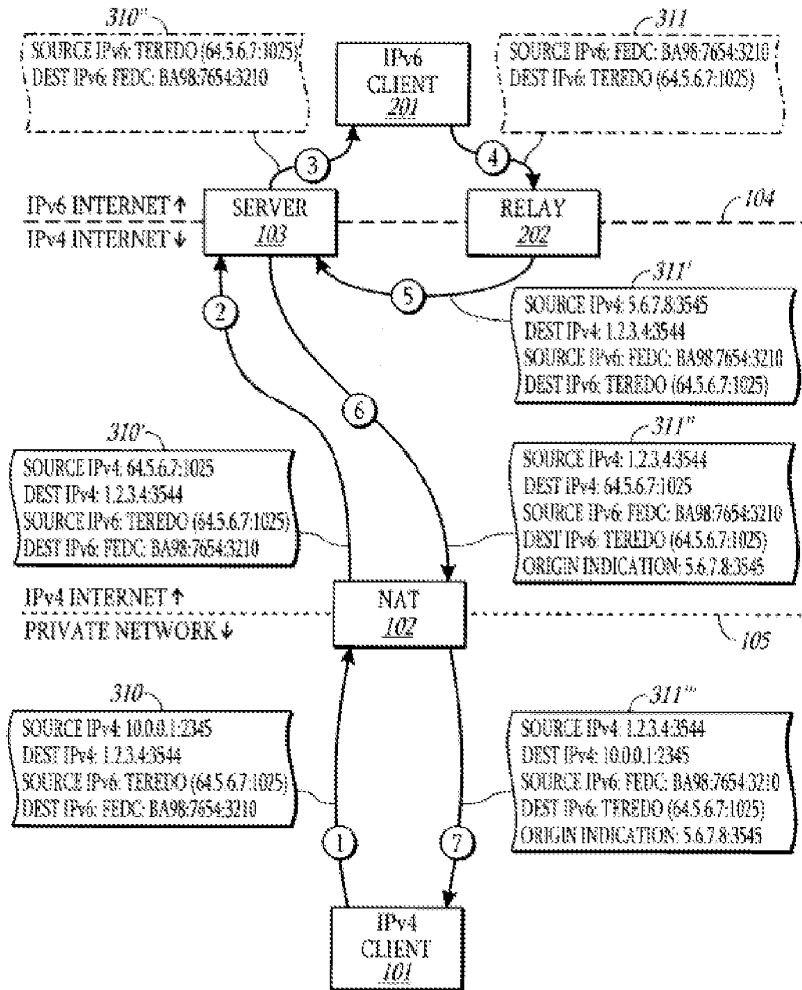


FIG. 3A

In Figure 3A, “IPv6 Internet illustrated as that portion that is above the dashed line 104, [the] IPv4 Internet illustrated as that portion between the dashed line 104 and the dotted line 105, and a private network illustrated as that portion below the dotted line 105.” Ex. 1011, 7:4-9.

Because it resides on IPv4 Internet, IPv4 client 101 technically has no IPv6 address. Ex. 1026, ¶ 33. Rather, “client 101 has the private IPv4 network address 10.0.0.1” and “the globally unique [public] IPv4 network address 64.5.6.7.” Ex. 1011, 7:51-55. Likewise, “IPv6 client 201 presumably has no IPv4 address since it is not in the IPv4 Internet,” but has “IPv6 address ‘FEDC:BA98:7654:3210.’” Ex. 1011, 10:26-40. “Even though [IPv4] client 101 has this IPv4 address, the client now desires an IPv6 network address” to communicate with IPv6 client 201.

Ex. 1011, 7:56-57; Ex. 1026, ¶ 33.

Figure 1 shows a method by which IPv4 client 101 first obtains information to construct its own IPv6 address for later communication with IPv6 client 201 in Figures 2, 3A-3B, and 4. Ex. 1011, 7:42-8:60; Ex. 1026, ¶ 34. First, IPv4 client 101 sends IPv4 message 110 containing “a source address of 10.0.0.1 (i.e., the private network address of the client 101) and a source UDP port (e.g., 2345) and a destination address of 1.2.3.4 (i.e., the

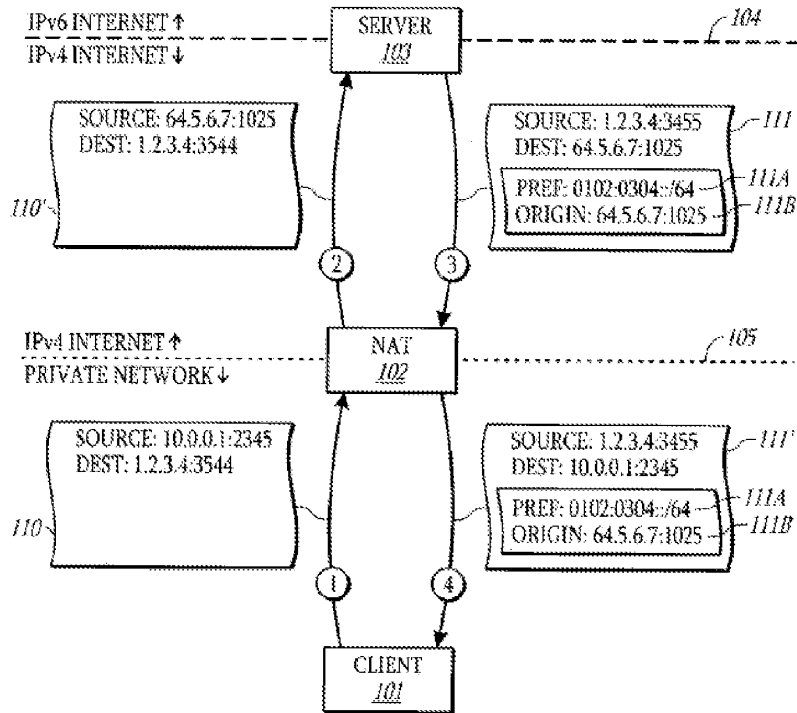


FIG. 1

public network address of the server 103) and destination port (e.g. 3544).” Ex. 1011, 7:58-63. Server 103 receives message 110’ from NAT 102, notes the source address and port (e.g., 64.5.6.7:1025), and sends back a response [111] to that address and port” containing information that IPv4 client 101 can use to construct its own IPv6 address to communicate with IPv6 client 201. Ex. 1011, 8:8-17; Ex. 1026, ¶ 34.

As seen in Figure 1 above, that “response [111] includes a prefix assigned to the server 10[3], as well as an origin indication,” for example, “prefix 111A of ‘PREF:0102:0304::/64’ and an origin indication 111B of ‘64.5.6.7:1025’.” Ex. 1011, 8:12-17; *see also id.*, 8:17-25, FIG. 5. Now that IPv4 client 101 has IPv6 prefix 111A and origin indication 111B, “client 101 then constructs its own IPv6 address, which in this example may be ‘PREF:102:304::FBFE:BFFA:F9F8’.” Ex. 1011, 8:35-27.

Reproduced below, Figure 7 shows data structure 700 of client 101’s constructed IPv6

address. *Id.*, 8:35-60; Ex. 1026, ¶ 36.



FIG. 7

Huitema, FIG. 7

Data structure 700, constructed by IPv4 client 101, contains five concatenated pieces of information:

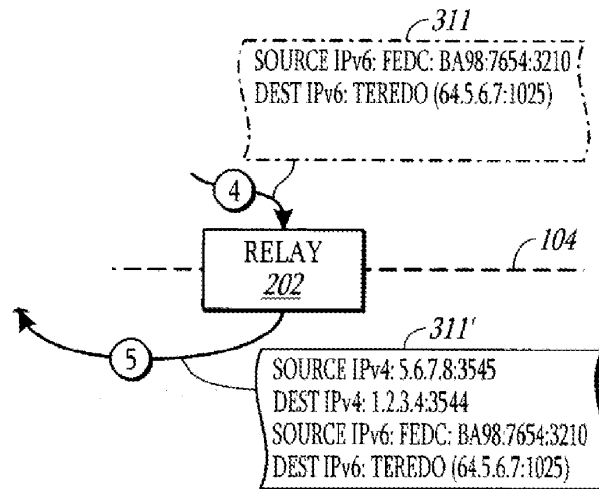
- **IPv6 service field 701 (TEREDO):** Prefix field 701 contains a prefix “identif[ying] that this IPv6 address corresponds to an IPv4 client.” Ex. 1011, 8:39-40; Ex. 1026, ¶ 36 “For example, in the IPv6 address ‘PREF:102:304::FBFE:BFFA:F9F8’,” service field 701 contains the prefix “PREF”. Ex. 1011, 8:38-42; Ex. 1026, ¶ 36.
- **Server IPv4 address field 702:** This “field 702 identifies the IPv4 address of the server 103. For example, the ‘102:304’ in the example IPv6 address is the hexadecimal notation of the address of the server 103 (namely, 1.2.3.4).” Ex. 1011, 8:42-46; Ex. 1026, ¶ 36.
- **NAT flag field 703:** This “field 703 may identify whether or not the NAT 102 is a cone NAT.” Ex. 1011, 8:46-47; Ex. 1026, ¶ 36.
- **Client port field 704:** “[C]lient port field 704 identifies the mapped port (e.g., UDP port) of the client 101. For example, the ‘FBFE’ in the example IPv6 address is the mapped port 1025 exclusive OR’ed with hexadecimal number ‘FFFF’.” Ex. 1011, 8:47-51; Ex. 1026, ¶ 36.
- **Client IPv4 address field 705.** “[C]lient IPv4 address field 705 identifies the IPv4 address of the client 101. For example, the ‘BFFA:F9F8’ in the example IPv4 address is the mapped address 64.5.6.7 exclusive OR’ed with hexadecimal number ‘FFFF:FFFF’.” Ex. 1011, 8:51-55; Ex. 1026, ¶ 36.

Once IPv4 client 101 constructs its own IPv6 address as described above, “client 101 is now properly provisioned to communicate with IPv6 computing systems according to the methods”

described in Figures 2, 3A-3B, and 4. Ex. 1011, 8:55-60; Ex. 1026, ¶ 36.

Returning to Figure 3A, IPv4 client 101 now sends an IPv4 message 310 to IPv6 client 201. Ex. 1011, 11:66-12:12; Ex. 1026, ¶ 37. In its payload, message 310 contains IPv4 client 101's constructed IPv6 address (represented as TEREDO (64.5.6.7:1025)) as the IPv6 source address and the destination IPv6 address for IPv6 client 201 (FEDC:BA98:7654:3210). Ex. 1011, 11:66-12:12; Ex. 1026, ¶ 37.

When IPv6 client 201 receives message 310, it sends IPv6 response 311, shown in the excerpt of Figure 3A on the right. Ex. 1011, 12:33-41; Ex. 1026, ¶ 38. Response 311 contains a "Source IPv6 Address: FEDC:BA98:7654:3210" for IPv6 client 201 and a "Destination IPv6 Address: "TEREDO (64.5.6.7:1025)" for IPv4 client 101. Ex. 1011, 12:33-41. Here, "TEREDO (64.5.6.7:1025)" represents the IPv6 address IPv4 client 101 constructed for itself as described above with reference to Figures 1 and 7. Ex. 1011, 12:33-54; Ex. 1026, ¶ 38.



To transmit IPv6 response 311 to IPv4 client across IPv4 Internet, relay 202 converts IPv6 response 311 into IPv4 response 311'. Ex. 1011, 12:42-13:3; Ex. 1026, ¶ 39; *see also* Ex. 1011, 11:3-6 ("all IPv6 messages are represented with intermittent dashed/dotted lines, as contrasted with IPv4 message which are represented with solid lines"). Specifically, relay 202 replaces *the IPv6 source address* FEDC:BA98:7654:3210 and *the IPv6 destination address* TEREDO (64.5.6.7:1025) in the header of response 311 respectively with a corresponding IPv4 source address 5.6.7.8:3545 and a corresponding IPv4 destination address 1.2.3.4:3544 and moves the IPv6 address to the payload, forming response 311'. Ex. 1011, 12:33-13:3; Ex. 1026, ¶ 39; Relay 202 then routes response 311'—via server 103 and NAT 102 (as response 311'')—across IPv4 Internet to IPv4 client 101. Ex. 1011, 13:3-43.

C. RFC 4380

RFC 4380, entitled “Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs),” is the IETF technical standard for the embodiments described in Huitema. Ex. 1009, 1; Ex. 1003, ¶ 92. The inventor of the Huitema patent—Christian Huitema—is the author of RFC 4380. *Compare* Ex. 1011, 1 *with* Ex. 1009, 1.

Like Huitema, RFC 4380 is directed to providing interoperability between IPv4 nodes and IPv6 nodes across an IPv6 network. Ex. 1009, 1; Ex. 1003, ¶ 92. RFC 4380 provides interconnection between IPv6 and IPv4 terminals by allowing IPv4 terminals (referred to as a Teredo client) to communicate with an IPv6 network through the use of a Teredo server arranged in an IPv4 network. Ex. 1009, 4 (§ 2.3). The Teredo server communicates with a Teredo relay, which can communicate according to the IPv6 protocol—the Teredo relay thus functions as an IPv6 router. Ex. 1009, 4 (§ 2.4); Ex. 1003, ¶ 92.

To construct an IPv6 packet, the Teredo relay concatenates an IPv6 prefix together with the IPv4 address of the Teredo server, a port number (referred to as a UDP port), and an IPv4 address of the original IPv4 packet, as shown below. Ex. 1009, 11 (§ 4); Ex. 1003, ¶ 92. The UDP port refers to the port number of the Teredo server that is waiting for packets from the IPv4 terminal. Ex. 1009, 4 (§ 2.7); Ex. 1003, ¶ 92

```
+-----+-----+-----+-----+-----+
| Prefix      | Server IPv4 | Flags | Port | Client IPv4 |
+-----+-----+-----+-----+-----+
```

RFC 4380, 11 (§ 4)⁹

RFC 4380 thus provides a 64-bit identifier that contains the UDP port and IPv4 address at which an IPv4 client can be reached through the Teredo service. Ex. 1009, 5 (§ 2.13). RFC 4380 is designed for use when an IPv4 terminal communicates with an IPv4 router that uses network address translation (NAT) to communicate with a public network such as the Internet. Ex. 1009, 8 (§ 3.2.1); Ex. 1003, ¶ 93.

⁹ This Figure corresponds to Figure 7 of Huitema showing the data structure 700 of an IPv6 address. *Compare id.*, 11 *with* Ex. 1011, 8:35-60, FIG. 7; Ex. 1026, ¶ 87.

D. Kong

Kong discloses a system for communicating between IPv4 and IPv6 networks, shown in Figure 1. Ex. 1026, ¶¶ 40-41.

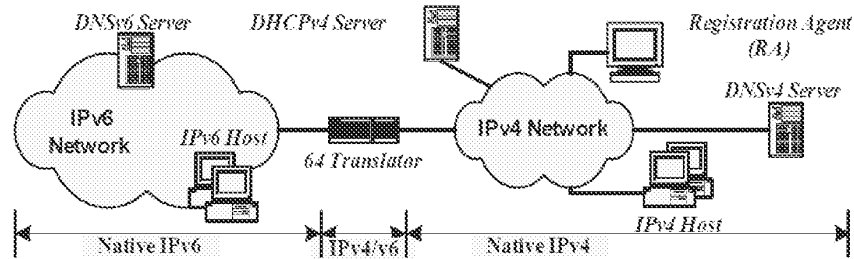


Fig. 1. The overall system architecture using 64TSD

Kong, FIG. 1

As seen above in Figure 1 of Kong, “[t]he overall system architecture using 64TSD consists of IPv6 Network, IPv4 Network and the 64 Translator” allowing an IPv6 Host on an IPv6 Network to communicate with an IPv4 Host on an IPv4 Network. Ex. 1020, 498-499. The 64 Translator “translates a protocol header between IPv6 and IPv4” for packets transitioning from the IPv6 Network to the IPv4 Network, and vice versa. *Id.*, 499.

Figure 2, reproduced below, shows an “outbound” session procedure in which the IPv6 Host establishes communication with the IPv4 Host:

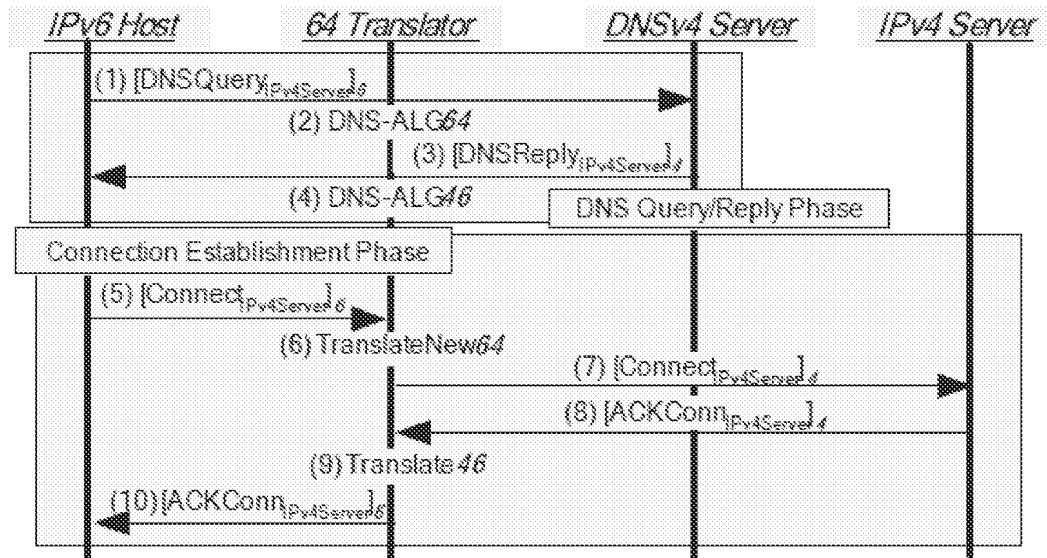


Fig. 2. The procedure of the outbound session

Kong, FIG. 2

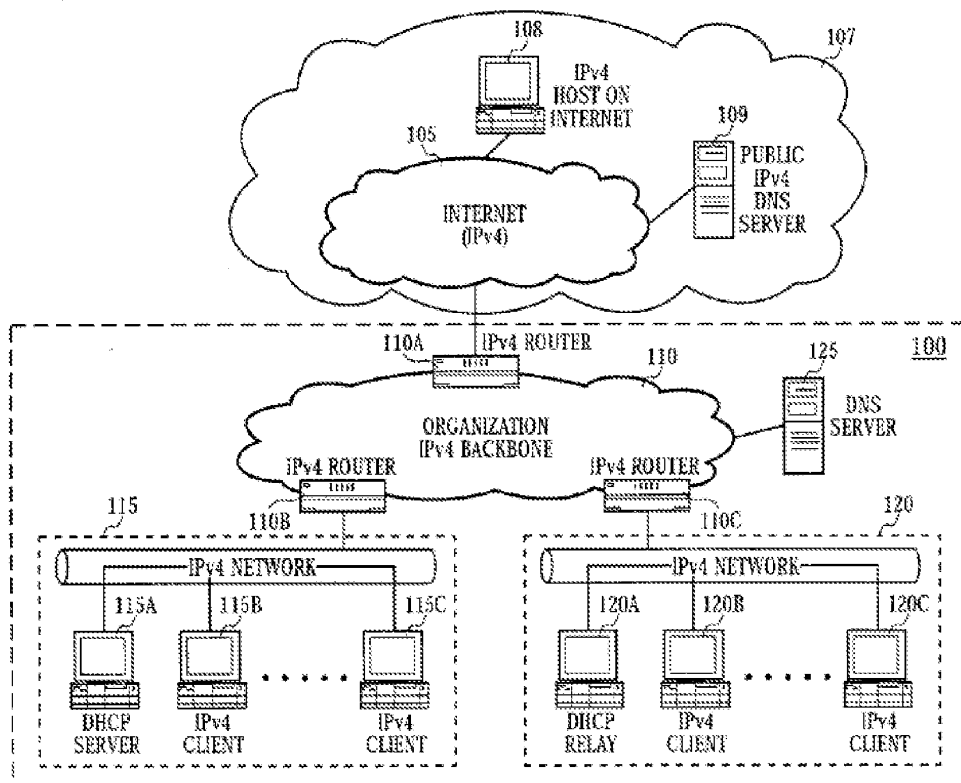
In step “(5) IPv6 Host sends the [Connect_{IPv4Server}]₆ through the 64 Translator to the IPv4 Server.” Ex. 1020, 501. The [Connect_{IPv4Server}]₆ packet is an IPv6 packet, and Kong teaches that an “IPv6 address ... is created by the concatenation of dummy prefix and IPv4 address.” *Id.*; see also Ex. 1026, ¶ 41. And IPv4 addresses include a port number. See, e.g., Ex. 1020, 501 (“<IPv4 address, IPv4 Port>”); see also Ex. 1026, ¶ 41. Then, in step (6) [t]he 64 Translator creates [a] mapping entry and translates the [Connect_{IPv4Server}]₆ into the [Connect_{IPv4Server}]₄—an IPv4 packet—using the mapping entry.” Ex. 1020, 501; see also Ex. 1026, ¶ 41. In step “(7) [t]he 64 Translator sends the translated packet to IPv4 Server.” Ex. 1020, 501.

E. Ananda

Like the ’845 patent, Ananda is directed to enabling devices that are configured for IPv4 networking to communicate across IPv6 networks, and vice versa. Ex. 1005, ¶ 17. Ananda explains that computing devices will transition from IPv4 to IPv6 due to the shortage of address space in IPv4, but that transition will be gradual. Ex. 1005, ¶¶ 6-9, 17. Ananda notes that “IPv6 and IPv4 communication protocols do not interoperate,” and thus there is “a need for transparent communication between IPv4 and IPv6 on communication networks, particularly among IPv4

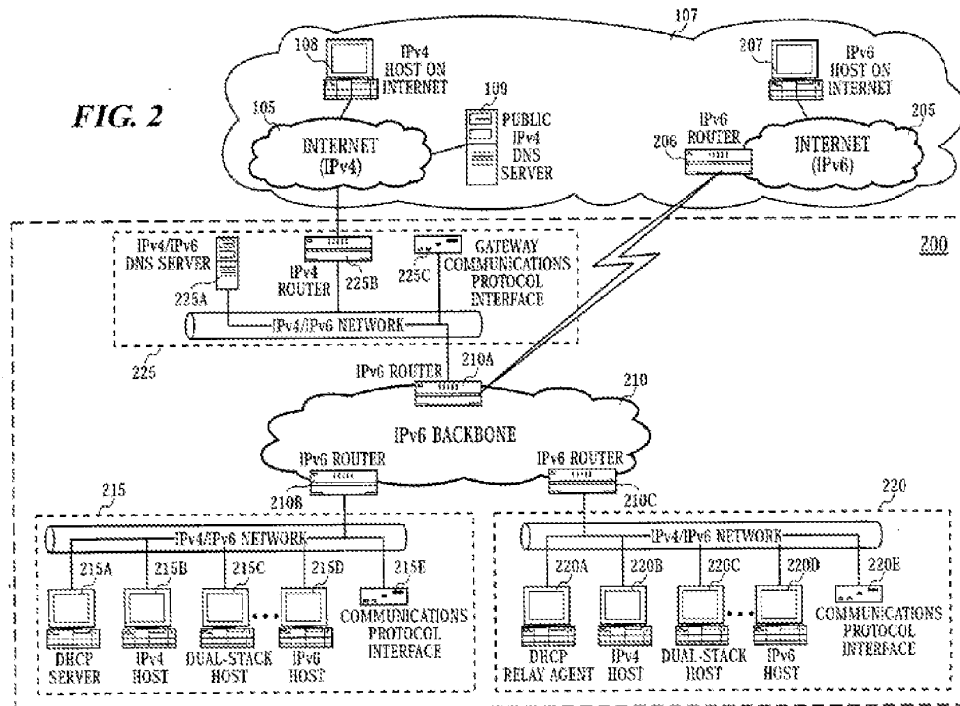
nodes over IPv6 communication networks.” Ex. 1005, ¶ 9.

With reference to Figure 1 below, Ananda first describes a conventional IPv4 network 100 that includes two distinct subnetworks (“subnets”) 115, 120, which are each connected to a private IPv4 backbone network 110 via respective IPv4 routers 110B, 110C. Ex. 1005, ¶¶ 65, 67. The IPv4 backbone network 110 is connected to a public network (IPv4 portion 105 of the Internet 107) via an IPv4 router 110A. Ex. 1005, ¶ 67.



Ananda, Fig. 1.

Ananda discloses that one way to transition IPv4 network 100 to an IPv6 network is to change all the devices in network 100 to operate according to the IPv6 standard, but that may be cost-prohibitive. Ex. 1005, ¶¶ 71-72. Ananda discloses an alternative transition approach, illustrated in Fig. 2 below, that replaces all IPv4 routers 110A-110C in Figure 1 with IPv6 routers 210A-210C and upgrades the IPv4 backbone 110 to an IPv6 backbone 210. Ex. 1005, ¶ 73, 75-77.



Ananda, Fig. 2.

Ananda discloses that the subnet 215 transitions to an IPv4/IPv6 network by using a communications protocol interface (CPI) 215E that can communicate according to both the IPv4 and IPv6 protocols, and the IPv4/IPv6 network includes an IPv4 host 215B and an IPv6 host 215D. Ex. 1005, ¶ 78. The subnet 220 similarly transforms to an IPv4/IPv6 network using an IPv4/IPv6 CPI 220E, an IPv4 host 220B, and an IPv6 host 220D. Ex. 1005, ¶ 79. Similar to the IPv4 clients 115B, 115C in subnet 115 and the IPv4 clients 120B, 120C in subnet 120 in Figure 1, Ananda discloses that the IPv4 host 215B in subnet 215 and IPv4 host 220B in subnet 220 in Figure 2 may be assigned private IPv4 addresses. Ex. 1005, ¶¶ 83-84.

Like the '845 patent, Ananda discloses that when the IPv4 host 215 transmits an IPv4 data packet, the IPv4 data packet is encapsulated in an IPv6 data packet carrying an IPv6 destination address, as discussed below in the detailed application section. Ex. 1005, ¶¶ 103, 105, 137-138; Ex. 1003, ¶¶ 88-89. Further, like the '845 patent, Ananda discloses that the IPv6 destination address is constructed by combining an IPv6 prefix and an IPv4 address of the destination terminal as discussed in Section X.E.5[1.1] below. Ex. 1005, ¶¶ 116, 103, 105-107; Ex. 1003, ¶ 90.

F. Wetterwald

Like the '845 patent and Ananda, Wetterwald is directed to providing interoperability between IPv4 nodes and IPv6 nodes across an IPv6 network. Ex. 1008, 1:8-14. Wetterwald describes an address assignment scheme that allows an enterprise to assign private IPv4 addresses to IPv4 hosts connected to a private IPv4 network, thus limiting the number of globally-unique IPv4 addresses required for hosts connecting to a public IPv4 network, *e.g.*, the Internet. In particular, Wetterwald explains that concerns arise in the event that an IPv6 node is coupled to a private IPv4 network having a Network Address Translator (NAT). NATs perform a Layer-3 translation of IP-Addresses, so that public Internet addresses map to private IP addresses, as described in detail by RFC1918 (Ex. 1007). This mapping has allowed enterprises to map a large number of private addresses to a limited number of public addresses, thus limiting the number of public addresses required by Internet users. Ex. 1008, 1:35-44; Ex. 1003, ¶ 91; *see also* Ex. 1043, ¶¶ 41-44 (citing Exs. 1007, 1047-1048).

IV. Substantial New Questions of Patentability

This Request identifies the following exemplary substantial new questions (SNQ) of patentability for the Challenged Claims.

A. Grounds 1 and 2: Substantial New Questions Based on Huitema

First, Huitema presents a new technological teaching of “[*identifying / means for identifying*] an IPv6 destination address constructed by concatenating an IPv6 prefix, an IPv4 destination address and a destination port number” (elements [1.1], [8.1]). As explained below in Grounds 1 and 2, Huitema discloses an IPv4-IPv6 communication system in which a relay 202 identifies an IPv6 destination address by detecting an IPv6 prefix (a TEREDO prefix) that an IPv4 client used to construct its own IPv6 address for purposes of communicating with the IPv6 client. Ex. 1026, ¶ 44. Moreover, in Huitema, the IPv4 client constructs its own IPv6 address by concatenating the three claimed pieces of information: (1) a TEREDO prefix signifying that the address is an IPv6 address corresponding to an IPv4 client; (2) an IPv4 destination address of that IPv4 client; and (3) a destination port number. *Id.* RFC 4380 also constructing an IPv6 address by concatenating an IPv6 prefix with an IPv4 destination address and port number. *Id.*, ¶ 45.

Huitema also provides a new technological teaching of “[*if necessary, regularizing / means for regularizing, if necessary,*] at least one of the IPv6 source address and the IPv6 destination

address of the data packet by replacing said at least one IPv6 address” (elements [1.2.1], [8.2.1]). As explained below in Grounds 1 and 2, Huitema’s relay 202 regularizes an IPv6 response into an IPv4 response by: (1) replacing the IPv6 source and destination addresses in the header of the IPv6 with corresponding IPv4 source and destination addresses; and (2) moving the replaced IPv6 addresses into the payload of the response. *Id.*, ¶ 46.

Such teachings of Huitema directly undercut the purported reasons the Examiner allowed the ’845 Patent claims. While the Examiner allowed the claims because the prior art purportedly did not disclose the *identifying* and *regularizing* elements [1.1], [8.1], Ex. 1002, 35, as explained in this Request, Huitema discloses these elements. Thus, in Huitema, “the teaching of the (prior art) patents and printed publications is such that a reasonable examiner would consider the teaching to be important in deciding whether or not the claim is patentable.” *See* MPEP § 2242; Ex. 1026, ¶¶ 47-48. Further, because Huitema was cited but not appreciated, the “same question of patentability as to the claim has not been decided by the Office in an earlier concluded examination or review of the patent.” *See* MPEP § 2242; Ex. 1026, ¶ 48. The following grounds of rejection flow from these substantial new questions of patentability: Huitema anticipates and/or would have rendered obvious claims 1 and 5-8.

B. Grounds 3 and 4: Substantial New Questions Based on Kong

Kong presents a new technological teaching of “[*identifying / means for identifying*] an IPv6 destination address constructed by concatenating an IPv6 prefix, an IPv4 destination address and a destination port number” (elements [1.1], [8.1]). Ex. 1026, ¶ 49. As explained below in Grounds 3 and 4, Kong discloses an IPv4-IPv6 communication system in which a 64 Translator identifies a data packet with IPv6 destination address constructed by concatenating a dummy IPv6 prefix, an IPv4 destination address, and a destination port number. Kong further presents a new technological teaching of “*if necessary, regularizing at least one of the IPv6 source address and the IPv6 destination address of the data packet by replacing said at least one IPv6 address*” (elements [1.2.1], [8.2.1]). *Id.*, ¶ 50. Kong’s 64 Translator regularizes an IPv6 data packet into an IPv4 data packet by replacing the IPv6 address with a corresponding IPv4 address. In Kong, “[t]he 64 Translator creates the mapping entry and translates the [ConnectIPv4Server]₆ into the [ConnectIPv4Server]₄ using the mapping entry” and “[t]he 64 Translator sends the translated packet to IPv4 Server.” Ex. 1020, 501; Ex. 1026, ¶ 50.

Such teachings of Kong directly undercut the purported reasons the Examiner allowed the '845 Patent claims. While the Examiner allowed the claims because the prior art purportedly did not disclose the identifying and regularizing elements [1.1], [8.1], Ex. 1002, 35, as explained in this Request, Kong discloses these elements. Ex. 1026, ¶ 51. Thus, in Kong, "the teaching of the (prior art) patents and printed publications is such that a reasonable examiner would consider the teaching to be important in deciding whether or not the claim is patentable." *See* MPEP § 2242. Further, because Kong was not before the Examiner at all, "same question of patentability as to the claim has not been decided by the Office in an earlier concluded examination or review of the patent." *See* MPEP § 2242. Accordingly, the following grounds of rejection flow from these substantial new questions of patentability: Kong anticipates and/or would have rendered obvious claims 1 and 5-8. Ex. 1026, ¶ 51.

C. Ground 5: Substantial New Questions Based on Ananda, Wetterwald, and RFC 4380

Ananda, Wetterwald, and/or RFC 4380 also present a new technological teaching of "[*identifying / means for identifying*] an IPv6 destination address constructed by concatenating an IPv6 prefix, an IPv4 destination address and a destination port number" (elements [1.1], [8.1]). For example, as explained below in Ground 5, Ananda discloses identifying an IPv6 data packet by detecting that the packet has IPv4 data in its payload and/or that the IPv6 packet has a destination address within an IPv4-IPv6 multi-protocol subnet. Ex. 1026, ¶ 52. Ananda discloses constructing an IPv6 destination address by concatenating a 48-bit IPv6 prefix with an IPv4 destination address, and Wetterwald further renders obvious including a destination port number as part of the concatenation. *Id.*

Ananda also provides a new technological teaching of "[*if necessary, regularizing / means for regularizing, if necessary,*] at least one of the IPv6 source address and the IPv6 destination address of the data packet by replacing said at least one IPv6 address" (elements [1.2.1], [8.2.1]). As explained below in Ground 5, Ananda converts an IPv6 packet into an IPv4 packet by decapsulating the IPv6 packet, producing the original IPv4 packet in which an IPv4 destination address replaces the IPv6 destination address of the IPv6 packet. *Id.*

Accordingly, with respect to Ananda, Wetterwald, and RFC 4380, "the teaching of the (prior art) patents and printed publications is such that a reasonable examiner would consider the teaching to be important in deciding whether or not the claim is patentable." *See* MPEP § 2242.

And because the Office did not consider Ananda, Wetterwald, RFC 4380, or the merits of Cisco's IPR petition in IPR2020-01679, the "same question of patentability as to the claim has not been decided by the Office in an earlier concluded examination or review of the patent." *See* MPEP § 2242. The following ground of rejection flows from these substantial new questions of patentability: Ground 5: Ananda, Wetterwald, and RFC 4380 render obvious claims 1 and 5-8.

V. Detailed Application of Challenged Claims to Prior Art

In Grounds 1-5 below, Requester explains in detail and on an element-by-element basis how the prior-art references listed above establish the unpatentability of the Challenged Claims under 35 U.S.C. §§ 102(a), (b), and/or (e) (pre-AIA) or under 35 U.S.C. §§ 103(a) (pre-AIA). To aid the analysis and more easily identify claim elements, Requester has added claim element numbers within square brackets to identify certain phrases within the claims. Requester respectfully requests the issuance of an Office Action rejecting the Challenged Claims based on one or more of Grounds 1-5.

In support of Ground 5, Requester relies on a Declaration of Paul Min, Ph.D. (Ex. 1003) submitted in IPR2020-01679 regarding the proposed combination of Ananda, Wetterwald, and RFC 4380 (Ground 5 in this Request). Additionally, Requester relies on an additional Declaration of Paul Min, Ph.D. in Support of Request for *Ex Parte* Reexamination (Ex. 1026) regarding Grounds 1 and 2 based on Huitema and Grounds 3 and 4 based on Kong.

A. Grounds 1 and 2: Huitema Anticipates and/or Would Have Rendered Obvious Claims 1 and 5-8

As explained below on an element-by-element basis, Huitema anticipates and/or would have rendered obvious would have rendered obvious challenged claims 1 and 5-8 of the '845 Patent under 35 U.S.C. § 103(a) (pre-AIA).

Huitema describes several embodiments of methods in which an IPv4 client 101 and an IPv6 client 201 establish communication, including the methods shown in Figure 2, Figures 3A-3B, and Figure 4. Ex. 1011, 5:54-6:2; Ex. 1026, ¶ 55. These embodiments overlap substantially, and each independently discloses or suggests each element of '845 Patent claims 1 and 5-8. Ex. 1026, ¶ 55. Thus, while Requester focuses below primarily on the Figures 3A-3B embodiment, Requester also identifies below how each of the remaining Huitema embodiments separately discloses or suggest each claim element and renders the challenged claims unpatentable.

1. Independent Claim 1

- a. [1.pre.1] “A method for receiving an IPv6 data packet in an IPv6 domain connected to an IPv4 domain,”

To the extent the preamble is limiting, Huitema discloses this element. Ex. 1026, ¶¶ 56-61. Specifically, Huitema discloses *a method* (message flow of Figures 3A, 3B) for receiving an IPv6 data packet (response 311) in an IPv6 domain (IPv6 Internet) connected to an IPv4 domain (IPv4 Internet):

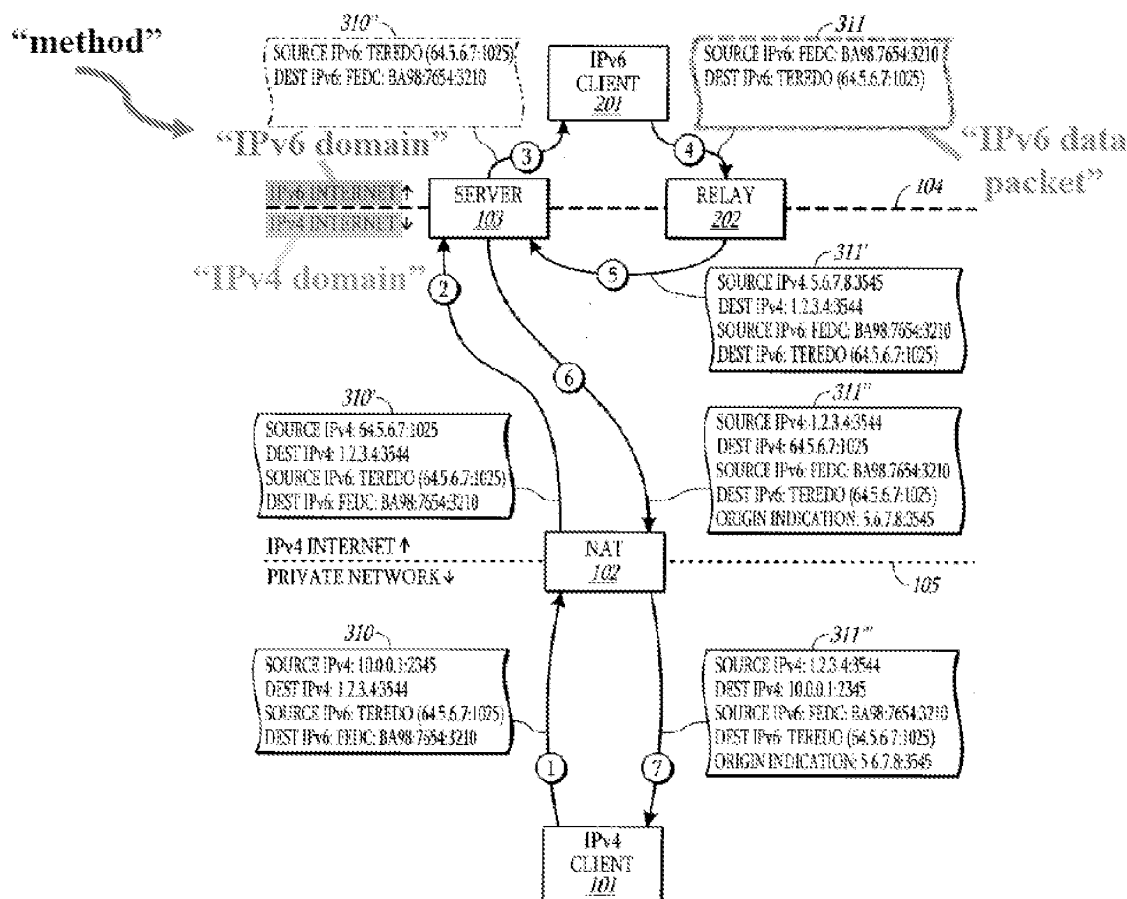


FIG. 3A

Huitema, FIG. 3A*

In Figures 3A, 3B, Huitema shows a “message flow” in which IPv4 client 101 communicates with IPv6 client 201. Ex. 1011, 11:62-14:52. IPv6 client 201 resides in an *IPv6 domain* (IPv6 Internet), IPv4 client 101 resides in a private network of an *IPv4 domain* (IPv4 Internet):

Internet), and relay 202 and server 103 connect the IPv6 domain to the IPv4 domain. *Id.*, 3:3-5:25, 5:57-65, 6:15-18, 11:62-14:52, FIGS. 3A, 3B. As explained above, the claimed “*domain*” refers to a “*network*.” *Supra* Section I.E.1. Huitema’s IPv4 Internet and IPv6 Internet constitute *networks* and thus satisfy this construction. Ex. 1026, ¶ 57.

As shown in Figure 3A of Huitema, to communicate with IPv4 client 101, IPv6 client 201 sends to the IPv4 client 101 an IPv6 response 311 corresponding to the claimed *IPv6 data packet*. Ex. 1011, 12:33-48, FIG. 3A; Ex. 1026, ¶ 58. As seen above, IPv6 response 311 contains the following source and destination IPv6 addresses:

Source IPv6 Address: ::FEDC:BA98:7654:3210

Destination IPv6 Address: Teredo (64.5.6.7:1025)

Ex. 1011, 12:32-40. Huitema explains that, in the method of Figure 3A, “IPv4 client [101] establishes communication with [] IPv6 client [201] while reducing the workload of the server computing system[.]” *Id.*, 5:57-65.

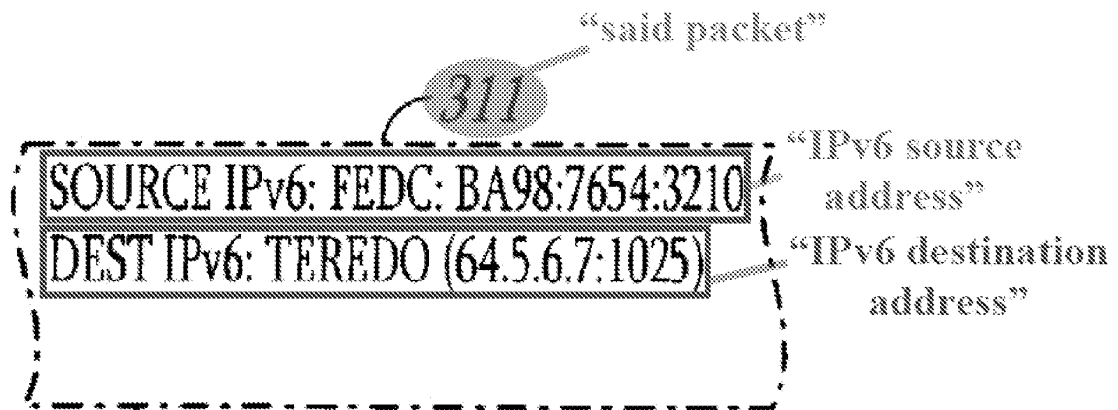
Requester notes that Huitema also discloses the claimed *IPv4 domain* even if improperly interpreted to mean a part of a computer network that uses the IPv4 protocol and in which the data processing resources are under common control. *Supra* Section. I.E.1; Ex. 1026, ¶ 59. For example, in Huitema, “the IPv4 client 101 ... is located behind a non-cone, non-symmetric NAT [102]” or “behind a cone NAT (i.e., the cone bit is set to one) [102]” on a private IPv4 network. Ex. 1011, 13:44-50, FIG. 3A; *see also id.*, 9:39-40 (“the client 101 resides in a private network behind the NAT”); Ex. 1026, ¶ 59. And NAT 102 replaces IPv4 client 101’s private IPv4 address with its public IPv4 address and vice versa, depending on the direction of communication. Ex. 1011, 12:13-23, 13:18-30; *see also id.*, 7:20-41; Ex. 1026, ¶ 59. Thus, the private network on which IPv4 client 101 resides is a part of a computer network that uses the IPv4 protocol and in which the data processing resources are under common control—control by the owner/administrator of that private network. Ex. 1026, ¶ 59.

In addition to Figure 3A, Huitema’s method of Figure 2 and method of Figure 4 separately disclose or suggest *a method* (message flow of Figures 2, 4) *for receiving an IPv6 data packet* (response 211, 410) *in an IPv6 domain* (IPv6 Internet) *connected to an IPv4 domain* (IPv4 Internet). Ex. 1011, 9:35-11:61, 14:43-16:35, FIGS. 2, 4; Ex. 1026, ¶ 60.

Accordingly, Huitema discloses or suggests “a method for receiving an IPv6 data packet in an IPv6 domain connected to an IPv4 domain,” as claimed.

b. [1.pre.2] “said packet comprising an IPv6 destination address and an IPv6 source address,”

Huitema discloses or suggests this element. Ex. 1026, ¶¶ 62-65. Specifically, Huitema discloses *said packet* (response 311) *comprising an IPv6 destination address (DEST IPv6) and an IPv6 source address (SOURCE IPv6)*:



Huitema, FIG. 3A*

As highlighted in the excerpt of Figure 3A above, Huitema’s response 311 (*said packet*) contains a “Source IPv6 Address” of “FEDC:BA98:7654:3210” (*IPv6 source address*) and a “Destination IPv6 Address” of “Teredo(64.5.6.7:1025)” (*IPv6 destination address*) Ex. 1011, 12:32-40. Response 311 is an IPv6 packet because “all IPv6 messages are represented with intermittent dashed/dotted lines, as contrasted with IPv4 message which are represented with solid lines.” *Id.*, 11:3-6; Ex. 1026, ¶ 63.

Separately, Huitema’s method of Figure 2 and method of Figure 4 disclose *said packet* (response 211, 410) *comprising an IPv6 destination address (DEST IPv6: TEREDO (64.5.6.7:1025)) and an IPv6 source address (SOURCE IPv6: FEDC:BA98:7654:3210)*. Ex. 1011, 11:10-19, 14:58-15:3, FIGS. 2, 4; Ex. 1026, ¶ 64.

Accordingly, Huitema discloses “*said packet comprising an IPv6 destination address and an IPv6 source address,*” as claimed.

- c. [1.pre.3] “said method being executed in a home gateway adapted to connect a user terminal to said IPv6 domain, and comprising the following steps:”

To the extent the preamble is limiting, Huitema discloses or suggests this element. Ex. 1026, ¶¶ 66-70. Specifically, Huitema discloses *said method* (message flow of Figures 3A, 3B) being executed in a home gateway (relay 202) adapted to connect a user terminal (IPv4 client 101 or IPv6 client) to said IPv6 domain (IPv6 Internet):

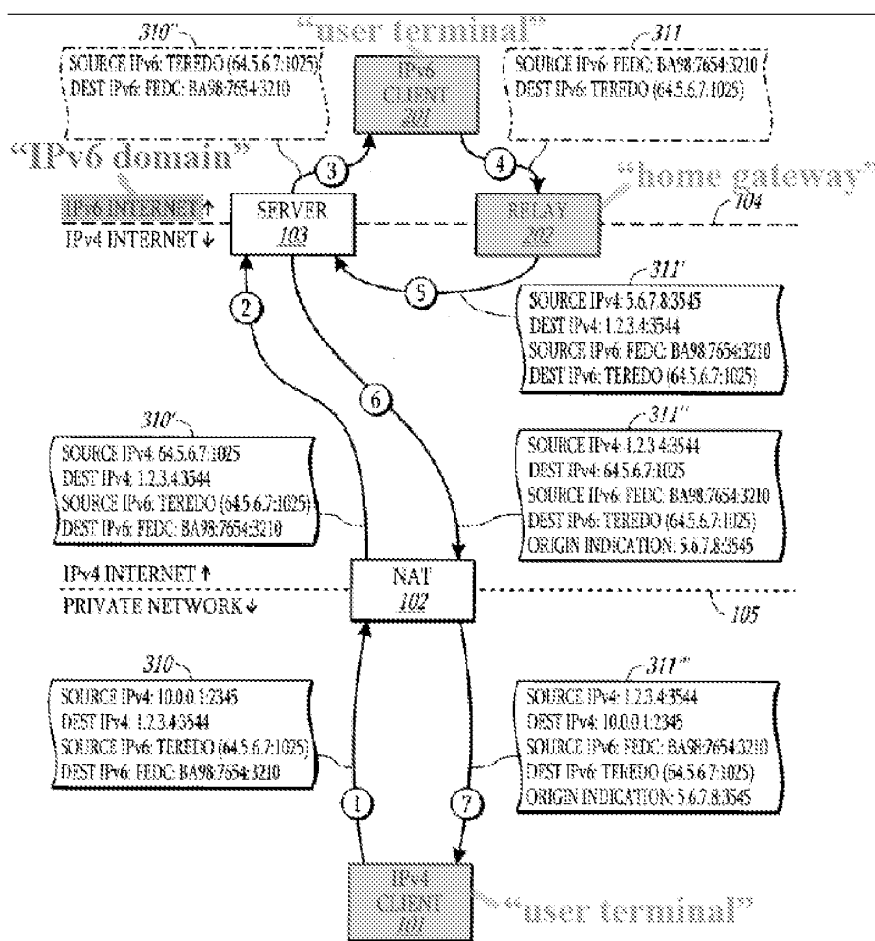


FIG. 3A

Huitema, FIG. 3A*

Huitema’s relay 202 corresponds to the claimed *home gateway* while IPv4 client 101 or IPv6 client 202 corresponds to the claimed *user terminal*. Ex. 1026, ¶ 67. The ’845 Patent describes the *home gateway* broadly as “any equipment” connecting a home network to a wider network.

See Ex. 1001, 1:45-58 (“the expression ‘home gateway’ refers to any equipment for interconnecting a private network and a network operated by a service provider, the private network being either a home network or a business network”). Here, Huitema’s relay 202 sits between the IPv6 and IPv4 Internets and connects IPv4 client 101 on a private network to the IPv6 client 201 on the IPv6 network and vice versa. See, e.g., Ex. 1011, 9:51-52, 11:62-14:52, FIGS. 3A, 3B. Thus, Huitema’s relay 202 is equipment interconnecting a private network (IPv4 client 101’s private network) and a network (IPv6 Internet) operated by a service provider—in the same way the ’845 Patent describes the *home gateway* in the specification. Ex. 1026, ¶ 67.

Indeed, Huitema’s relay 202 plays the same role as products that Patent Owner accuses of infringing claim 1 in related litigation: “The Accused Products comprise a home gateway by virtue of their connection to a private network of user terminals and a service provider network.” Ex. 1073, ¶ 63. The same is true of Huitema’s relay 202 because it connects the private network of IPv4 client 101 and the IPv6 Internet (service provider network). Ex. 1026, ¶ 68.

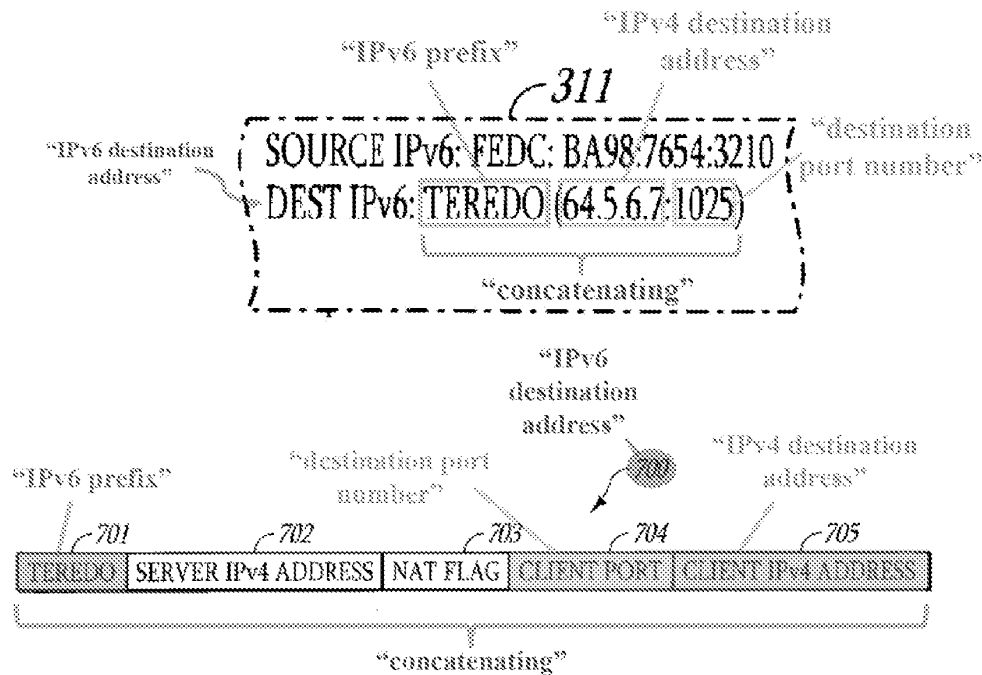
Either IPv4 client 101 or IPv6 client 201 may correspond to the claimed *user terminal* because relay 202 *connects* both devices to the IPv6 Internet (*IPv6 domain*). See, e.g., Ex. 1011, 9:45-54, FIG. 3A; Ex. 1026, ¶ 69. For example, relay 202 connects IPv4 client 101 to the IPv6 Internet because relay 202 sits between the IPv6 and IPv4 Internet and IPv4 client resides in a private network of the IPv4 Internet. See, e.g., Ex. 1011, 9:45-54, FIG. 3A; Ex. 1026, ¶ 69. A POSITA would also understand that relay 202 connects IPv6 client 201 to other IPv6 devices on the IPv6 Internet (not shown). See Ex. 1011, 9:45-54, FIG. 3A; Ex. 1026, ¶ 69; see also Ex. 1011, 18:8-10, 21:65-67, 23:14-16, 24:31-33 (“an IPv6 Internet in which a plurality of IPv6 clients may communicate using IPv6”). Similarly, in the ’845 Patent, terminals 21 and 22 *connect* to IPv6 domain 1 via home gateway 20. See, e.g., Ex. 1001, 8:9-25, FIG. 1.

Separately, Huitema’s method of Figure 2 and method of Figure 4 disclose *said method* (message flow of Figure 2, message flow of Figure 4) *being executed in a home gateway* (relay 202 *adapted to connect a user terminal* (IPv4 client 101 or IPv6 client) *to said IPv6 domain* (IPv6 Internet)). Ex. 1011, 9:35-11:61, 14:43-16:35, FIGS. 2, 4; Ex. 1026, ¶ 70.

Accordingly, Huitema discloses or suggests “*said method being executed in a home gateway adapted to connect a user terminal to said IPv6 domain*,” as claimed.

- d. [1.1] “identifying an IPv6 destination address constructed by concatenating an IPv6 prefix, an IPv4 destination address and a destination port number;”

Huitema discloses or suggests this element. Ex. 1026, ¶¶ 72-89. Huitema discloses relay 202 *identifying* in response 311 *an IPv6 destination address (DEST IPv6) constructed by concatenating an IPv6 prefix (TEREDO service prefix), an IPv4 destination address (64.5.6.7) and a destination port number (1025)*:



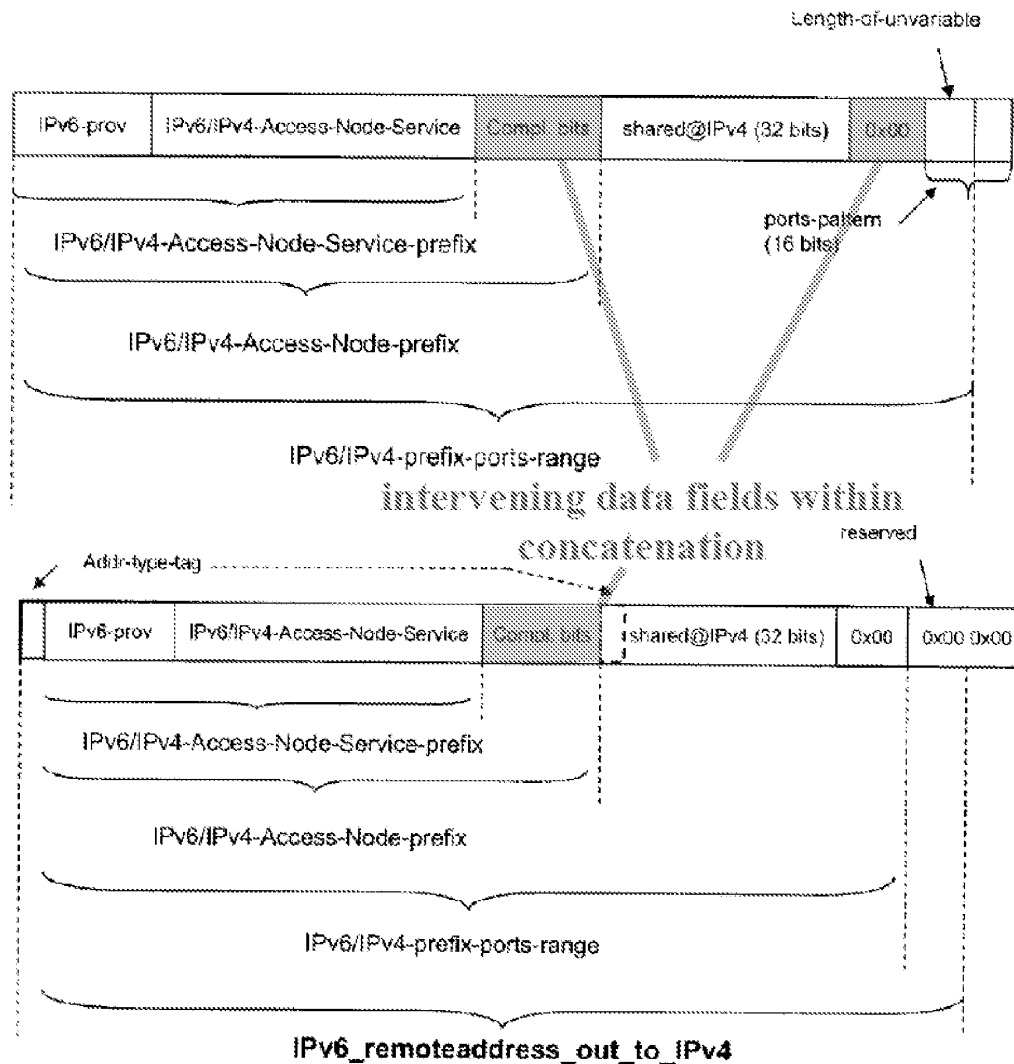
Huitema, FIGS. 3A, 7*

IPv6 prefix (TEREDO): TEREDO represents “[a]n IPv6 addressing prefix that is used to construct the IPv6 address of Teredo clients” from their IPv4 address and port number. Ex. 1009, 4; Ex. 1026, ¶ 73. In the example of Figure 3A, “the address ‘Teredo (64.5.6.7:1025),” shown in message 311 of Figure 3A, “is the same as the address ‘PREF:102:304::FBFE:BFBA:F9F8’.” Ex. 1011, 10:31-33. Here, “PREF:102:304” is “the prefix associated with the server 103.” *Id.*, 12:49-58; Ex. 1026, ¶ 73; *see also id.*, 8:12-16 (“The response includes a prefix assigned to the server 101,” which is “a prefix 111A of ‘PREF:0102:0304::/64’”), FIG. 1.

Huitema explains that IPv6 addresses—including the destination IPv6 address in response 311—have “data structure 700” shown in Figure 7 of Huitema above. Ex. 1011, 8:37-61, FIG. 7;

Ex. 1026, ¶ 74. Appearing in the first field of data structure 700—service field 701—a TEREDO service prefix “PREF” “identifies that this IPv6 address corresponds to an IPv4 client.” Ex. 1011, 8:39-40; Ex. 1026, ¶ 74. Immediately following the TEREDO service prefix “PREF” in service field 702, “[a] server IPv4 address field 702 identifies the IPv4 address of the server 103. For example, the ‘102:304’ in the example IPv6 address is the hexadecimal notation of the address of the server 103 (namely, 1.2.3.4).” Ex. 1011, 8:42-46; Ex. 1026, ¶ 74. The TEREDO service prefix “PREF” in field 701, alone or in combination with the hexadecimal IPv4 address of server 101 in field 702, discloses the claimed *IPv6* prefix. Ex. 1026, ¶ 75.

Requester notes that neither the claim language nor the specification of the ’845 Patent requires the claimed the claimed IPv4 destination address and destination port number to immediately follow the IPv6 prefix, to be adjacent to one another, or to be in any specific order in the concatenation. Ex. 1026, ¶ 76. Rather, the claimed concatenation merely requires a string containing at least these three pieces of information. *Id.*; *see also* Ex. 1072 (defining “concatenate” as “to link structures together”). Indeed, Figures 2 and 4 of the ’845 Patent show exemplary structures of a constructed IPv6 address having intervening data fields separating the IPv4 destination address and/or port number from the IPv6 prefix and/or from one another in the concatenation. *See, e.g.*, Ex. 1001, 6:58-60, 6:63-65, 10:8-50, 11:43-12:23, FIGS. 2, 4; Ex. 1026, ¶ 76. Specifically, Figure 2 shows a first data field (“Compl. bits”) between the IPv6 prefix (“Pv6/IPv4_Access_Node_Service Prefix”) and a second field (“0x00”) between the IPv4 destination address and port number, while Figure 4 similarly shows a “Compl. bits” field between the IPv6 prefix and the IPv4 address:



'845 Patent, FIGS. 2, 4*

If Patent Owner contends (incorrectly) that the claim language requires the IPv4 destination address and destination port number to follow the IPv6 prefix in any specific configuration, such configurations would have been an obvious matter of design choice because the '845 Patent does not identify any significance to the location or ordering of these fields in the concatenation. *See, e.g.,* MPEP 2144.04.VI.C; *In re Chu*, 66 F.3d 292, 298-99 (Fed. Cir. 1995) (affirming design choice rejection where applicant failed to explain how the claimed structure performed differently than the prior art); *In re Kuhle*, 526 F.2d 553, 555 (C.C.P.A. 1975); Ex. 1026, ¶ 77. Indeed, it would have been obvious to arrange Huitema's IPv4 destination address and destination port

number in any operable order following the IPv6 prefix in the concatenation. Ex. 1026, ¶ 77.

IPv4 destination address (64.5.6.7): *Concatenated* with the *IPv6 prefix* TEREDO, IP address 64.5.6.7 is the *IPv4 address* of IPv4 client 101—the *destination* of response 311. Ex. 1026, ¶ 78. As Huitema explains, IPv4 client 101 has “the globally unique [public] IPv4 network address 64.5.6.7[.]” Ex. 1011, 7:51-53. Figure 7 also shows, *concatenated* with the *IPv6 prefix* TEREDO in data structure 700, “client IPv4 address field 705 [that] identifies the IPv4 address of the client 101.” *Id.*, 8:51-55; Ex. 1026, ¶ 78.

Destination port number (1025): As seen in Figure 3A above, the *IPv6 destination address* includes *port number* 1025 *concatenated* with *IPv4 destination address* 64.5.6.7. Ex. 1026, ¶ 79; Specifically, port number 1025 is the UDP port number of IPv4 client 101—the *destination* of response 311. Ex. 1011, 8:6 (“UDP port 1025”), 8:9-10 (“address and port (e.g., 64.5.6.7:1025)”), 8:49-50 (“the mapped port 1025), 11:35, 13:8, 15: 54, 16:7 (“Destination IPv4 Address and Port: 64.5.6.7.1025”); Ex. 1026, ¶ 79 (IPv4 addresses generally have the form <IP address>:<port number>). In Figure 7, Huitema shows “client port field 704” that “identifies the mapped port (e.g., UDP port) of the client 101” *concatenated* with the *IPv4 destination address* and *IPv6 prefix* in data structure 700. Ex. 1011, 8:46-51, FIG. 7; Ex. 1026, ¶ 79.

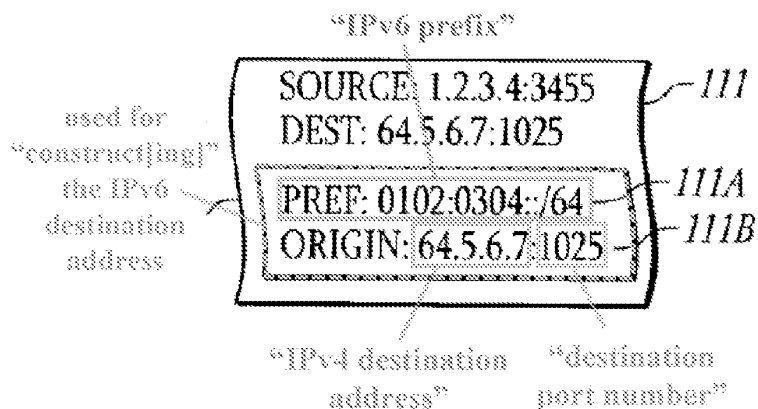
Second, relay 202 *identifies* the *IPv6 destination address*, as claimed, in at least two ways. Ex. 1026, ¶ 80. First, relay 202 identifies the IPv6 destination address by detecting the presence of the IPv6 prefix in that address. *Id.*, ¶ 80. Specifically, “relay 202 receives the response 311” and “*detects the presence of the prefix* associated with the server 10[1] *encoded within its destination IPv6 address.*” Ex. 1011, 12:41, 49-51. That is, relay 202 identifies that it has received an IPv6 destination address by detecting the TEREDO prefix “PREF” contained in field 701 of data structure 700 immediately followed by the address (“102:304”) of server 101 in field 702, discussed and highlighted above in Figures 3A and 7. Ex. 1026, ¶ 80. This is the same way the ’845 Patent teaches identifying a constructed IPv6 destination address: “If a constructed destination address is identified, for example *with the aid of an identifier contained in the destination address*, the step of regularizing” Ex. 1001, 16:23-28; *see also* Ex. 1026, ¶ 80.

Alternatively or additionally, relay 202 *identifies* the IPv6 destination address by modifying IPv6 response 311 into IPv4 response 311’—extracting the IPv6 destination address from the header of IPv6 response 311 and moving it to the payload of IPv4 response 311’, as

discussed below with respect to element [1.2.1]. *Infra* Section V.A.1.e; Ex. 1026, ¶ 81.

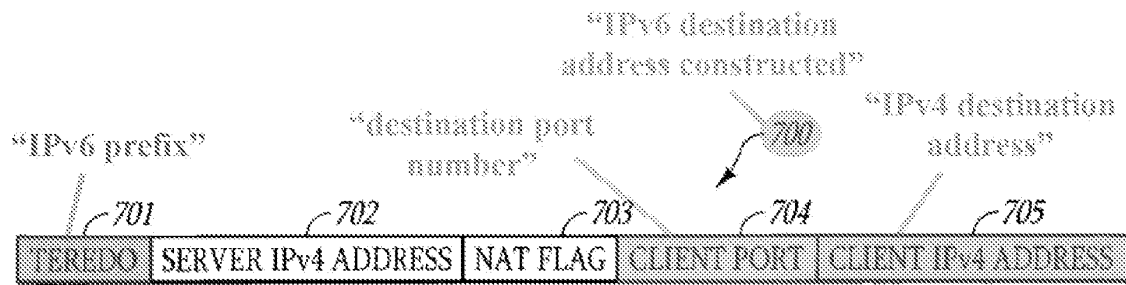
Finally, the *IPv6 destination address* in response 311 constitutes a *constructed* address, as claimed. Ex. 1026, ¶ 82. IPv4 client 101 technically has no IPv6 address because it resides in the IPv4 Internet. *Id.* Rather, as Huitema explains with reference to Figure 1, “client 101 has the private IPv4 network address 10.0.0.1” and a “globally unique [public] IPv4 network address 64.5.6.7.” Ex. 1011, 7:51-55.

But “[e]ven though the client 101 has this IPv4 address, the client now desires an IPv6 network address” to communicate with IPv6 client 202. Ex. 1011, 7:56-63. So, IPv4 client 101 initially sends a request to server 103, which sends a response 111 containing



information 111A, 111B—shown in the excerpt of Figure 1 on the right—*constructing* an *IPv6 destination address* for IPv4 client 101. Ex. 1011, 7:64-8:16, FIG. 1; Ex. 1026, ¶ 83. Specifically, server 103 “sends back a response [111] to that address and port [64.5.6.7.1025]” that “includes a prefix assigned to the server 101, as well as an origin indication. For example, router advertisement message 111 includes a prefix 111A of ‘PREF:0102:0304::/64’ and an origin indication 111B of ‘64.5.6.7:1025.’” Ex. 1011, 8:8-16; Ex. 1026, ¶ 83.

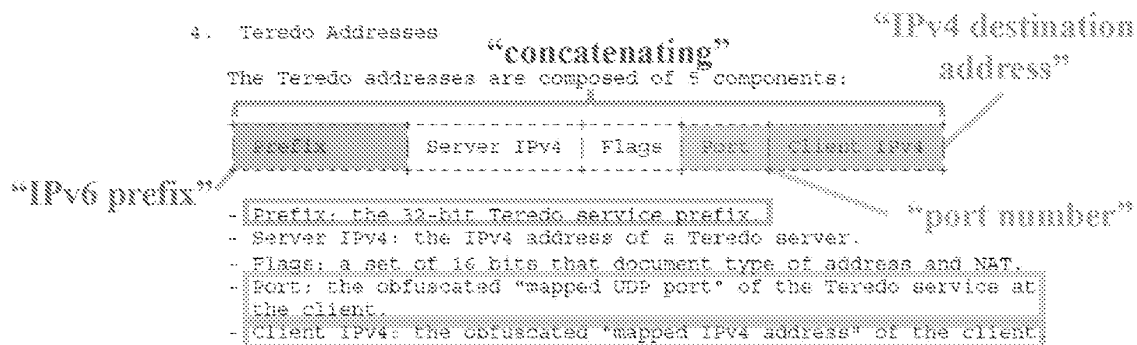
After receiving response 111, “[t]he client 101 then *constructs* its own IPv6 address, which in this example may be ‘PREF:102:304::FBFE:BFFA:F9F8’” that “ha[s] a general structure illustrated in FIG. 7 as data structure 700” highlighted below. Ex. 1011, 8:35-60; *see also* Ex. 1026, ¶ 84.



Huitema, FIG. 7*

Then, when responding to message 310 in Figure 3A, IPv6 client 201 sends response 311 to IPv4 client 101 that includes this *constructed destination IPv6 address* "Teredo (64.5.6.7:1025)." Ex. 1011, 8:35-60; *see also* Ex. 1026, ¶ 84.

RFC 4380 (Ex. 1009) is the IETF specification for "Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)"—the same Teredo service Huitema uses for IPv6 addresses. Ex. 1009, 1; Ex. 1026, ¶ 85. RFC 4380 further confirms¹⁰ that Huitema's *IPv6 destination address* TEREDO (64.5.6.7:1025) comprises a *constructed concatenation of an IPv6 prefix, an IPv4 destination address, and a destination port number*, as claimed:



RFC 4380, 11-12*

¹⁰ See MPEP § 2131.01.III (citing *Cont'l Can Co. USA v. Monsanto Co.*, 948 F.2d 1264, 1268, 20 USPQ2d 1746, 1749-50 (Fed. Cir. 1991) (In the context of anticipation by a first reference, a second reference may be relied on to show inherent characteristics of the first reference)).

As seen above, RFC 4380 provides a data structure for Teredo IPv6 addresses identical to data structure 700 of Huitema Figure 7. *Compare* Ex. 1009, 11-12 *with* Ex. 1011, 6:7-8, 8:35-60, FIG. 7; *see also* Ex. 1026, ¶ 87. Like Huitema's data structure 700, RFC 4380 shows Teredo IPv6 addresses comprise a *concatenation of an IPv6 prefix* ("Prefix: the 32-bit Teredo service prefix" alone or together with the adjacent "Server IPv4: the IPv4 address of a Teredo server"), an *IPv4 destination address* ("Client IPv4: the ... 'mapped IPv4 address' ... at the client") and a *destination port number* ("Port: the ... 'mapped IPv4 address' of the client"). Ex. 1009, 11-12; Ex. 1026, ¶ 87. In Huitema, the 32-bit Teredo service prefix is "[a]n IPv6 addressing prefix whose value is 2001:0000:/32." Ex. 1009, 4.

Separately, Huitema's method of Figure 2 and method of Figure 4 respectively disclose *identifying* in responses 211, 410 *an IPv6 destination address (DEST IPv6) constructed by concatenating an IPv6 prefix (TEREDO service prefix), an IPv4 destination address (64.5.6.7) and a destination port number (1025)*. Ex. 1011, 11:19-26, 14:58-15:16; Ex. 1026, ¶ 88.

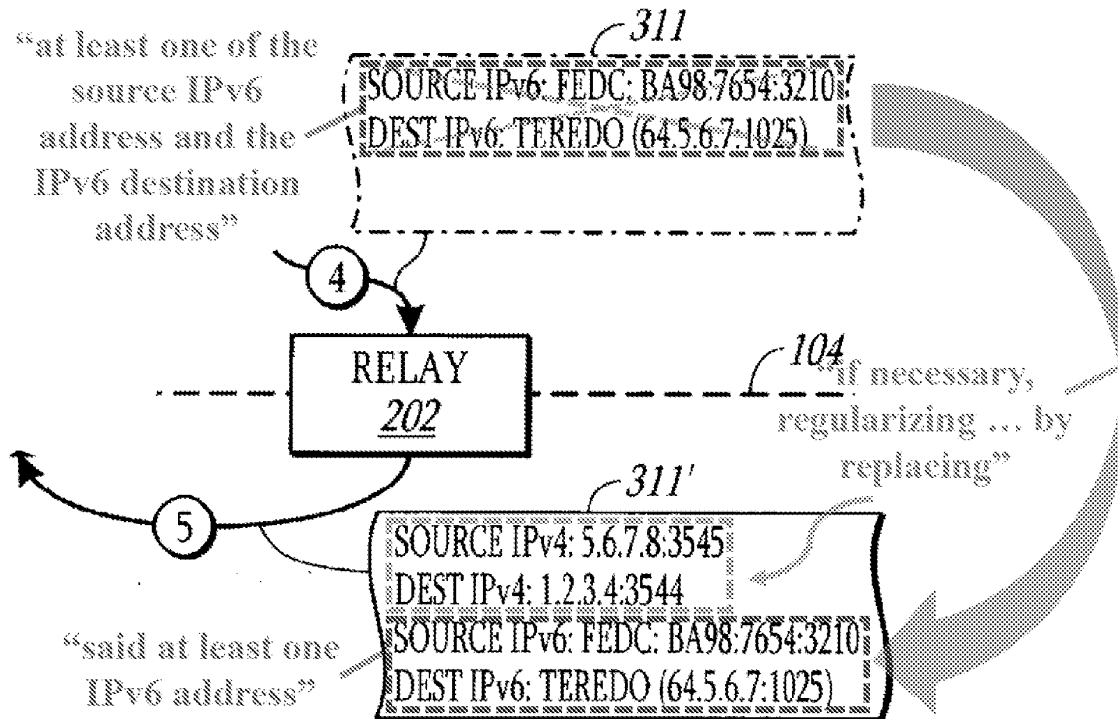
Accordingly, Huitema discloses "*identifying an IPv6 destination address constructed by concatenating an IPv6 prefix, an IPv4 destination address and a destination port number,*" as claimed.

- e. **[1.2.1] "if necessary, regularizing at least one of the IPv6 source address and the IPv6 destination address of the data packet by replacing said at least one IPv6 address,"**¹¹

Huitema discloses or suggest this element. Ex. 1026, ¶¶ 90-102. Huitema's relay 202, *if necessary* (if a TEREDO prefix is detected), *regularizes* (modifies from IPv6 to IPv4) *at least one*

¹¹ The phrase "if necessary" makes element [1.2.1] a conditional limitation, because performance of the recited features is predicated on whether the condition "if necessary" is satisfied. Elements [1.2.2] and [1.2.3] are conditional limitations because they relate to limitation [1.2.1]. *See Ex Parte Schulhauser*, Appeal No. 2013-007847, at 8-10 (PTAB Apr. 28, 2016) (precedential) (conditional steps in method claims need not be carried out when conditions precedent to those steps are not satisfied). Requester shows that all features of limitations [1.2.1]-[1.2.3], as well as the corresponding limitations [8.2.1]-[8.2.3] of claim 8, are obvious based on the assumption that such steps are "necessary." Ex. 1003, ¶ 150.

of the IPv6 source address (FEDC:BA98:7654:3210) and the IPv6 destination address (TEREDO (64.5.6.7:1025)) of the data packet (response 311) by replacing said at least one IPv6 address (replacing FEDC:BA98:7654:3210 and TEREDO (64.5.6.7:1025) respectively with IPv4 source address 5.6.7.8:3545 and IPv4 destination address 1.2.3.4:3544):



Huitema, FIG. 3A*

- **Regularizing at least one of the IPv6 source address and the IPv6 destination address of the data packet by replacing said at least one IPv6 address**

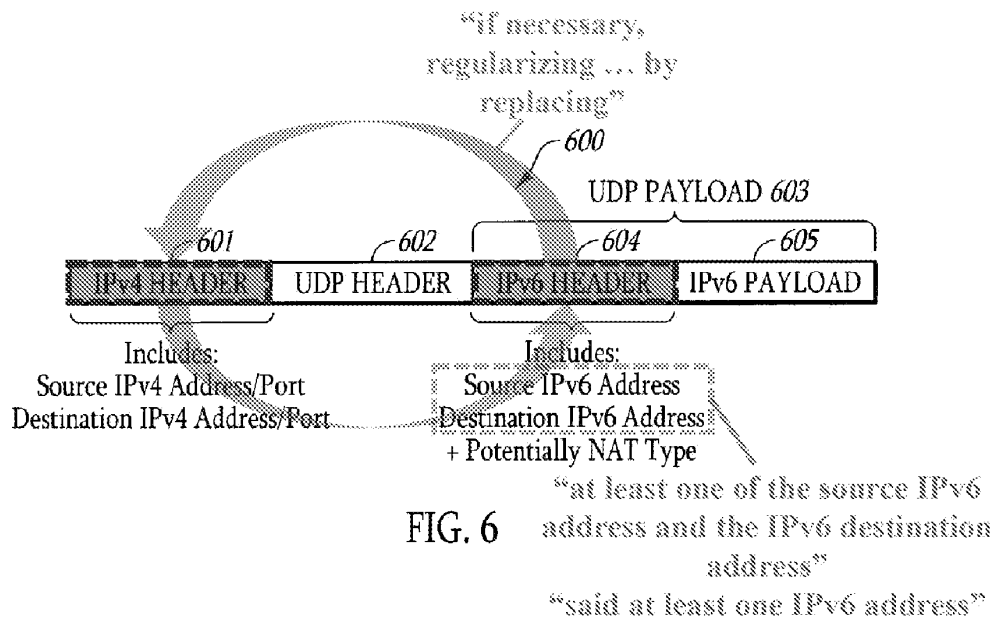
As explained above, this phrase means (1) regularizing the IPv6 source address by replacing the IPv6 source address or (2) regularizing the IPv6 destination address by replacing the IPv6 destination address. *Supra* Section I.E.2. Huitema does both. Ex. 1026, ¶ 91.

As highlighted above, response 311 has an IPv6 format because "all IPv6 messages are represented with intermittent dashed/dotted lines" and thus its header contains IPv6 source and destination addresses. Ex. 1011, 11:3-6; *see also* Ex. 1026, ¶ 92. Whereas modified response 311' has an IPv4 format because "IPv4 message[s] ... are represented with solid lines" and thus has IPv4 source and destination addresses in its header. Ex. 1011, 11:3-6; Ex. 1026, ¶ 92.

Relay 202 *regularizes* IPv6-format response 311 into IPv4-format response 311'. Ex. 1011, 12:33-67; Ex. 1026, ¶ 93. To do so, relay 202 *regularizes the IPv6 source address* (FEDC:BA98:7654:3210) in the header of IPv6 response 311 *by replacing the IPv6 source address* with a corresponding IPv4 source address (5.6.7.8:3545) in the header of IPv4 response 311' and moving the IPv6 source address (FEDC:BA98:7654:3210) from the header to the payload in IPv4 response 311'. Ex. 1011, 12:33-67; Ex. 1026, ¶ 93.

Likewise, relay 202 *regularizes the IPv6 destination address* (TEREDO (64.5.6.7:1025)) *by replacing the IPv6 destination address* in the header of IPv6 response 311 with a corresponding IPv4 destination address (1.2.3.4:3544) in the header of IPv4 response 311' and moving the IPv6 destination address (TEREDO (64.5.6.7:1025)) from the header to the payload in IPv4 response 311'. Ex. 1011, 12:33-67; Ex. 1026, ¶ 94.

Figure 6 of Huitema further illustrates how relay 202 *regularizes* IPv6 response 311 into IPv4 response 311'. Ex. 1026, ¶ 95. Specifically, when converting IPv6 messages to IPv4, "relay 202 reconfigures the IPv6 message back into an IPv4 message that has the data structure [600] illustrated and described with respect to FIG. 6." Ex. 1011, 11:27-29; *see also* Ex. 1026, ¶ 95.



Huitema, FIG. 6*

In Figure 6, Huitema shows "a general IPv4 message data structure 600" for all IPv4 messages.

Ex. 1011, 9:55-56; *see also* Ex. 1026, ¶ 95. That “data structure 600 includes an IPv4 header field 601 ... includ[ing] the source IPv4 address and port, and the destination IPv4 address and port” and “an IPv6 header field 604 ... includ[ing] the source IPv6 address and the destination IPv6 address.” Ex. 1011, 9:58-64.

As highlighted above, to *regularize* IPv6 response 311 into an IPv4 response 311', relay 202: (1) moves the IPv6 addresses from the header to the payload; and (2) moves the corresponding IPv4 addresses from the payload to the header to replace the IPv6 addresses, generating a regularized response. *See* Ex. 1011, 12:33-67; Ex. 1026, ¶ 97. Consistent with this, Huitema's claims describe systems and methods in which the relay “intercepts” an IPv6 response with IPv6 addresses and having IPv4 addresses in its payload and constructs a corresponding IPv4 response with the IPv4 addresses from the payload of the IPv6 response and moving the IPv6 addresses to the payload of the IPv4 response). *See* Ex. 1011, 18:7-28:6; Ex. 1026, ¶ 97.

Accordingly, Huitema *regularizes* IPv6 response 311 in the same way the '845 Patent teaches that the gateway regularizes IPv6 packets—translating IPv6 response 311 into IPv4 response 311' by replacing the constructed IPv6 address. Ex. 1026, ¶ 98. For example, the '845 Patent specification describes receiving an IPv6 packet, “extract[ing] an IPv4 destination address and a destination port from the IPv6 destination address,” and “then translat[ing] the IPv6 packet into an IPv4 packet” having the extracted IPv4 destination address and port number. Ex. 1001, 14:15-46; *see also* Ex. 1026, ¶ 98; Ex. 1001, 16:23-45 (“If a constructed destination address is identified,” then “the step of regularizing ... replaces the constructed destination address in the IPv6 data packet with this native address”).

- *If necessary*

Relay 202 regularizes IPv6 response 311 into IPv4 response 311' *if necessary*, as claimed, by “*detect[ing] the presence of the prefix associated with the server 103* encoded within its destination IPv6 address.” Ex. 1011, 12:41, 49-51; Ex. 1026, ¶ 99. That is, relay 202 determines *if* modification is *necessary* by detecting the *IPv6 prefix* (Teredo prefix) in the header of IPv4 response 311, and relay 202 modifies IPv4 response 311 into IPv6 response 311' *if* relay 202 detects the prefix. Ex. 1026, ¶ 99.

Separately, Huitema's method of Figure 2 discloses, *if necessary* (if a TEREDO prefix is detected), *regularizing* (modifying from IPv6 to IPv4) *at least one of the IPv6 source address*

(FEDC:BA98:7654:3210) *and the IPv6 destination address (TEREDO (64.5.6.7:1025)) of the data packet (response 211) by replacing said at least one IPv6 address (replacing FEDC:BA98:7654:3210 and TEREDO (64.5.6.7:1025) respectively with IPv4 source address 1.2.3.4:3544 and IPv4 destination address 64.5.6.7:1025. Ex. 1011, 11:18-56; Ex. 1026, ¶ 100.*

Likewise, Huitema's method of Figure 4 discloses, *if necessary (if a TEREDO prefix is detected), regularizing (modifying from IPv6 to IPv4) at least one of the IPv6 source address (FEDC:BA98:7654:3210) and the IPv6 destination address (TEREDO (64.5.6.7:1025)) of the data packet (response 410) by replacing said at least one IPv6 address (replacing FEDC:BA98:7654:3210 and TEREDO (64.5.6.7:1025) respectively with IPv4 source address 5.6.7.8:3545 and IPv4 destination address 64.5.6.7:1025. Ex. 1011, 14:58-15:3, 15:66-16:12; Ex. 1026, ¶ 101.*

Accordingly, Huitema discloses "*if necessary, regularizing at least one of the IPv6 source address and the IPv6 destination address of the data packet by replacing said at least one IPv6 address,*" as claimed.

f. [1.2.2] "*the replacement process belonging to a group comprising replacement of a constructed address with a native address and replacement of a native address with a constructed address, and*"

Huitema discloses or suggest this element. Ex. 1026, ¶ 103. Specifically, Huitema discloses *the replacement process (replacing the IPv6 addresses in the header of IPv6 response 311 with corresponding IPv4 addresses to form IPv4 response 311') belonging to a group comprising replacement of a constructed address (TEREDO (64.5.6.7:1025)) with a native address (1.2.3.4:3544) and replacement of a native address (FEDC:BA98:7654:3210) with a constructed address (5.6.7.8:3545), as claimed. Ex. 1026, ¶ 103.*

As explained above with reference to element [1.2.1], in regularizing IPv6 response 311 into IPv4 response 311', Huitema's relay 202 *replaces* IPv6 destination address TEREDO (6.4.5.6.7:1025) with corresponding IPv4 destination address 1.2.3.4:3544. *Supra* Section V.A.1.e; Ex. 1011, 12:31-67; Ex. 1026, ¶ 104. Here, TEREDO (6.4.5.6.7:1025) is a *constructed address* for IPv4 client 101, as claimed, because IPv4 client 101 does not have a native IPv6 address. Ex. 1026, ¶ 104. Instead, "client 101 ... *constructs* its own IPv6 address" from the IPv6 TEREDO service prefix 111a and client 101's own IPv4 address 111B of 64.5.6.7:1025 provided in response 111 received from server 103. Ex. 1011, 8:8-26, 35-60, FIGS. 1, 7; *see also* Ex. 1026, ¶ 104. And IPv4

destination address 1.2.3.4:3544 that replaces the IPv6 address is a *native address* because it is native to IPv4 Internet on which IPv4 client 101 resides. Ex. 1026, ¶ 104.

Also explained with reference to element [1.2.1], in regularizing IPv6 response 311 into IPv4 response 311', Huitema's relay 202 further replaces IPv6 source address FEDC:BA98:7654:3210 with corresponding IPv4 source address 5.6.7.8:3544. *Supra* Section V.A.1.e; Ex. 1011, 12:31-67; Ex. 1026, ¶ 105. Here, FEDC:BA98:7654:3210 is a *native* IPv6 address on IPv6 Internet for IPv6 client 201. Ex. 1011, 10:34-41 ("The destination IPv6 address '::FEDC:BA98:7654:3210' is provided by way of being an example of the IPv6 address of the IPv6 client 201"); Ex. 1026, ¶ 105. But "IPv6 client 201 presumably has no IPv4 address since it is not in the IPv4 Internet," so IPv4 source address 5.6.7.8:3544 that replaces IPv6 client 201's native IPv6 address in response 311' constitutes a *constructed address* for IPv6 client 201. Ex. 1011, 10:26-27; *see also* Ex. 1026, ¶ 105. In particular, IPv4 source address 5.6.7.8:3544 corresponds to the IPv4 address of relay 202, which serves as a proxy for an IPv4 address of IPv6 client 201. Ex. 1011, 10:23-27 ("Note that the IPv4 message includes the IPv4 address and port of the relay 202 (i.e., 5.6.7.8:3455), just what the IPv4 client 101 was trying to discover"); *see also* Ex. 1026, ¶ 105.

Separately, Huitema's method of Figure 2 discloses *the replacement process* (replacing the IPv6 addresses in the header of IPv6 response 211 with corresponding IPv4 addresses to form IPv4 response 211') *belonging to a group comprising replacement of a constructed address* (TEREDO (64.5.6.7:1025)) *with a native address* (64.5.6.7:1025) *and replacement of a native address* (FEDC:BA98:7654:3210) *with a constructed address* (1.2.3.4:3544), as claimed. Ex. 1011, 11:29-57; Ex. 1026, ¶ 106.

Likewise, Huitema's method of Figure 4 discloses *the replacement process* (replacing the IPv6 addresses in the header of IPv6 response 411 with corresponding IPv4 addresses to form IPv4 response 410') *belonging to a group comprising replacement of a constructed address* (TEREDO (64.5.6.7:1025)) *with a native address* (64.5.6.7:1025) *and replacement of a native address* (FEDC:BA98:7654:3210) *with a constructed address* (5.6.7.8:3545), as claimed. Ex. 1011, 14:58-67, 15:66-16:12; Ex. 1026, ¶ 107.

Accordingly, Huitema discloses or suggests "*the replacement process belonging to a group comprising replacement of a constructed address with a native address and replacement of a*

native address with a constructed address,” as claimed.

g. [1.3] “*modifying the data packet using the regularized address; and*”

Huitema discloses or suggests this element. Ex. 1026, ¶¶ 109-112. Specifically, for elements [1.2.1] and [1.2.2], Huitema’s relay 202 *modifies the data packet* (converts IPv6 response 311 into IPv4 response 311’) *using the regularized address* (using IPv4 destination address 1.2.3.4:3544 and/or IPv4 source address 5.6.7.8:3544 in the header of response 311’). *Id.*, ¶ 109.

As explained above with reference to element [1.2.1], Huitema’s relay 202 *modifies* IPv6 response 311 into IPv4 response 311’ by replacing IPv6 destination address TEREDO (6.4.5.6.7:1025) and IPv6 source address FEDC:BA98:7654:3210 in the header of response 311 respectively with IPv4 destination address 1.2.3.4:3544 and IPv4 source address 5.6.7.8:3544 in the header of response 311’. *Supra* Section V.A.1.e; Ex. 1011, 12:33-13:3; Ex. 1026, ¶ 110. Here, IPv4 destination address 1.2.3.4:3544 and/or IPv4 source address 5.6.7.8:3544 disclose the claimed *regularized address* because IPv6 response 311 has been regularized into response 311’ using these addresses. Ex. 1026, ¶ 110. Thus, relay 202 *modifies* response 311 (*the data packet*) *using* IPv4 destination address 1.2.3.4:3544 and IPv4 source address 5.6.7.8:3544 (*the regularized address*), as claimed. *Id.*

Separately, Huitema’s method of Figure 2 discloses *modifying the data packet* (IPv6 response 211) *using the regularized address* (source IPv4 address 1.2.3.4:3544 and/or destination IPv4 address 64.5.6.7:1025), as claimed. Ex. 1011, 11:29-57; Ex. 1026, ¶ 111. Likewise, Huitema’s method of Figure 4 discloses *modifying the data packet* (IPv6 response 410) *using the regularized address* (source IPv4 address 64.5.6.7:1025 and/or destination IPv4 address 5.6.7.8:3545). Ex. 1011, 14:58-67, 15:66-16:12; Ex. 1026, ¶ 111.

Accordingly, Huitema discloses “*modifying the data packet using the regularized address,” as claimed.*

h. [1.4] “*routing the IPv6 data packet to its destination.*”

Huitema discloses or suggests this element. Ex. 1026, ¶ 113-116. Specifically, Huitema’s relay 202 *routes the IPv6 data packet* (response 311, 311’, 311’) *to its destination* (IPv4 client 101):

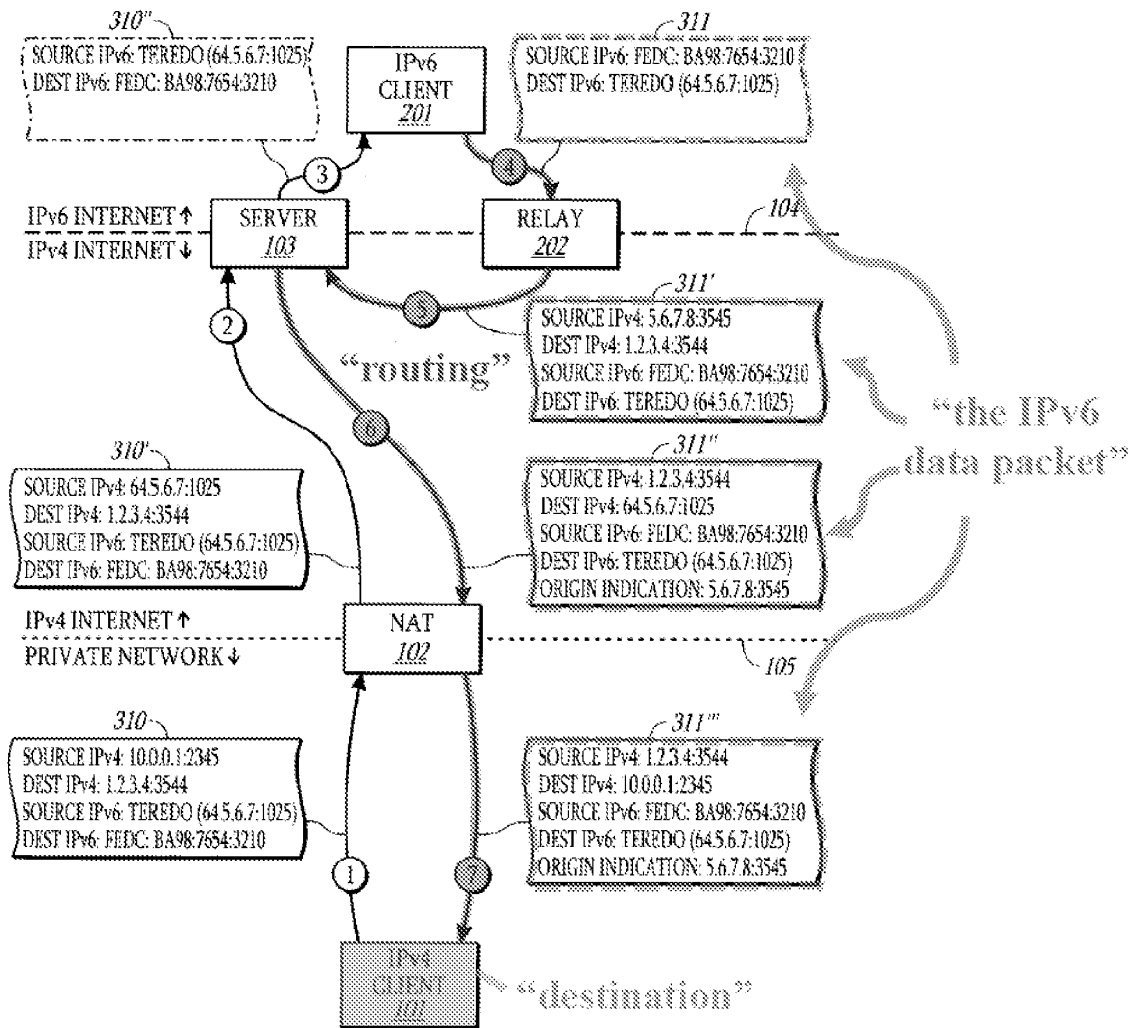


FIG. 3A

Huitema, FIG. 3A*

As highlighted above, relay 202 receives IPv6 response 311, regularizes response 311 into IPv4 response 311', and routes IPv4 response to server 103. Ex. 1011, 12:34-67; Ex. 1026, ¶ 114. Server 103 then modifies IPv4 response 311' into IPv4 response 311'' in which “the IPv4 address of the relay 202 is preserved within an origin indication field ... included as part of the UDP payload along with the Source and Destination IPv6 addresses” and routes modified response 311'' to NAT 102. Ex. 1011, 12:55-13:19; *see also* Ex. 1026, ¶ 114. NAT 102 then modifies response 311'' into response 311''' by substituting the public IPv4 destination address 64.5.6.7:1025 of IPv4 client

101 (*destination*) with client 101's private IPv4 destination address 10.0.0.0.1:2345, and then routes response 311''' to IPv4 client 101 (*destination*). Ex. 1011, 13:19-43; Ex. 1026, ¶ 114.

Separately, Huitema's method of Figure 2 discloses *routing the IPv6 data packet* (response 211, 211', 211'') *to its destination* (IPv4 client 101). Ex. 1011, 11:10-61, FIG. 2; Ex. 1026, ¶ 115. Likewise, Huitema's method of Figure 4 discloses *routing the IPv6 data packet* (response 410, 410', 410'', 410''') *to its destination* (IPv4 client 101). Ex. 1011, 14:58-16:35, FIG. 4; Ex. 1026, ¶ 115.

Accordingly, Huitema discloses "*routing the IPv6 data packet to its destination,*" as claimed.

2. Dependent Claim 5

Claim 5 depends from independent claim 1 and recites the additional elements quoted below. Huitema renders unpatentable independent claim 1 for the reasons discussed above. *Supra* Section V.A.1. Moreover, Huitema discloses or suggests the additional elements of claim 5. Ex. 1026, ¶¶ 117-121.

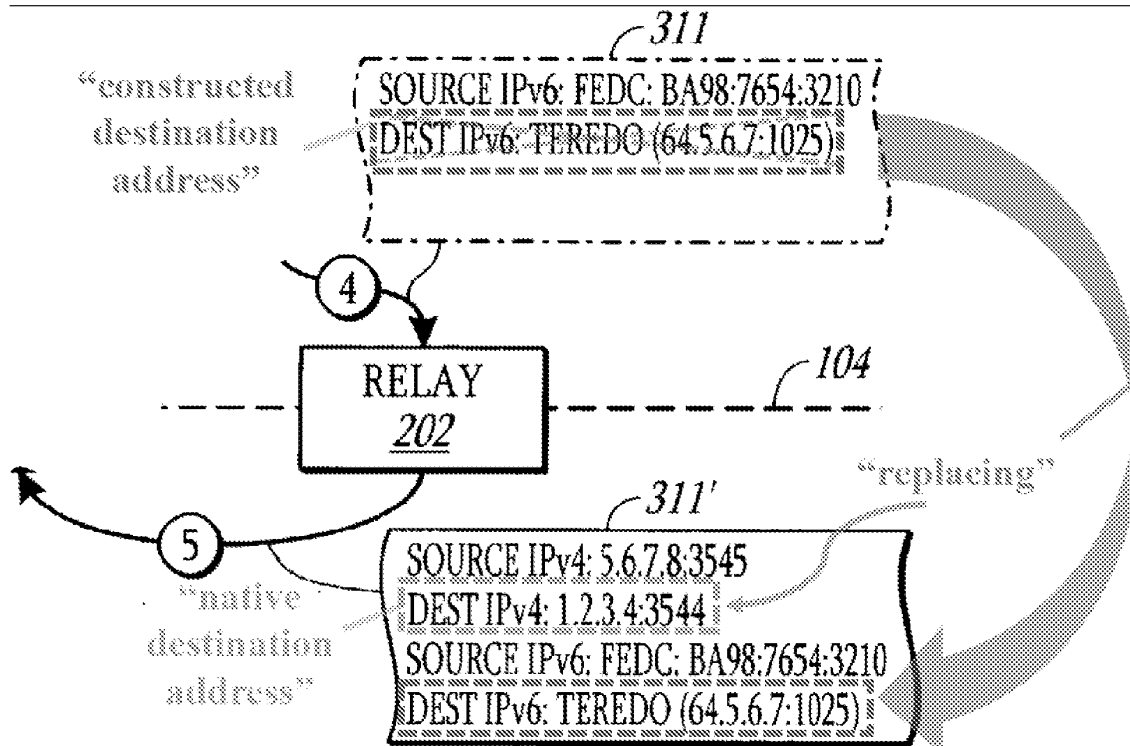
- a. [5.pre] "*The method according to claim 1, wherein when the data packet comes from the IPv4 domain and goes to said user terminal, the step of regularizing the destination address comprises the following steps:*"

Huitema discloses or suggests this element. Ex. 1026, ¶ 118. As explained above for element [1.4], relay 202 receives IPv6 response 311, regularizes it into IPv4 response 311', and routes response 311'—through server 103 (as response 311') and NAT 102 (as response 311'')—to IPv4 client (*user terminal*) across IPv4 Internet (*IPv4 domain*). Ex. 1011, 11:62-14:42, FIG. 3A; Ex. 1026, ¶ 118. Thus, Huitema discloses that *the data packet* (response 311-311'') *comes from the IPv4 domain* (IPv4 Internet) *and goes to said user terminal* (IPv4 client 101), as claimed. Ex. 1026, ¶ 118.

- b. [5.1] "*searching an address translation table of the gateway for an entry associating with said constructed destination address a native destination address;*"
[5.2] "*if an entry is found, replacing the constructed destination address of the data packet with the native destination address from that entry.*"

Huitema discloses or suggests these elements. Ex. 1026, ¶¶ 119-121. Specifically, Huitema

discloses searching an address translation table of the gateway (relay 202) for an entry associating with said constructed destination address (DEST IPv6: TEREDO (6.4.5.6.7:1025)) a native destination address (DEST IPv4: 1.2.3.4:3544). *Id.* Additionally, Huitema discloses if an entry is found, replacing the constructed destination address (DEST IPv6: TEREDO (6.4.5.6.7:1025)) of the data packet (response 311) with the native destination address (DEST IPv4: 1.2.3.4:3544) from that entry.” *Id.*



Huitema, FIG. 3A*

As explained before and highlighted above, relay 202 replaces the *constructed destination address* TEREDO (6.4.5.6.7:1025) in the header of IPv4 response 311 with the *native destination address* 1.2.3.4:3544 to generate IPv4 response 311'. Ex. 1011, 12:37-67; Ex. 1026, ¶ 120. While Huitema does not expressly disclose that relay 202 searches a table associating the IPv6 destination address TEREDO (6.4.5.6.7:1025) with the IPv4 destination address 1.2.3.4:3544, such a table would have been necessary for relay 202 to know the IPv4 destination address from the corresponding IPv6 destination address. Ex. 1026, ¶ 120. That is, the mechanism that relay 202 uses to replace TEREDO (6.4.5.6.7:1025) (*constructed destination address*) with the IPv4

destination address 1.2.3.4:3544 is or discloses *searching an address translation table*, as claimed. *Id.* Otherwise, Huitema's system would not work as described to convert response 311 into response 311'' because relay 202 could not replace TEREDO (6.4.5.6.7:1025) with 1.2.3.4:3544 without a table associating the two addresses. *Id.*

Moreover, Network Address Translators, such as NAT 102 in Huitema, use address translation tables to replace private addresses with public addresses, and vice versa. Ex. 1026, ¶ 121. As Huitema explains, NAT 102 maintains a "mapping" between IPv4 client 101's public and private addresses to accomplish translation. Ex. 1011, 11: 46-50, 12:13-23, 13:19-30, 14: 15-25, 15:35-65, 16:13-28; Ex. 1026, ¶ 121. These mappings maintained by NAT 102 disclose an *address translation table*. Ex. 1026, ¶ 121. And, like relay 202 replaces addresses to modify response 311 to response 311', NAT 102 replaces IPv4 client 101's public IPv4 address 64.5.6.7:1025 in response 311'' with client 101's corresponding private IPv4 address 10.0.0.1:2345 to generate response 311'''. Ex. 1011, 13:19-30; Ex. 1026, ¶ 121. Thus, it would have been apparent to a POSITA that relay 102 also needs to maintain a similar table/mapping to replace constructed IPv6 addresses with IPv4 addresses to convert IPv6 response 211, 311, and 410 in Figures 2, 3A, and 4 respectively into corresponding IPv4 responses 211', 311', and 410' as Huitema teaches. *Id.* For example, as discussed below in Grounds 3 and 4, Kong's 64 Translator performs the same function using an *address translation table* that Kong calls "mapping entries." *Infra* Section V.B.2. Thus, Huitema discloses, suggests, or renders obvious relay 202 "*searching an address translation table ... for an entry associating with said constructed destination address a native destination address*" and, "*if an entry is found, replacing the constructed destination address of the data packet with the native destination address from that entry*" as recited element [5.1] and [5.2]. Ex. 1026, ¶ 121.

3. Dependent Claim 6

Claim 6 depends from independent claim 5 and recites the additional elements quoted below. Huitema renders unpatentable claim 5 for the reasons discussed above. *Supra* Section V.A.2. Moreover, Huitema discloses or suggests the additional elements of claim 6. Ex. 1026, ¶ 122.

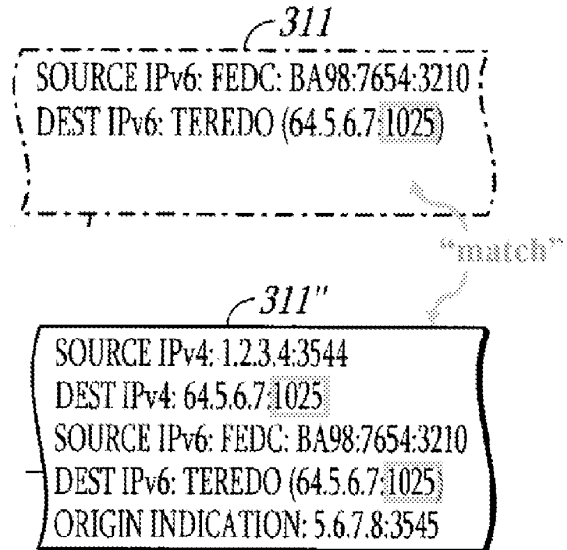
- a. [6.1] "*The method according to claim 5, further comprising a step of verifying that the destination port number of the data packet*

and the destination port number contained in the constructed destination address match.

Claim 6 lacks antecedent basis for “the destination port number of the data packet” because independent claim 1 does not recite that the *data packet* has a destination port number. Rather, claim 1 only recites that the *IPv6 destination address* contains a concatenated port number: “identifying an IPv6 destination address constructed by concatenating ... a destination port number.”

To the extent claim 6 is not indefinite, however, Huitema renders this element obvious. Ex. 1026, ¶ 124. For example, in Figure 3A of Huitema, *the destination port number (1025) of the data packet (response 311’) and the destination port number (1025) contained in the constructed destination address (TEREDO (6.4.5.6.7:1025)) match*. See Ex. 1011, 12:37-13:18, FIG. 3A (excerpt reproduced above on the right); Ex. 1026, ¶ 124.

Moreover, when relay 202 regularizes response 311 into 311’, the port number 1025 for IPv4 client 101 in the constructed IPv6 destination address TEREDO (6.4.5.6.7:1025) of response 311 corresponds to the port number 3545 in the IPv4 destination address of response 311’. Ex. 1011, 12:37-13:3; Ex. 1026, ¶ 124. This is because, for the initial response 311 from IPv6 client 201 to IPv4 client 101, relay uses the port number 3545 of server 103 as the port number for IPv4 client 101 because it does not yet know the port number for IPv4 client 101. Ex. 1011, 12:54-58 (“Since the relay 202 had not previously received any direct messages from IPv4 client 101, the relay 202 sends the response to the server 103 over UDP to the address of server 103 (e.g., 1.2.3.4:3544).”); Ex. 1026, ¶ 125. Thus, from IPv6 response 311 to IPv4 response 311’ the two port numbers *match*, as claimed, because they both correspond to IPv4 client 101. Ex. 1026, ¶ 125.



Moreover, in subsequent responses sent from IPv6 client 201 to IPv4 client 101 *the destination port number (1025) of the data packet and the destination port number (1025) contained in the constructed destination address (TEREDO (6.4.5.6.7:1025)) match*, as claimed. See Ex. 1011, 13:65-14:31, FIG. 3B; Ex. 1026, ¶ 126. This is because, “[o]nce the knowledge of the IPv4 address and port of the relay 202 has been learned, the IPv4 client 101 may *directly* communicate over IPv4 with the relay 202 and need not burden the server 201 any further with communications to the IPv6 client 201.” Ex. 1011, 13:38-41; Ex. 1026, ¶ 126.

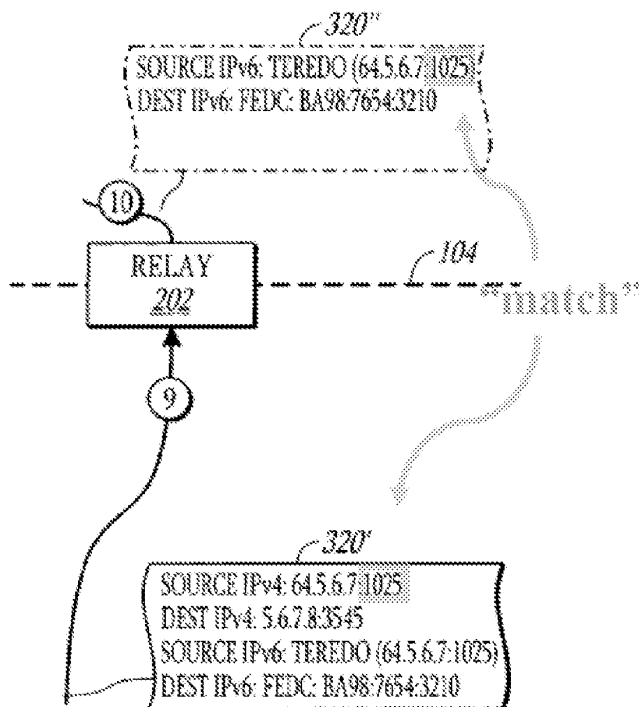


Figure 3B shows message flows in the same embodiment of Figure 3A but “[a]t this stage [where] the IPv4 client 101 has learned of the IPv4 address and port of the relay 202,” so “IPv4 client 101 may communicate with the IPv6 client directly through the relay 202 without the further assistance of the server 103.” Ex. 1011, 13:60-64; *see also* Ex. 1026, ¶ 127. As highlighted in the Figure 3B excerpt above on the right, the *destination port number (1025) of the data packet and the destination port number (1025) contained in the constructed destination address (TEREDO (6.4.5.6.7:1025)) again match*, as claimed. Ex. 1026, ¶ 127.

Separately, Huitema’s Figure 2 method and Figure 4 method respectively disclose that *the destination port number (1025) of the data packet (211’, 410’) and the destination port number (1025) contained in the constructed destination address (TEREDO (6.4.5.6.7:1025)) match*, as claimed. Ex. 1011, 11:10-38, 14:58-67, 15:66-16:10; Ex. 1026, ¶ 128.

To the extent that Huitema does not expressly disclose *verifying* that the port numbers match, this would have been obvious for the reasons explained below in Ground 5. *Infra* Section V.C.2; Ex. 1026, ¶ 129; Ex. 1003, ¶¶ 217-220. In short, it would have been obvious to *verify* the matching/corresponding port numbers to detect and discard potentially fraudulent/spoofed traffic.

Infra Section V.C.2; Ex. 1026, ¶ 129; Ex. 1003, ¶¶ 217-220.

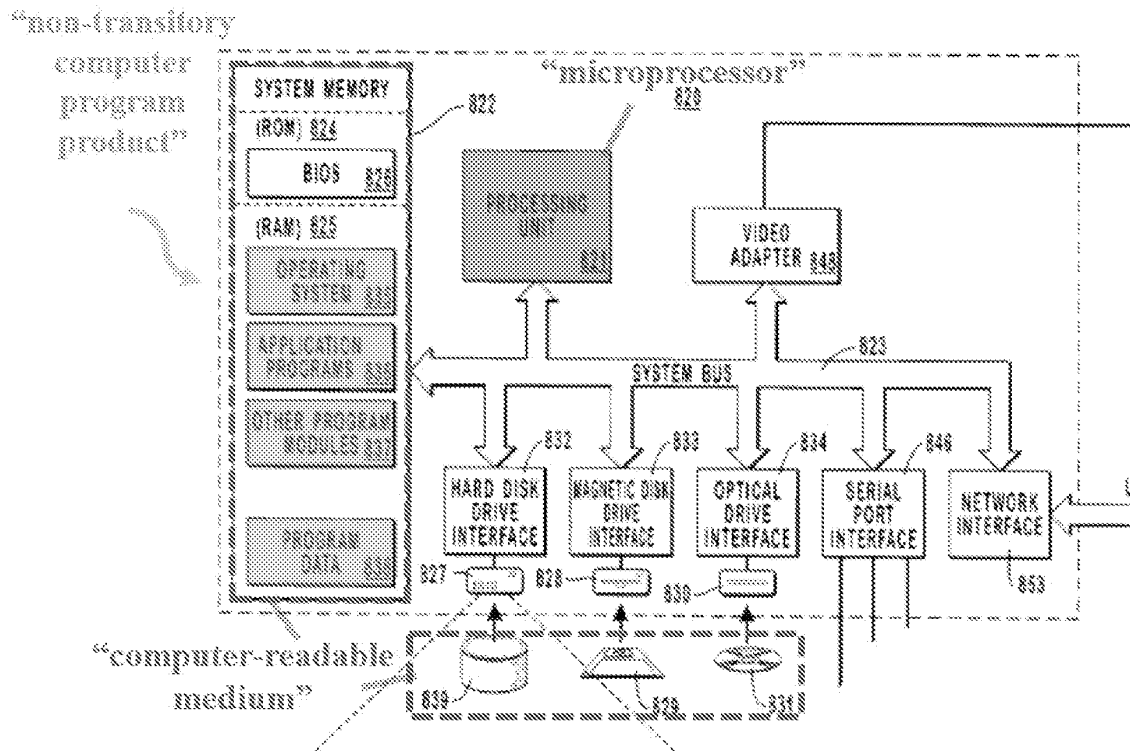
Accordingly, Huitema discloses or suggests “*verifying that the destination port number of the data packet and the destination port number contained in the constructed destination address match,*” as claimed.

4. Dependent Claim 7

- a. [7.1] “*A non-transitory computer program product that is at least one of downloadable from a communications network and stored on a computer-readable medium and executable by a microprocessor, comprising program code instructions for executing the method according to claim 1 of receiving a data packet in an IPv6 domain.*”

Claim 7 depends from independent claim 1 and recites the additional elements quoted above. Huitema renders unpatentable independent claim 1 for the reasons discussed above. *Supra* Section V.A.1. Moreover, Huitema discloses or suggests the additional elements of claim 7. Ex. 1026, ¶ 131.

Specifically, independent claim 7 recites “*a non-transitory computer program product ... stored on a computer-readable medium and executable by a microprocessor comprising program code instructions for executing the method according to claim 1[.]*” As explained above, Huitema’s relay 202—in each message flow of Figures 2, 3A-3B, and 4—performs the method steps of independent claim 1. *Supra* Sections V.A.1. Moreover, Huitema discloses that relay 202 (as well as clients 101, 201; NAT 102; and server 103) “may be structured as ... described with respect to the conventional computer 820 of FIG. 8.” Ex. 1011, 16:36-39.



Huitema, FIG. 8*

As highlighted above, computer 802 (i.e., relay 202) has *a non-transitory computer program product* (e.g., operating system 835, application programs 836, program modules 837, program data 838) *stored on a computer-readable medium* (e.g., system memory 822 including RAM 824 and/or ROM 825, hard disk 839, removable hard disk 829, optical disk 831) *and executable by a microprocessor* (e.g., processing unit 821). See Ex. 1011, 16:39-17:23; Ex. 1026, ¶ 132. Accordingly, and for the reasons discussed above for independent claim 1, relay 202 includes “*a non-transitory computer program product ... stored on a computer-readable medium and executable by a microprocessor comprising program code instructions for executing the method according to claim 1 ...*,” as recited in independent claim 7. *Supra* Section V.A.1; Ex. 1026, ¶ 133.

5. Independent Claim 8

In means-plus-function form, independent claim 8 calls for a device with means for performing the method steps of independent claim 1. For the reasons discussed above for independent claim 1, as well as for the additional reasons below, Huitema renders independent

claim 8 unpatentable. *Supra* Section V.A.1; Ex. 1026, ¶ 134.

a. [8.pre.1] “*A device for receiving an IPv6 data packet in an IPv6 domain connected to an IPv4 domain,*”

This element is identical to element [1.pre.1] of independent claim 1, except that element [8.pre.1] recites a “device” instead of a “method.” To the extent this element is limiting, Huitema’s relay 202 corresponds to the claimed *device*, and relay 202 *receives an IPv6 data packet in an IPv6 domain connected to an IPv4 domain* for the reasons discussed above for element [1.pre.1]. *Supra* Section V.A.1.a; Ex. 1026, ¶ 135.

b. [8.pre.2] “*said packet comprising an IPv6 destination address and an IPv6 source address,*”

This element is identical to element [1.pre.2] of independent claim 1. Accordingly, to the extent the preamble is limiting, Huitema discloses or suggests this element for the reasons discussed above for element [1.pre.2]. *Supra* Section V.A.1.b; Ex. 1026, ¶ 136.

c. [8.pre.3] “*said device for use in a home gateway adapted to connect a user terminal to said IPv6 domain, comprising:*”

This element is virtually identical element [1.pre.3] of independent claim 1. Accordingly, to the extent the preamble is limiting, Huitema discloses or suggests this element for the reasons discussed above for element [1.pre.3]. *Supra* Section V.A.1.c; Ex. 1026, ¶ 137.

d. [8.1] “*means for identifying an IPv6 destination address constructed by concatenating an IPv6 prefix, an IPv4 destination address, and a destination port number;*”

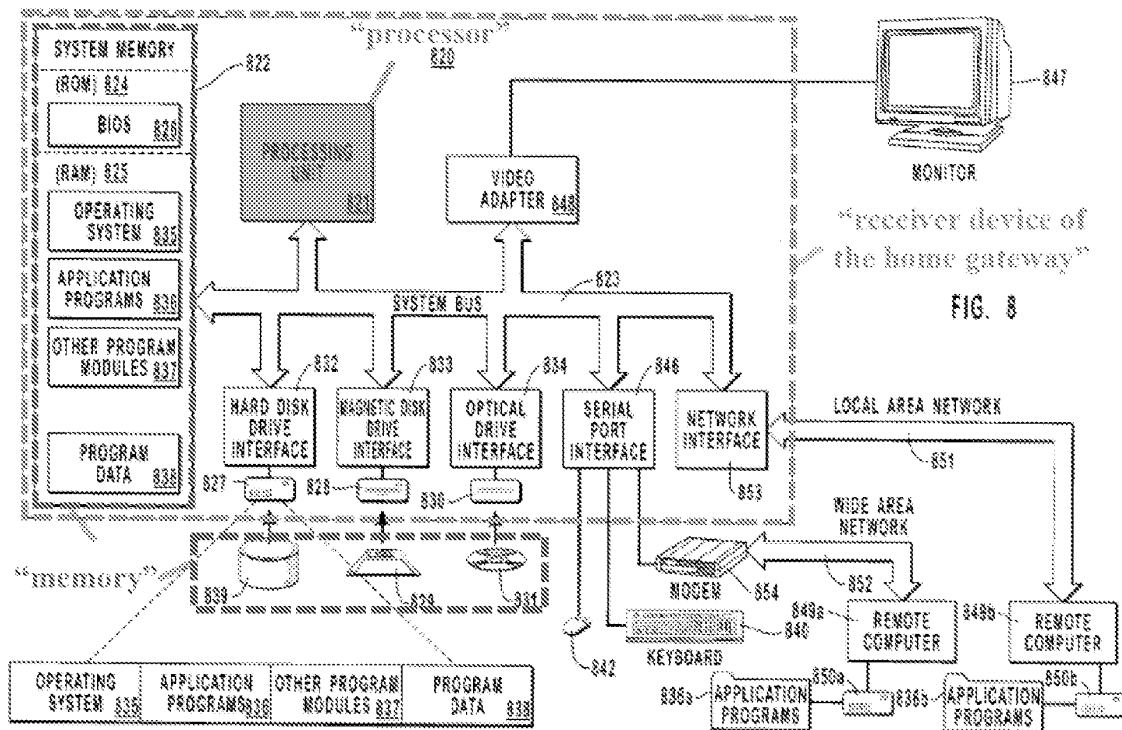
This element recites means for performing a function identical to element [1.1] of independent claim. As explained above, this element invokes the following construction under § 112, 6th Paragraph:

- **Claimed Function:** identifying an IPv6 destination address constructed by concatenating an IPv6 prefix, an IPv4 destination address, and a destination port number; and
- **Corresponding Structure:** a receiver device of the home gateway having a processor and memory, and equivalents

Supra Section I.E.3.a. For the reasons discussed above for element [1.1], Huitema’s relay 202 performs the claimed function. *Supra* Section V.A.1.d; Ex. 1026, ¶ 138.

Moreover, Huitema’s relay 202 satisfies the corresponding structure. Ex. 1026, ¶ 139. As discussed, Huitema’s relay 202 corresponds to the claimed home gateway. *Supra* Section V.A.1.c.

Moreover, Huitema teaches that relay 202 has structure of Figure 8's computer 820, which is a receiver device of the home gateway having a processor and memory:



Huitema, FIG. 8*

As highlighted above, computer 820 corresponds to the *receiver device of the home gateway*, and computer 820 has a processor 821 and memory 822, 825, 825, 829, 831, and/or 839. Ex. 1011, 16:36-17:23; Ex. 1026, ¶ 140.

Accordingly, Huitema discloses “means for identifying an IPv6 destination address constructed by concatenating an IPv6 prefix, an IPv4 destination address, and a destination port number,” as claimed.

- e. [8.2-3] “means for
- [8.2.1] *regularizing, if necessary, at least one of the IPv6 source address and the IPv6 destination address of the data packet by replacing said at least one IPv6 address,*
- [8.2.2] *the replacement process belonging to a group comprising replacement of a constructed address with a native address and replacement of a native address with a constructed address, and*
- [8.2.3] *modifying the data packet using the regularized address; and”*

Element [8.2-3] recites means for performing functions [8.2.1], [8.2.2], and [8.2.3] respectively identical to elements [1.2.1], [1.2.2], and [1.3]. As explained above, elements [8.2-3] invokes the following construction under § 112, 6th Paragraph:

- **Claimed Functions:**
 - [8.2.1] if necessary, (1) regularizing the IPv6 source address by replacing the IPv6 source address or (2) regularizing the IPv6 destination address by replacing the IPv6 destination address;¹²
 - [8.2.2] the replacement process belonging to a group comprising replacement of a constructed address with a native address and replacement of a native address with a constructed address; and
 - [8.2.3] modifying the data packet using the regularized address.
- **Corresponding Structure:** the device of the home gateway having a processor and a memory, where the memory stores a table for translating between addresses, and equivalents.

Supra Section I.E.3.b. For the reasons discussed above for elements [1.2.1], [1.2.2], and [1.3], Huitema’s relay 202 performs the claimed functions [8.2.1], [8.2.2], and [8.2.3], respectively. *Supra* Sections V.A.1.e-g.

Moreover, Huitema’s relay 202 satisfies the corresponding structure. Ex. 1026, ¶ 143. As explained above for element [8.1], relay 202 is the device of the home gateway having a processor and memory. *Supra* Section V.A.5.d. Additionally, as explained above for dependent claim 5,

¹² Function [8.2.1] incorporates the proposed construction for element [1.2.1] discussed above. *Supra* Section I.E.2.

Huitema discloses or suggests that relay 202 has a table for translating between addresses (i.e., converting between IPv6 and IPv4 addresses, and vice versa). *Supra* Section V.A.2; Ex. 1026, ¶ 143.

Accordingly, Huitema discloses “*means for regularizing, if necessary, at least one of the IPv6 source address and the IPv6 destination address of the data packet by replacing said at least one IPv6 address, the replacement process belonging to a group comprising replacement of a constructed address with a native address and replacement of a native address with a constructed address, and modifying the data packet using the regularized address,*” as claimed.

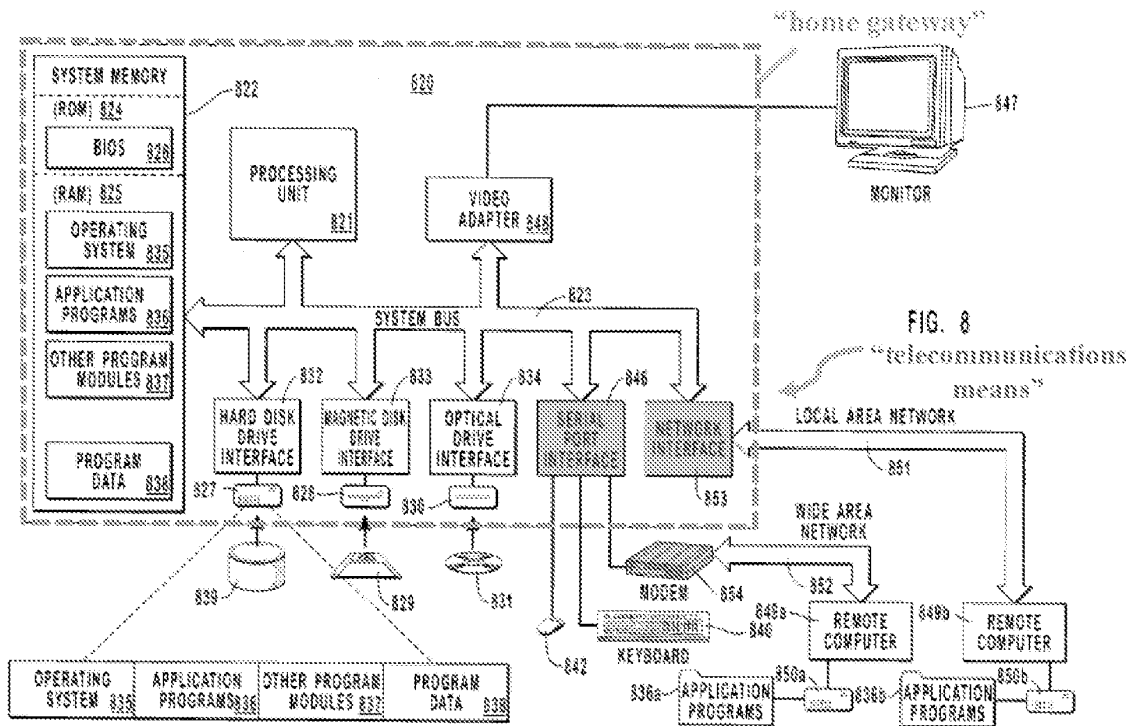
f. [1.4] “*means for routing the IPv6 data packet to its destination.*”

This element recites means for performing a function identical to element [1.4] of independent claim. As explained above, this element invokes the following construction under § 112, 6th Paragraph:

- **Claimed Function:** routing the IPv6 data packet to its destination.
- **Corresponding Structure:** telecommunications means of the device of the home gateway, and equivalents

Supra Section I.E.3.c. For the reasons discussed above for element [1.4], Huitema’s relay 202 performs the claimed function. *Supra* Section V.A.1.h.

Moreover, Huitema’s relay 202 satisfies the corresponding structure. Ex. 1026, ¶ 146. As discussed, Huitema’s relay 202 discloses the claimed home gateway. *Supra* Section V.A.5.c. Huitema teaches that relay 202 has structure of Figure 8’s computer 820, which includes *telecommunications means*:



Huitema, FIG. 8*

As highlighted above, computer 820 corresponds to the *device of the home gateway*, and at least serial port interface 846 and modem 854 for communicating on wide area network 852 and/or network interface 853 for communicating on local area network 851 correspond to the *telecommunications means*. Ex. 1011, 17:13-48, FIG. 8; Ex. 1026, ¶ 147.

Accordingly, Huitema discloses “*means for routing the IPv6 data packet to its destination,*” as claimed.

B. Grounds 3 and 4: Kong Anticipates and/or Would Have Rendered Obvious Claims 1 and 5-8

As explained below on an element-by-element basis, Kong anticipates and/or would have rendered obvious claims 1 and 5-8 of the '845 Patent under 35 U.S.C. § 102(a) (pre-AIA).

1. Independent Claim 1

- a. [1.pre.1] “*A method for receiving an IPv6 data packet in an IPv6 domain connected to an IPv4 domain,*”

To the extent the preamble is limiting, Kong discloses this element. Ex. 1026, ¶ 150. Specifically, Kong discloses *a method* (Connection Establishment Phase of Figure 2) *for receiving*

an IPv6 data packet ([ConnectIPv4Server]₆) in an IPv6 domain (IPv6 Network) connected to an IPv4 domain (IPv4 Network):

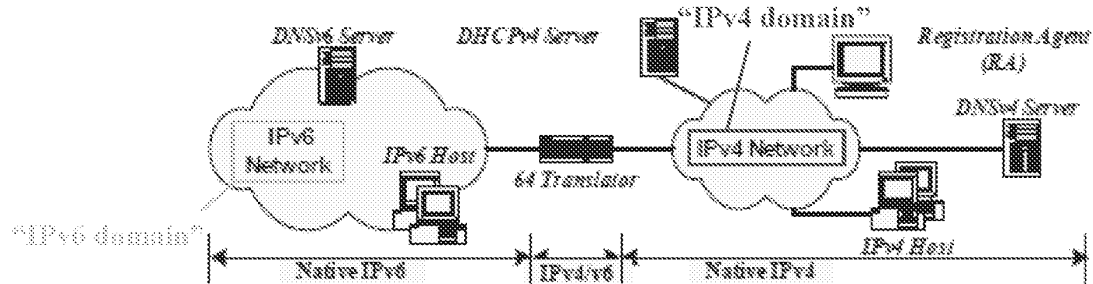


Fig. 1. The overall system architecture using 64TSD

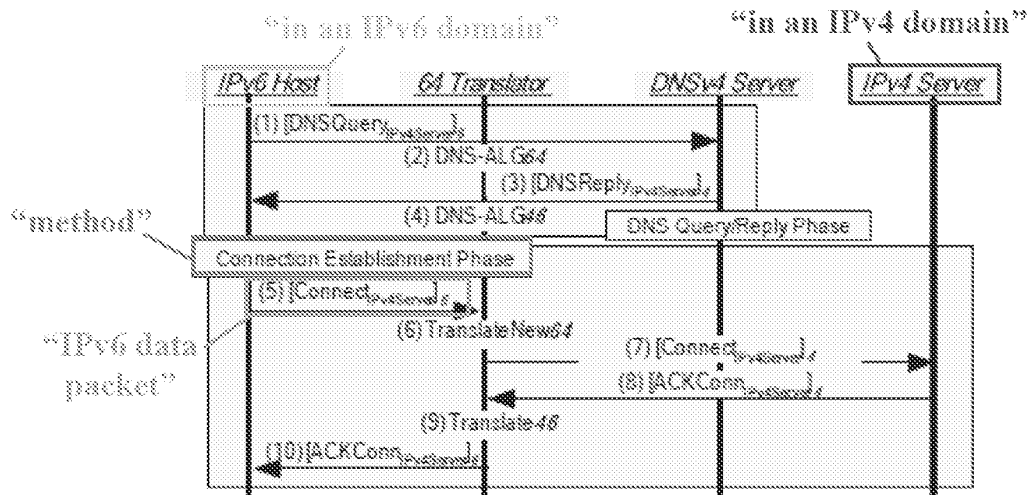


Fig. 2. The procedure of the outbound session

Kong, FIGS. 1, 2*

As highlighted in Figure 1 above, an IPv6 host and server in an *IPv6 domain* (IPv6 network) connects to an IPv4 host and server in an *IPv4 domain* (IPv4 network) via a 64 Translator. Ex. 1020, 498 (“The overall system architecture using 64TSD consists of IPv6 Network, IPv4 Network and the 64 Translator”); *see also id.*, 499. The 64 Translator connects the IPv6 network to the IPv4 network. Ex. 1020, 498-499, 501; Ex. 1026, ¶ 151. As explained above, the claimed *domain* refers to a “network.” *Supra* Section I.E.1. Kong’s IPv6 and IPv4 networks satisfy this construction. Ex. 1026, ¶ 151.

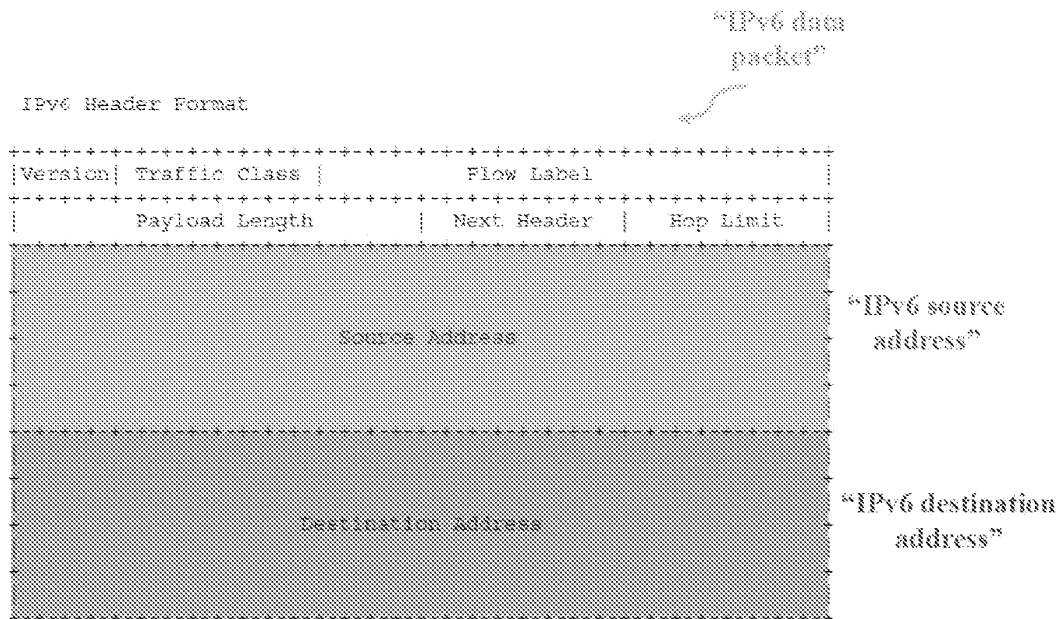
Also as highlighted above in Figure 2, Kong’s “Connection Establishment Phase,” by

which an IPv6 Host establishes communication with an IPv4 server, corresponds to the claimed *method*. Ex. 1020, 501; Ex. 1026, ¶ 152. Kong shows the 64 Translator receiving a [ConnectIPv4Server]₆ packet from the IPv6 Host in the IPv6 network. Ex. 1020, 501. The [ConnectIPv4Server]₆ packet is an IPv6 packet because, as Kong explains, “[Connect_H]_x ... is a packet to connect to host H using IPv_x” and, here, *X*=6. Ex. 1020, 500; Ex. 1026, ¶ 152. The [ConnectIPv4Server]₆ packet discloses the claimed *IPv6 data packet*. Ex. 1026, ¶ 152.

Requester notes that Kong also discloses the claimed *IPv4 domain*, even if improperly interpreted to mean a part of a computer network that uses the IPv4 protocol and in which the data processing resources are under common control. *Supra* Section. I.E.1; Ex. 1026, ¶ 153. For example, when Kong built a working experimental prototype of its system, Kong implemented the “IPv4 Host ... on an 866MHz Mobile Pentium III with Windows XP” and connected the “IPv4 Host ... to a 100Mbps Ethernet.” Ex. 1020, 503; Ex. 1026, ¶ 153. Thus, Kong’s IPv4 domain is a part of a computer network that uses the IPv4 protocol and in which the data processing resources are under common control—control by the authors of Kong who built the system. *Id.* Further, Kong’s IPv4 domain is supported by, and under the control of, a common DHCPv4 server, registration agent (RA), and DNSv4 server. *See* Ex. 1020, 498-500, FIG. 1; Ex. 1026, ¶ 153. Accordingly, Kong discloses or suggests “*a method for receiving an IPv6 data packet in an IPv6 domain connected to an IPv4 domain,*” as claimed.

b. [1.pre.2] “said packet comprising an IPv6 destination address and an IPv6 source address,”

Kong discloses or suggests this element. Ex. 1026, ¶¶ 155-157. In Kong, *said packet* ([ConnectIPv4Server]₆) *comprising an IPv6 destination address and an IPv6 source address*:



Ex. 1029, 4*

The Figure above shows the required format for the header of an IPv6 packet to comply with IP Version 6 protocol. Ex. 1029, 3-4; Ex. 1026, ¶156. As highlighted, the header of an IPv6 packet necessarily includes *an IPv6 destination address and an IPv6 source address*, as claimed. Ex. 1029, 3-4; Ex. 1026, ¶ 156. And because Kong’s [ConnectIPv4Server]₆ packet is an IPv6 packet, it necessarily includes *an IPv6 destination address and an IPv6 source address*, as claimed. Ex. 1029, 3-4; Ex. 1026, ¶ 156. Alternatively, a POSITA would have found it obvious to include *an IPv6 destination address and an IPv6 source address* in the header of Kong’s [ConnectIPv4Server]₆ packet to comply with the requirements of the IPv6 specification and so that Kong’s system worked as described. Ex. 1026, ¶ 156. Indeed, without a source and destination address, communication would be impossible because [ConnectIPv4Server]₆ packet could not propagate across the IPv6 network from the IPv6 Host to the 64 Translator. *Id.*

Accordingly, Kong discloses or suggests “*said packet comprising an IPv6 destination address and an IPv6 source address*,” as claimed.

- c. [1.pre.3] “*said method being executed in a home gateway adapted to connect a user terminal to said IPv6 domain, and comprising the following steps:*”

Kong discloses or suggests this element. Ex. 1026, ¶¶ 158-161. Specifically, Kong

discloses *said method* (Connection Establishment Phase of Figure 2) *being executed in a home gateway (64 Translator) adapted to connect a user terminal (IPv4 server/host of Figure 1) to said IPv6 domain* (IPv6 Network of Figure 1):

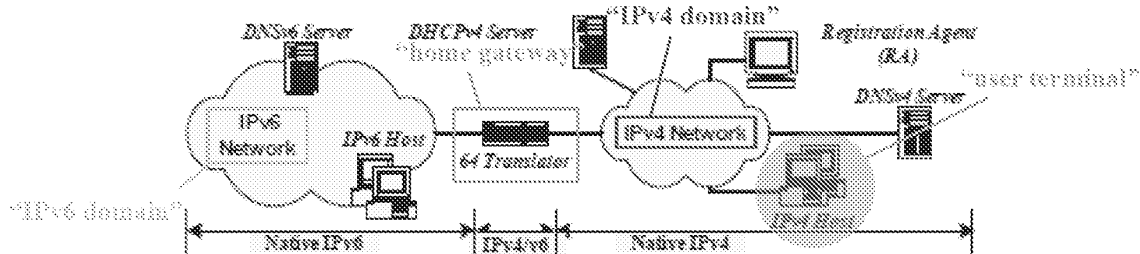


Fig. 1. The overall system architecture using 64TSD

Kong, FIG. 1*

Kong's 64 Translator discloses the claimed *home gateway*. Ex. 1020, 496, 499-501; Ex. 1026, ¶ 159. The '845 Patent describes the home gateway broadly as "any equipment" connecting a home network to another network. *See* Ex. 1001, 1:45-58. Here, Kong's 64 Translator connects the IPv4 host on the IPv4 network to the IPv6 network. Ex. 1020, 496, 499-501; Ex. 1026, ¶ 159. The 64 Translator also performs various functions to allow the IPv4 host to communicate with the IPv6 host, and vice versa, which the two devices could not otherwise do:

Table 1. The functions of the 64 Translator

Function	Description
Address/port mapping	It allocates a port number for each outbound session. It also maintains the mapping entries. ¹
Protocol translation	It translates a protocol header between IPv6 and IPv4 as well as ICMPv6 and ICMPv4. It functions checksum recalculation and TCP/UDP port number translation by NAT algorithm.
DHCP client module	It allocates IPv4 address dynamically from DHCP server instead of IPv6 servers. It registers RegInfo ² of each IPv6 hosts to the RA.

Kong, 499 (Table 1)

Thus, the 64 Translator serves as a home gateway to the IPv4 host on the IPv4 network to access the IPv6 network. Ex. 1020, 496, 499-501; Ex. 1026, ¶ 159.

Moreover, Kong's IPv4 host discloses the claimed *user terminal*. Ex. 1026, ¶ 160. In Figure 1, Kong shows the IPv4 host as a personal desktop computer, which is a user terminal. Ex. 1020, 498; Ex. 1026, ¶ 160. Additionally, Kong's 64 Translator (*home gateway*) connects the IPv4 host (*user terminal*) to the IPv6 network (*IPv6 domain*), as recited in the preamble. Ex. 1020, 496, 499-501; Ex. 1026, ¶ 160. Alternatively, Kong's IPv6 host may correspond to the claimed *user terminal*, as Kong also shows this component as a personal desktop computer and the IPv6 host connects to the IPv6 network with the 64 Translator. Ex. 1020, 498; Ex. 1026, ¶ 160.

Accordingly, Kong discloses or suggests "*said method being executed in a home gateway adapted to connect a user terminal to said IPv6 domain*," as claimed.

d. [1.1] "identifying an IPv6 destination address constructed by concatenating an IPv6 prefix, an IPv4 destination address and a destination port number;"

Kong discloses or suggests this element. Ex. 1026, ¶ 162. Kong discloses the 64 Translator identifying an IPv6 destination address constructed by concatenating an IPv6 prefix (dummy prefix), an IPv4 destination address (<IPv4 address ... >) and a destination port number (<... , IPv4 Port>):

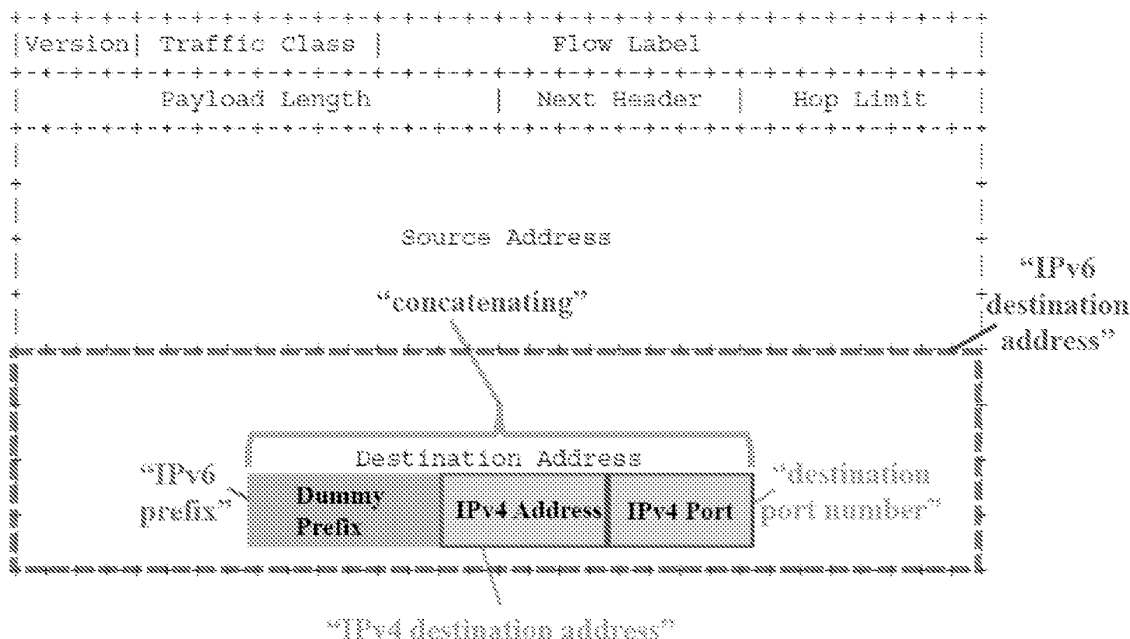


Figure: Kong's IPv6 Destination Address*

- *An IPv6 destination address constructed by concatenating an IPv6 prefix, an IPv4 destination address and a destination port number an IPv4 destination address and a destination port number*

Kong discloses that the [ConnectIPv6Server]₆ (*IPv6 data packet*) has an *IPv6 destination address* in the form shown above—i.e., *constructed by concatenating an IPv6 prefix* (dummy prefix), *an IPv4 destination address and a destination port number* (<IPv4 address, IPv4 Port>). Ex. 1026, ¶ 163. As explained in the previous section, the [ConnectIPv6Server]₆ necessarily has an IPv6 destination address and/or an IPv6 destination address would have been obvious. *Supra* Section 163.

Moreover, Figure 3 of Kong shows “[t]he procedure of the inbound session” in which the IPv4 host connects to the IPv6 network—i.e., communication in the opposition direction of Figure 2’s outbound session discussed above. Ex. 1020, 501-502, FIG. 3. In step (9) of the outbound session, the “64 Translator translates the [ConnectIPv6Server]₄ to [ConnectIPv6Server]₆ using *the IPv6 address that is created by the concatenation of dummy prefix and IPv4 address.*” Ex. 1020, 502, FIG. 3. This means that IPv6 addresses in Kong—including the IPv6 destination address of the [ConnectIPv6Server]₆ packet relied on as the claimed *IPv6 data packet*—are *constructed by concatenating* a dummy prefix (*IPv6 prefix*) and an IPv4 address (*IPv4 destination address*). Ex. 1020, 502, FIG. 3; Ex. 1026, ¶ 164. Thus, Kong discloses or suggest that the required *IPv6 destination address* in the [ConnectIPv4Server]₆ packet is *constructed by concatenating* a dummy prefix (*IPv6 prefix*) and an IPv4 address (*IPv4 destination address*), as claimed. Ex. 1020, 502, FIG. 3; Ex. 1026, ¶ 164. Alternatively or additionally, by teaching that the [ConnectIPv4Server]₆ packet in the Figure 3 outbound session has the claimed structure, Kong at least renders obvious that the same [ConnectIPv4Server]₆ packet in the Figure 2 inbound session has this same structure. Ex. 1026, ¶ 164.

Requester notes that the dummy prefix of Kong is an *IPv6 prefix*, as claimed. Ex. 1026, ¶ 165. First, the dummy prefix qualifies as an *IPv6 prefix* by virtue of being a prefix in an IPv6 destination address. *Id.* Second, Kong teaches that “6TO4” translation, like the translation performed by 64 Translator in Figure 2, “uses the special *16-bit IPv6 address prefix* for 6TO4 defined by IANA (Internet Assigned Numbers Authority).” Ex. 1020, 497; *see also* Ex. 1026, ¶ 165. Thus, a POSITA would have understood the dummy prefix in the IPv6 destination address

of the [ConnectIPv4Server]₆ packet to be an *IPv6 prefix*, as claimed.

Moreover, Kong makes clear that its IPv4 addresses are complete IPv4 addresses—including a port number. Ex. 1026, ¶ 166. Thus, when Kong explains that an IPv6 address embodies a “concatenation of dummy prefix and IPv4 address”—the “IPv4” address within that concatenation has the form “<IPv4 Address, IPv4 Port>” and thus includes the claimed *destination port number*. Ex. 1020, 502; Ex. 1026, ¶ 166. For example, in step (8) of the Figure 2 outbound session, Kong explains that, “[w]hen the 64 Translator receives the [ACKConnIPv4Server]₄, it looks up the mapping entries with <IPv4 address, *IPv4 Port*>.” Ex. 1020, 501. This means that Kong’s system uses complete IPv4 addresses that include a port number. Ex. 1026, ¶ 166. Thus, Kong’s destination “IPv6 address ... created by the concatenation of dummy prefix and IPv4 address” includes a *destination port number*, as claimed, as part of the IPv4 destination address. Ex. 1020, 502; Ex. 1026, ¶ 166. Indeed, as Dr. Min explains, “IPv4 addresses generally have the form <IP address>:<port number>.” Ex. 1026, ¶ 166.

Further confirming that IPv4 addresses in Kong include port numbers—and thus Kong’s constructed IPv6 destination address includes a concatenated *destination port number*—Table 1 of Kong shows the functions of the 64 Translator include allocating a *port number* for each outbound session and “TCP/UDP *port number* translation.” Ex. 1020, 499; *see also* Ex. 1026, ¶ 167.

Table 1. The functions of the 64 Translator

Function	Description
Address/port mapping	It allocates a port number for each outbound session. It also maintains the mapping entries. ³
Protocol translation	It translates a protocol header between IPv6 and IPv4 as well as ICMPv6 and ICMPv4. It functions checksum recalculation and TCP/UDP port number translation by NAT algorithm.
DHCP client module	It allocates IPv4 address dynamically from DHCP server instead of IPv6 servers. It registers <i>RegInfo</i> ² of each IPv6 hosts to the RA.

Kong, 499 (Table 1)*

Kong also explains that the 64 Translator maintains “mapping entries” that in the form “<IPv4Addr, IPv6Addr, *IPv4Port*, IPv6Port, TCPTimer, UDPTimer>, ... where IPvXAddr and *IPvXPort* are IPvX address and *port number* of IPvX packet (X is 4 or 6)[.]” Ex. 1020, 499; Ex. 1026, ¶ 168. And, once a session is established, “all outbound and inbound packets can translate

with the *address/port number* using the mapping entries.” Ex. 1020, 502; *see also* Ex. 1026, ¶ 168. Thus, not only does Kong make clear that its system uses IPv4 addresses including port numbers, the 64 Translator needs port numbers to perform its functions, including creating and using mapping entries to route packets. *Id.* Thus, Kong discloses or suggests that the IPv6 destination address in the [ConnectIPv6Server]₆ packet (*IPv6 data packet*) further includes a concatenated *destination port number*. *Id.* And, to the extent Patent Owner contends that Kong does not disclose or suggest this, it would have been obvious to modify the IPv4 address in the [ConnectIPv6Server]₆ packet to further include the destination port number so that Kong’s system would work as Kong describes for its intended purpose. *Id.*

- ***Identifying an IPv6 destination address***

Kong further discloses *identifying* the IPv6 destination address in the [ConnectIPv6Server]₆ packet. Ex. 1026, ¶ 169. For example, Kong explains that “[t]he 64 Translator creates the mapping entry and translates the [ConnectIPv4Server]₆ into the [ConnectIPv4Server]₄ using the mapping entry.” Ex. 1020, 501, FIG. 2; *see also* Ex. 1026, ¶ 169. And Kong explains that “[th]e mapping entry is a tuple of <IPv4Addr, IPv6Addr, IPv4Port, IPv6Port, TCPTimer, UDPTimer > where IPvXAddr and IPvXPort are IPvX address and port number of IPvX packet (X is 4 or 6[]).” Ex. 1020, 499, n. 1. Thus, the 64 Translator *identifies* the IPv6 destination address at least by (1) recording the IPv6 destination address in the mapping entry; and (2) using the mapping entry to translate the [ConnectIPv4Server]₆ packet into the [ConnectIPv4Server]₄ packet. Ex. 1026, ¶ 169.

Accordingly, Kong discloses “*identifying an IPv6 destination address constructed by concatenating an IPv6 prefix, an IPv4 destination address and a destination port number,*” as claimed.

- e. ***[1.2.1] “if necessary, regularizing at least one of the IPv6 source address and the IPv6 destination address of the data packet by replacing said at least one IPv6 address,”***

Kong discloses or suggest this element. Ex. 1026, ¶¶ 171-176. Kong discloses the 64 Translator, *if necessary* (if communicating between IPv6 server and Ipv4 host), *regularizing* (translating from Ipv6 to Ipv4) *at least one of the Ipv6 source address and the Ipv6 destination address of the data packet* (addresses in [ConnectIPv4Server]₆) *by replacing* (replacing with address in [ConnectIPv4Server]₄) *said at least one Ipv6 address*:

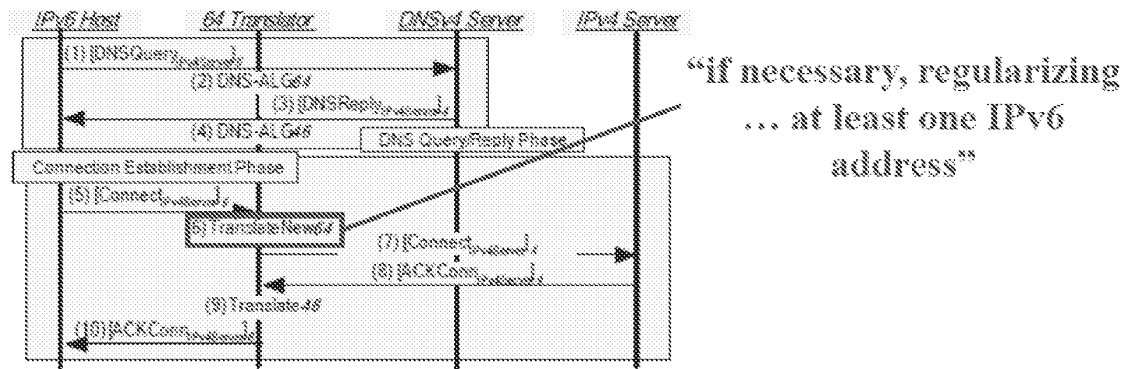


Fig. 2. The procedure of the outbound session

Kong, FIG. 2*

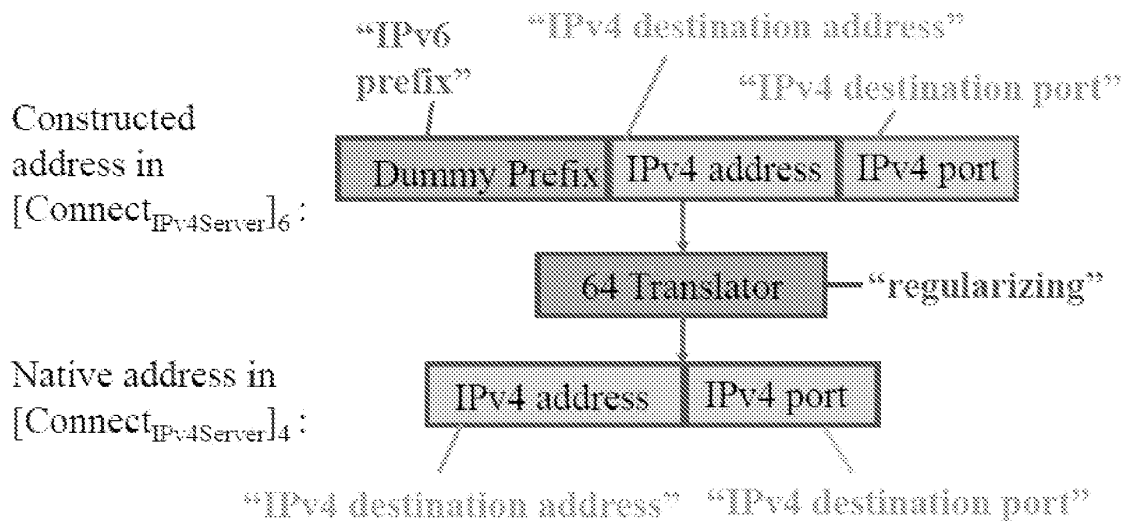


Figure*: Kong's *regularizing*

As explained above, this claim element means (1) regularizing the Ipv6 source address by replacing the Ipv6 source address or (2) regularizing the Ipv6 destination address by replacing the Ipv6 destination address. *Supra* Section I.E.2. Kong does both. Ex. 1020, 501. Specifically, in step (6) of Figure 3, "[t]he 64 Translator creates the mapping entry and translates the [Connect]_{IPv4Server}6 into the [Connect]_{IPv4Server}4 using the mapping entry." Ex. 1020, 501; Ex. 1026, ¶ 172. The [Connect]_{IPv4Server}4 packet resulting from the translation is an IPv4 packet because "[ConnectH]x

... is a packet to connect to host H using IPv X ” where $X=4$. Ex. 1020, 500; Ex. 1026, ¶ 172.

In Table 1, Kong explains that the translation function of the 64 Translator involves, among other things, “translating a protocol header between IPv6 and IPv4.” Ex. 1020, 499. And, as explained above for element [1.pre.2], IPv6 protocol requires IPv6 packets—including the [ConnectIPv4Server]₆ packet—to contain IPv6 source and destination addresses in the IPv6 protocol header. Ex. 1029, 3-4; Ex. 1026, ¶ 173; *supra* Section V.B.1.b. Thus, a POSITA would understand that, to translate (*regularize*) the [ConnectIPv4Server]₆ packet into the [ConnectIPv4Server]₄ packet, the 64 Translator *replaces* the IPv6 source and destination addresses in the header of the [ConnectIPv4Server]₆ packet with respective IPv4 source and destination addresses in the header of the resulting [ConnectIPv4Server]₄ packet. Ex. 1026, ¶ 173.

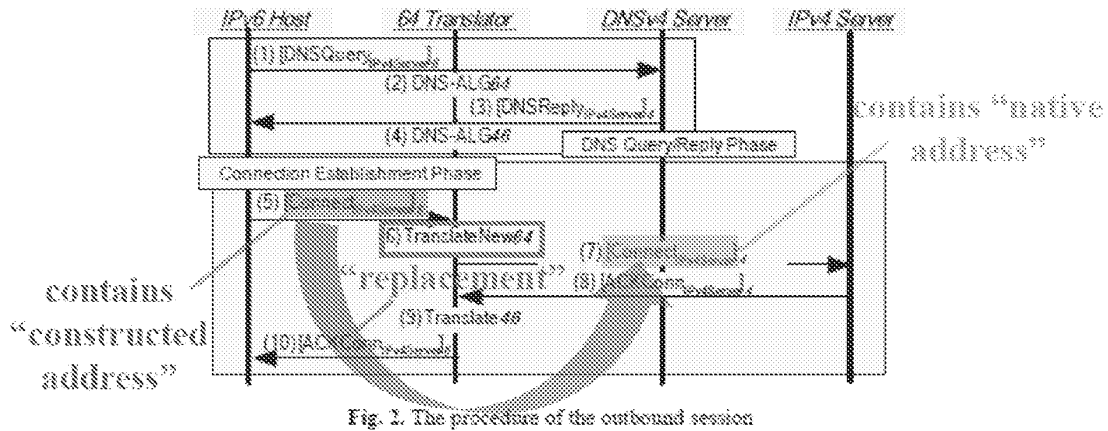
Accordingly, Kong teaches *regularizing* a [ConnectIPv4Server]₆ packet into a [ConnectIPv4Server]₄ packet in the same way the ’845 Patent teaches *regularizing* the IPv6 packet into an IPv4—by translating the packet from IPv6 to IPv4 by replacing the constructed IPv6 address. Ex. 1026, ¶ 174. For example, the specification describes receiving an IPv6 packet, “extract[ing] an IPv4 destination address and a destination port from the IPv6 destination address,” and “then translat[ing] the IPv6 packet into an IPv4 packet” having the extracted IPv4 destination address and port number. Ex. 1001, 14:15-46; *see also* Ex. 1026, ¶ 174; Ex. 1001, 16:23-45 (“If a constructed destination address is identified,” then “the step of regularizing ... replaces the constructed destination address in the IPv6 data packet with this native address”). With respect to the *if necessary* aspect of this claim element, Kong explains that translation is necessary when “communicating between IPv6 networks and IPv4 networks.” Ex. 1020, 498. Thus, sending a [ConnectIPv4Server]₆ packet from the IPv6 network to the IPv4 network in Kong always *necessitates* at 6TO4 translation. Ex. 1026, ¶ 174. Thus, Kong discloses or suggests translating a [ConnectIPv4Server]₆ packet into a [ConnectIPv4Server]₄ packet *if necessary*, as claimed. *Id.*

Accordingly, Kong discloses “*if necessary, regularizing at least one of the IPv6 source address and the IPv6 destination address of the data packet by replacing said at least one IPv6 address,*” as claimed.

- f. [1.2.2] “*the replacement process belonging to a group comprising replacement of a constructed address with a native address and replacement of a native address with a constructed address, and*”

Kong discloses or suggest this element. Ex. 1026, ¶¶ 177-180. Specifically, Kong discloses

the replacement process belonging to a group comprising replacement of a constructed address (IPv6 address in [ConnectIPv4Server]₆) with a native address (IPv4 address in [ConnectIPv4Server]₄) and replacement of a native address with a constructed address:



Kong, FIG. 2*

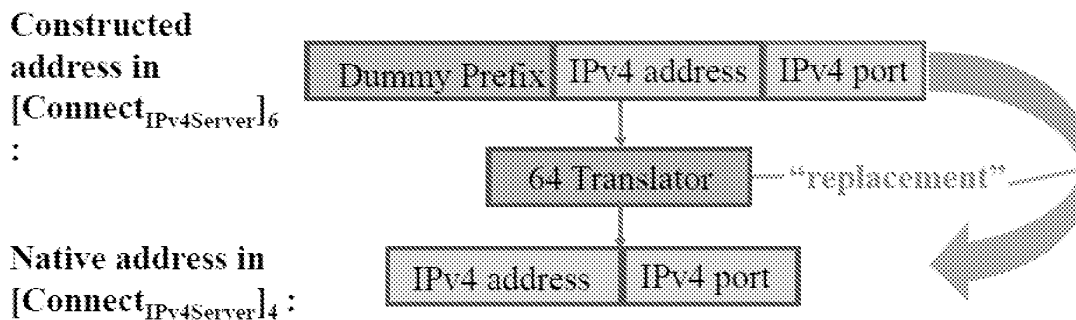


Figure*: Kong's "replacement"

As explained above for element [1.2.1], in step (6) of Figure 2, Kong's "64 Translator creates the mapping entry and translates the [ConnectIPv4Server]₆ into the [ConnectIPv4Server]₄ using the mapping entry," translating the Connect packet from IPv6 to IPv4. Ex. 1020, 501; Ex. 1026, ¶ 178; *Supra* Section V.B.1.e. And, in this process, the 64 Translator replaces the IPv6 destination address with a corresponding IPv4 destination address. Ex. 1020, 501; Ex. 1026, ¶ 178; *supra* Section V.B.1.e.

Here, the IPv6 destination address in the [ConnectIPv4Server]₆ packet corresponds to the claimed *constructed address* because, as Kong explains, an IPv6 address “is *created by the concatenation* of dummy prefix and IPv4 address.” Ex. 1020, 502; Ex. 1026, ¶ 179. On the other hand, the IPv4 destination address in the [ConnectIPv4Server]₄ packet, which replaces the IPv6 destination address in [ConnectIPv4Server]₆ packet, corresponds to the claimed *native address*. Ex. 1026, ¶ 179. This is because IPv4 addresses are native to the IPv4 network—the ultimate destination of the Connect packet. *Id.*

Accordingly, Kong discloses “*the replacement process belonging to a group comprising replacement of a constructed address with a native address and replacement of a native address with a constructed address,*” as claimed.

g. [1.3] “*modifying the data packet using the regularized address; and*”

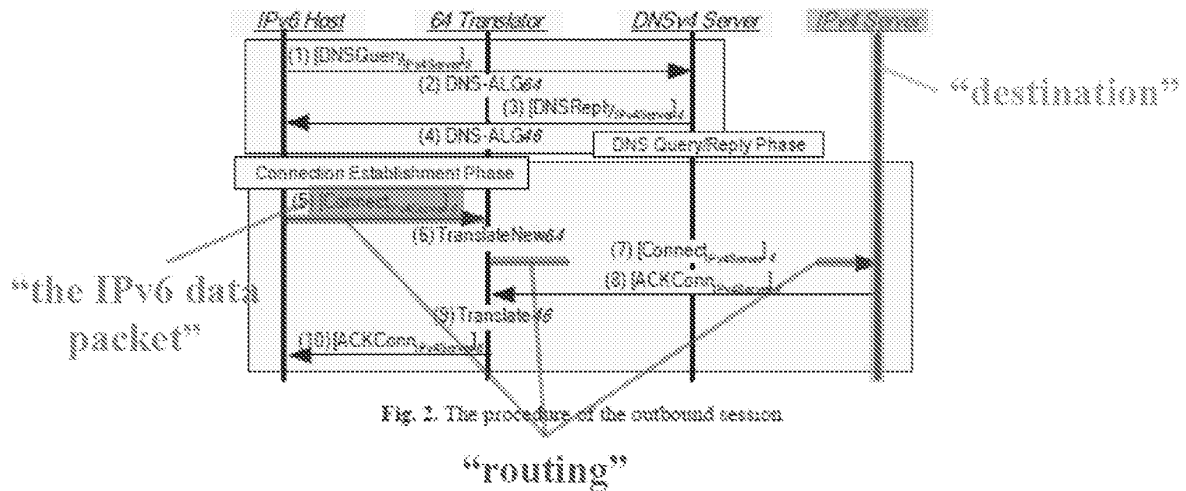
Kong discloses or suggests this element. Ex. 1026, ¶¶ 181-183. Specifically, Kong discloses *modifying the data packet* ([ConnectIPv4Server]₆ packet) *using the regularized address* (IPv4 address in the [ConnectIPv4Server]₄ packet). *Id.*

As explained above for element [1.2.1], Kong’s 64 Translator *modifies* the [ConnectIPv4Server]₆ packet into the [ConnectIPv4Server]₄ packet by replacing the IPv6 source and destination addresses in the header of the [ConnectIPv4Server]₆ packet with respective IPv4 source and destination addresses in the header of the resulting [ConnectIPv4Server]₄ the packet. *See* Ex. 1020, 499, 501, FIG. 2; Ex. 1029, 3-4; Ex. 1026, ¶ 182; *supra* Section V.B.1.e. Here, the IPv4 destination address corresponds to the *regularized address* because it replaces the IPv6 destination address in the *regularization* (translation) process. Ex. 1026, ¶ 183.

Accordingly, Kong discloses “*modifying the data packet using the regularized address,*” as claimed.

h. [1.4] “*routing the IPv6 data packet to its destination.*”

Kong discloses or suggests this element. Ex. 1026, ¶¶ 182-186. Specifically, Kong discloses *routing the IPv6 data packet* (the [ConnectIPv4Server]₆ packet regularized to the [ConnectIPv4Server]₄ packet) *to its destination* (IPv4 Server):



Kong, Fig 2*

In the Figure 2 procedure, Kong explains that “IPv6 Host sends the [ConnectIPv4Server]₆ through the 64 Translator to the IPv4 Server.” Ex. 1020, 501; *see also* Ex. 1026, ¶ 185. Specifically, the IPv6 host sends the [ConnectIPv4Server]₆ packet to the 64 Translator, the 64 Translator “(6) translates the [ConnectIPv4Server]₆ into the [ConnectIPv4Server]₄ using the mapping entry,” and then the 64 Translator “sends the translated packet to IPv4 Server.” Ex. 1020, 501; *see also* Ex. 1026, ¶ 185.

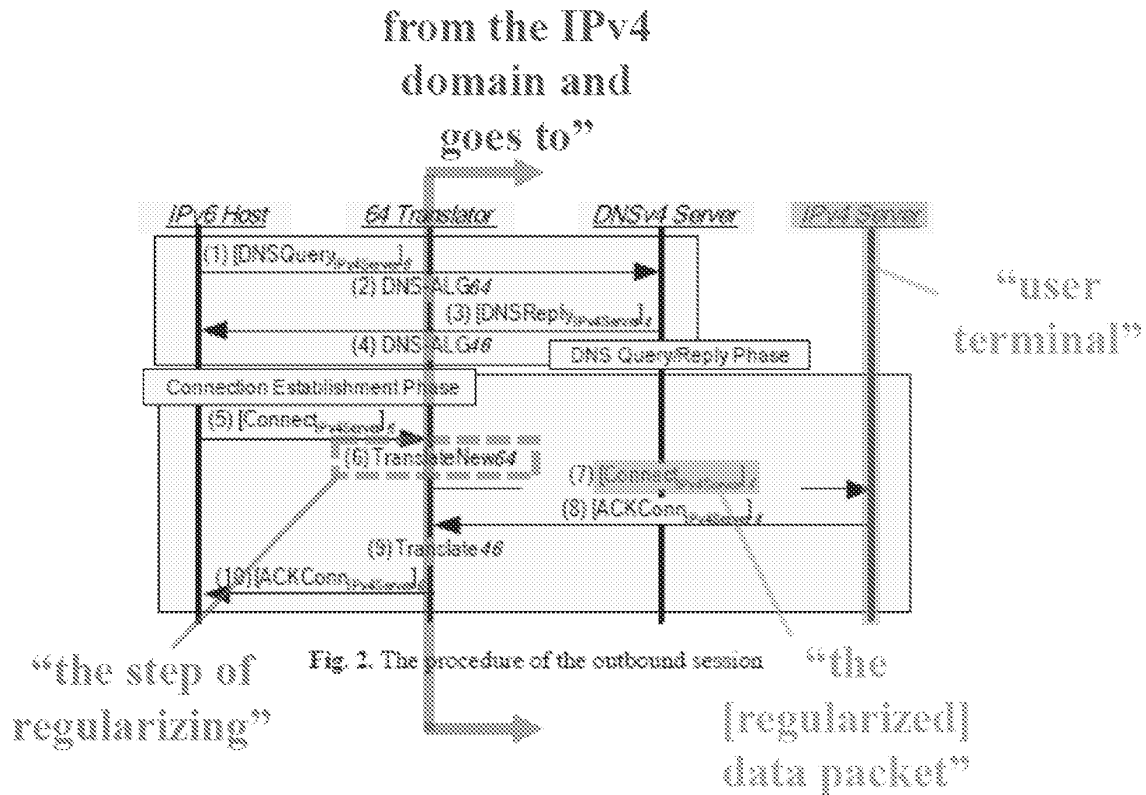
Accordingly, Kong discloses “routing the IPv6 data packet to its destination,” as claimed.

2. Dependent Claim 5

Claim 5 depends from independent claim 1 and recites the additional elements quoted below. Kong renders unpatentable independent claim 1 for the reasons discussed above. Moreover, Kong discloses or suggests the additional elements of claim 5. Ex. 1026, ¶ 187.

- a. [5.pre] “The method according to claim 1, wherein when the data packet comes from the IPv4 domain and goes to said user terminal, the step of regularizing the destination address comprises the following steps:”

Kong discloses or suggests this element. Ex. 1026, ¶¶ 188-192. To the extent that the preamble is limiting, Kong discloses *regularizing* the [ConnectIPv4Server]₆ packet *when the data packet comes from the IPv4 domain* (post-translation by the 64 Translator) *and goes to said user terminal* (IPv4 server), as claimed:



Kong, Fig. 2*

As an initial matter, a POSITA would have understood that IPv4 “Server” in Figure 2 of Kong above actually refers to the IPv4 “Host.” Ex. 1026, ¶ 189. Kong does not disclose an “IPv4 Server” on the Ipv4 network—only an “Ipv4 Host” along with a DNS server, a DHCP server, and a Registration Agent. Ex. 1020, 498. Confirming that the Ipv4 Server refers to Kong’s Ipv4 Host, Figure 4 shows an experimental environment Kong set up to implement its 6TO4 translation technique, and that environment only includes an Ipv6 Host connected to an IPv4 Host through the 64 Translator, with both hosts shown as user terminals:

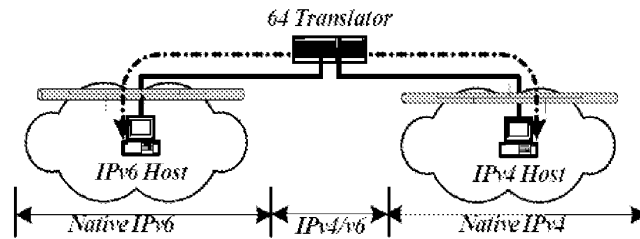


Fig. 4. Experimental environment for the 64 Translator

Kong, FIG. 4

The reason Kong calls the IPv4 Host a “server” in Figure 2 is because the IPv4 Host acts as a “server” during the Connection Establishment Phase by responding to the Connect packet sent from the IPv4 Host. Ex. 1026, ¶ 190; Ex. 1020, 501.

As explained above for element [1.2.1], step (6) of Figure 2 in which the 64 Translator translates the [ConnectIPv4Server]₆ packet to the [ConnectIPv4Server]₄ packet corresponds to the claimed *step of regularizing*. *Supra* Section V.B.1.e; Ex. 1020, 501, FIG. 2; Ex. 1026, ¶ 191. Once translated, “[t]he 64 Translator sends the translated packet to IPv4 Server” over the IPv4 network. Ex. 1020, 501; Ex. 1026, ¶ 191. Thus, Kong discloses performing *the step of regularizing* (translating) *when the data packet* (Connect packet) *comes from the IPv4 domain and goes to said user terminal* (IPv4 Server), as claimed. Ex. 1026, ¶ 191.

Accordingly, Kong discloses “*wherein when the data packet comes from the IPv4 domain and goes to said user terminal, the step of regularizing the destination address comprises the following steps,*” as claimed.

- b. [5.1] “*searching an address translation table of the gateway for an entry associating with said constructed destination address a native destination address;*”
[5.2] “*if an entry is found, replacing the constructed destination address of the data packet with the native destination address from that entry.*”

Kong discloses or suggests these elements. Ex. 1026, ¶¶ 193-197. Specifically, Kong discloses *searching an address translation table* (mapping entries) *of the gateway* (64 Translator) *for an entry* (mapping entry) *associating with said constructed destination address* (IPv6 address in the [ConnectIPv4Server]₆ packet) *a native destination address* (IPv4 address in the

[ConnectIPv4Server]₄ packet). *Id.* Kong also discloses *if an entry* (mapping entry) *is found, replacing the constructed destination address* (IPv6 destination address) *of the data packet* ([ConnectIPv4Server]₆ packet) *with the native destination address* (corresponding IPv4 address in the [ConnectIPv4Server]₄ packet) *from that entry. Id.*

Kong explains that the 64 Translator maintains multiple “mapping entries” used for address translation. *See, e.g.,* Ex. 1020, 499, 501, 501; Ex. 1026, ¶ 194. The mapping entries maintained by the 64 Translator disclose the claimed *address translation table. Id.*

As Kong explains, when translating the [ConnectIPv4Server]₆ packet to the [ConnectIPv4Server]₄ packet, “[t]he 64 Translator ... translates the [ConnectIPv4Server]₆ into the [ConnectIPv4Server]₄ *using the mapping entry.*” Ex. 1020, 501; *see also id.*, 500 (Table 3 explaining that functions of the 64 Translator include “creation of the mapping entry and a translation of the packet from IPvX to IPvY using that mapping entry”); Ex. 1026, ¶ 195. By using a stored mapping entry to translate the [ConnectIPv4Server]₆ packet into the [ConnectIPv4Server]₄ packet, the 64 Translator *searches an address translation table* (plurality of mapping entries) *for an entry* (the mapping entry needed for that particular translation). Ex. 1026, ¶ 195.

Moreover, Kong discloses or suggests that the mapping entry *associates with said constructed destination address* (IPv6 address in the [ConnectIPv4Server]₆ packet) *a native destination address* (IPv4 address in the [ConnectIPv4Server]₄ packet). Ex. 1026, ¶ 196. Kong explains that “[t]he mapping entry is a tuple of <IPv4Addr, IPv6Addr, IPv4Port, IPv6Port, TCPTimer, UDPTimer > where IPvXAddr and IPvXPort are IPvX address and port number of IPvX packet (X is 4 or 6[]).” Ex. 1020, 499, n. 1. Because the translation process *replaces the constructed* IPv6 address with the *native* IPv4 address as explained above, a POSITA would understand that “IPv4Addr” and “IPv6Addr” in the mapping entry respectively correspond to the claimed *native destination address* and *constructed destination address*. Ex. 1026, ¶ 196. Moreover, these addresses are *associated with* one another, as claimed, by appearing in the same mapping entry tuple: “<IPv4Addr, IPv6Addr ...>”. Ex. 1020, 499, n. 1; Ex. 1026, ¶ 196. And by using this mapping entry to translate the [ConnectIPv4Server]₆ packet as disclosed by Kong, the 64 Translator *replaces the constructed destination address* (IPv6 address) *of the data packet with the native destination address* (IPv4 address) *from that entry* (mapping entry tuple), as claimed.

Accordingly, Kong discloses or suggests “*wherein when the data packet comes from the*

IPv4 domain and goes to said user terminal, the step of regularizing the destination address comprises the following steps” and “if an entry is found, replacing the constructed destination address of the data packet with the native destination address from that entry,” as claimed.

3. Dependent Claim 6

Claim 6 depends from independent claim 5 and recites the additional elements quoted below. Kong renders unpatentable claim 5 for the reasons discussed above. Moreover, Kong discloses or suggests the additional elements of claim 6. Ex. 1026, ¶¶ 198-200.

- a. [6.1] “*The method according to claim 5, further comprising a step of verifying that the destination port number of the data packet and the destination port number contained in the constructed destination address match.*”

Kong discloses or suggests this element. Ex. 1026, ¶ 199. As explained above, in Kong, “an IPv6 address ... is created by the concatenation of dummy prefix and IPv4 address,” and that IPv4 address contains a destination port number. Ex. 1020, 501; Ex. 1026, ¶ 199; *Supra* Section V.A.B.1.d. As a result, a POSITA would understand that the destination port number in the *constructed* IPv6 address of the [ConnectIPv4Server]₆ packet *matches* the destination port number of the underlying IPv4 address. Ex. 1026, ¶ 199. Kong discloses or suggests *verifying* that the two port numbers match by looking up the correct mapping entry for the translation using the address information in the packet. *See, e.g.*, Ex. 1020, 501 (“it looks up the mapping entries with <IPv4 address”); Ex. 1026, ¶ 199. Moreover, as explained below in Ground 5, it also would have been obvious to *verify* that port numbers match to detect and discard potentially fraudulent/spoofed traffic. *Infra* Section V.C.3; Ex. 1026, ¶ 199.

Accordingly, Kong discloses or suggests “*a step of verifying that the destination port number of the data packet and the destination port number contained in the constructed destination address match,*” as claimed.

4. Dependent Claim 7

Claim 7 depends from independent claim 1 and recites the additional elements quoted below. Kong renders unpatentable independent claim 1 for the reasons discussed above. Moreover, Kong discloses or suggests the additional elements of claim 7. Ex. 1026, ¶¶ 201-205.

- a. [7.1] “*A non-transitory computer program product that is at least one of downloadable from a communications network and stored*

on a computer-readable medium and executable by a microprocessor, comprising program code instructions for executing the method according to claim 1 of receiving a data packet in an IPv6 domain.”

Kong discloses or suggests this element. Ex. 1026, ¶¶ 202-203. Specifically, Kong discloses *a non-transitory computer program product* (Linux algorithm based on NAT-PT code) *that is at least one of downloadable from a communications network and stored on a computer-readable medium and executable by a microprocessor* (333 MHz Pentium processor of the 64 Translator), *comprising program code instructions* (Linux algorithm based on NAT-PT code) *for executing the method according to claim 1 of receiving a data packet ([ConnectIPv4Server]₆) in an IPv6 domain* (IPv6 network). *Id.*

Kong explains that, “[i]n order to verify the function of the 64 Translator and measure its performance, we set up the experimental environment as shown in Fig.4.” Ex. 1020, 503.

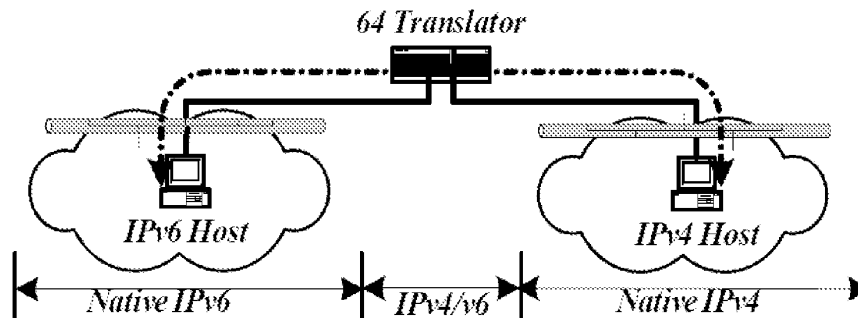


Fig. 4. Experimental environment for the 64 Translator

Kong, FIG. 4

Kong teaches that “the 64 Translator is implemented on a 333MHz Pentium PC,” disclosing the claimed *microprocessor*. Ex. 1020, 503; *see also* Ex. 1026, ¶ 204. Moreover, the functions of the 64 Translator were performed by the PC’s Pentium processor “running Linux by our algorithm ... based on NAT-PT code,” disclosing the claimed *computer program product* comprising *computer code* and *executable by a microprocessor*. Ex. 1020, 503; *see also* Ex. 1026, ¶ 204. A POSITA would have recognized that the Pentium PC necessarily has the claimed *computer-readable medium* (memory) storing Kong’s Linux algorithm to enable the Pentium processor to execute the algorithm and also to store the mapping entries. Ex. 1026, ¶ 204.

Accordingly, Kong discloses or suggests “*non-transitory computer program product that is at least one of downloadable from a communications network and stored on a computer-readable medium and executable by a microprocessor, comprising program code instructions for executing the method according to claim 1 of receiving a data packet in an IPv6 domain,*” as claimed.

5. Independent Claim 8

a. [8.pre.1] “A device for receiving an IPv6 data packet in an IPv6 domain connected to an IPv4 domain,”

This element is identical to element [1.pre.1] of independent claim 1, except that element [8.pre.1] recites a “device” instead of a “method.” To the extent this element is limiting, Kong’s 64 Translator corresponds to the claimed *device*, and the 64 Translator *receives an IPv6 data packet in an IPv6 domain connected to an IPv4 domain* for the reasons discussed above for element [1.pre.1]. *Supra* Section V.B.1.a; Ex. 1026, ¶ 206.

b. [8.pre.2] “said packet comprising an IPv6 destination address and an IPv6 source address,”

This element is identical to element [1.pre.2] of independent claim 1. Accordingly, to the extent the preamble is limiting, Kong discloses or suggests this element for the reasons discussed above for element [1.pre.2]. *Supra* Section V.B.1.b; Ex. 1026, ¶ 207.

c. [8.pre.3] “said device for use in a home gateway adapted to connect a user terminal to said IPv6 domain, comprising:”

This element is virtually identical element [1.pre.3] of independent claim 1. Accordingly, to the extent the preamble is limiting, Kong discloses or suggests this element for the reasons discussed above for element [1.pre.3]. *Supra* Section V.B.1.c; Ex. 1026, ¶ 208.

d. [8.1] “means for identifying an IPv6 destination address constructed by concatenating an IPv6 prefix, an IPv4 destination address, and a destination port number;”

This element recites means for performing a function identical to element [1.1] of independent claim. As explained above, this element invokes the following construction under § 112, 6th Paragraph:

- **Claimed Function:** identifying an IPv6 destination address constructed by concatenating an IPv6 prefix, an IPv4 destination address, and a destination port number; and
- **Corresponding Structure:** a receiver device of the home gateway having a

processor and memory, and equivalents

Supra Section I.E.3.a. For the reasons discussed above for element [1.1], Kong's 64 Translator performs the claimed function. *Supra* Section V.B.1.d; Ex. 1026, ¶ 209.

Moreover, Kong's 64 Translator satisfies the corresponding structure. Ex. 1026, ¶ 210. As discussed, the 64 Translator corresponds to the claimed home gateway. *Supra* Section V.B.1.c. Moreover, as explained, Kong implemented the 64 Translator as a Pentium PC, which has a 333 MHz Pentium processor and memory. Ex. 1020, 503; Ex. 1026, ¶ 210.

Accordingly, Kong discloses "*means for identifying an IPv6 destination address constructed by concatenating an IPv6 prefix, an IPv4 destination address, and a destination port number,*" as claimed.

e. [8.2-3] "*means for*

[8.2.1] *regularizing, if necessary, at least one of the IPv6 source address and the IPv6 destination address of the data packet by replacing said at least one IPv6 address,*

[8.2.2] *the replacement process belonging to a group comprising replacement of a constructed address with a native address and replacement of a native address with a constructed address, and*

[8.3] *modifying the data packet using the regularized address; and*"

Element [8.2-3] recites means for performing functions [8.2.1], [8.2.2], and [8.2.3] respectively identical to elements [1.2.1], [1.2.2], and [1.3]. As explained above, elements [8.2-3] invokes the following construction under § 112, 6th Paragraph:

• **Claimed Functions:**

- [8.2.1] if necessary, (1) regularizing the IPv6 source address by replacing the IPv6 source address or (2) regularizing the IPv6 destination address by replacing the IPv6 destination address;¹³
- [8.2.2] the replacement process belonging to a group comprising replacement of a constructed address with a native address and replacement of a native address with a constructed address; and

¹³ Function [8.2.1] incorporates the proposed construction for element [1.2.1] discussed above. *Supra* Section I.E.2.

- [8.2.3] modifying the data packet using the regularized address.
- **Corresponding Structure:** the device of the home gateway having a processor and a memory, where the memory stores a table for translating between addresses, and equivalents.

Supra Section I.E.3.b. For the reasons discussed above for elements [1.2.1], [1.2.2], and [1.3], the 64 Translator of Kong performs the claimed functions [8.2.1], [8.2.2], and [8.2.3], respectively. *Supra* Sections V.B.1.e-g; Ex. 1026, ¶ 212.

Moreover, Kong's 64 Translator satisfies the corresponding structure. Ex. 1026, ¶ 213. As explained above for element [8.1], the 64 Translator is the device of the home gateway having a 333 MHz Pentium processor and memory. *Supra* Section V.B.1.d. Additionally, as explained above for dependent claim 5, Kong discloses that the 64 Translator maintains mapping entries (*table*) for translating between addresses (i.e., converting between IPv6 and IPv4 addresses, and vice versa). *Supra* Section V.B.2; Ex. 1026, ¶ 203.

Accordingly, Kong discloses “*means for regularizing, if necessary, at least one of the IPv6 source address and the IPv6 destination address of the data packet by replacing said at least one IPv6 address, the replacement process belonging to a group comprising replacement of a constructed address with a native address and replacement of a native address with a constructed address, and modifying the data packet using the regularized address,*” as claimed.

f. [1.4] “means for routing the IPv6 data packet to its destination.”

This element recites means for performing a function identical to element [1.4] of independent claim. As explained above, this element invokes the following construction under § 112, 6th Paragraph:

- **Claimed Function:** routing the IPv6 data packet to its destination.
- **Corresponding Structure:** telecommunications means of the device of the home gateway, and equivalents

Supra Section I.E.3.c. For the reasons discussed above for element [1.4], Kong's 64 Translator performs the claimed function. *Supra* Section V.B.1.e-h.

Moreover, the 64 Translator satisfies the corresponding structure. Ex. 1026, ¶ 216. As discussed, the 64 Translator discloses claimed home gateway. *Supra* Section V.B.1.c. Moreover, Kong teaches that “the 64 Translator is implemented on a 333MHz Pentium PC” and that 64

Translator PC communicated with the IPv6 and IPv4 Host PCs via “100Mbps Ethernet,” which discloses *telecommunications means*, as claimed. Ex. 1020, 503. Figure 4 of Kong below shows the 64 Translator sitting between IPv6 and IPv4 networks to communicate with the Hosts. Ex. 1026, ¶ 216.

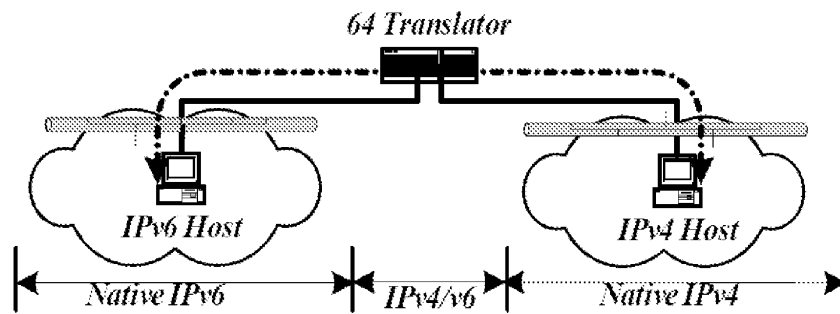


Fig. 4. Experimental environment for the 64 Translator

Kong, FIG. 4

Accordingly, Kong discloses “*means for routing the IPv6 data packet to its destination,*” as claimed.

C. Ground 5: Ananda in View of Wetterwald and RFC 4380 Would Have Rendered Obvious Claims 1 and 5-8

As explained below on an element-by-element basis, Ananda, Wetterwald, and RFC 4380 would have rendered obvious claims 1 and 5-8 of the '845 Patent under 35 U.S.C. § 103(a) (pre-AIA).

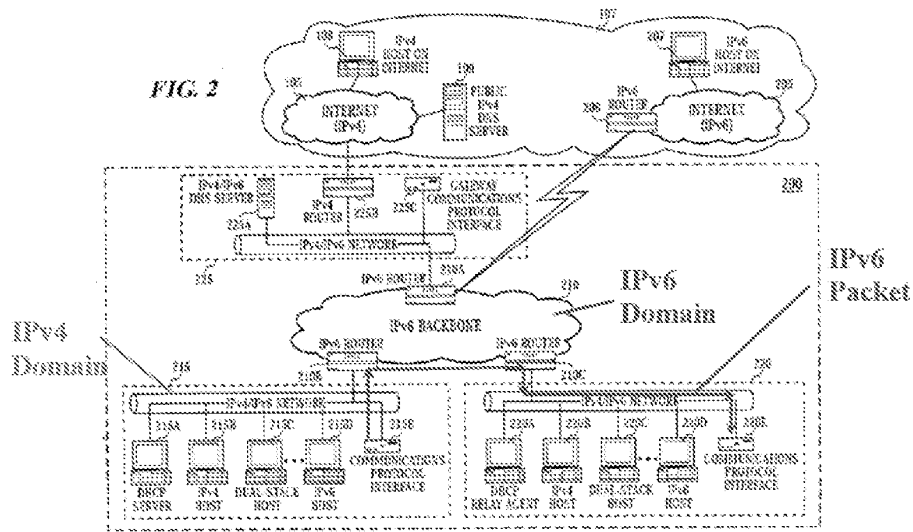
1. Claim 1

a. [1.pre.1] “A method for receiving an IPv6 data packet in an IPv6 domain connected to an IPv4 domain,”

Ananda renders obvious element [1.pre.1]. Ex. 1003, ¶ 95.

Ananda discloses “a method in a communication protocol interface (CPI) for communicating a variety of data packets that comply with a corresponding variety of communication protocols on a communication network,” where the method includes “*receiving a data packet.*” Ex. 1005, ¶¶ 35-36, 18. Ananda explains that CPIs in multi-protocol subnets of the communication network transport IPv4 data packets across an IPv6 communication backbone (“*IPv6 domain*”) encapsulated in IPv6 data packets between the subnets. Ex. 1005, ¶ 63; Ex. 1003, ¶¶ 96-98.

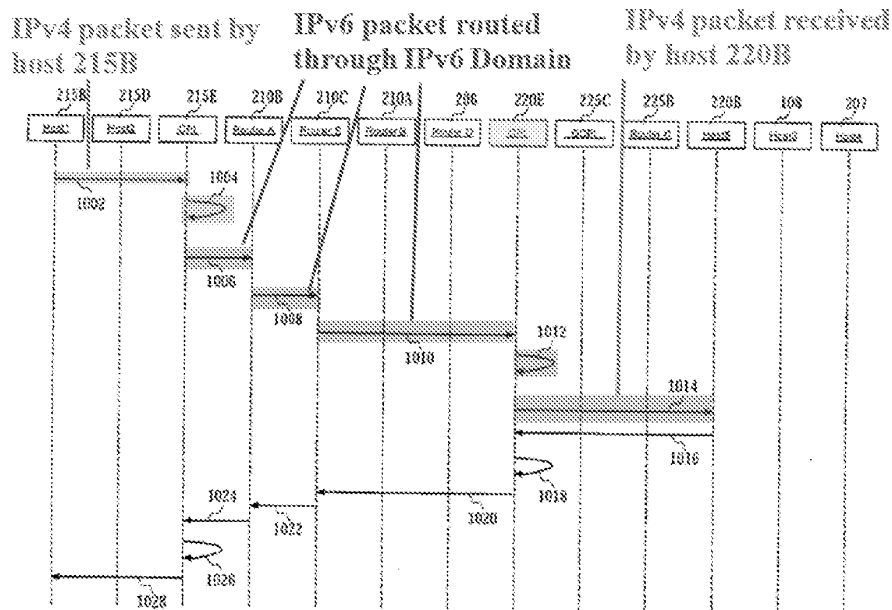
With reference to annotated Figure 2 below, Ananda discloses that the CPI 220E of multi-protocol IPv4/IPv6 subnet 220 receives an IPv6 packet in an IPv6 backbone 210 (“*IPv6 domain*”), which is connected to a multi-protocol IPv4/IPv6 subnet 215 (“*IPv4 domain*”). Ex. 1005, ¶¶ 79, 139-142. Ananda discloses that CPI 220E of subnet 220 and CPI 215E of subnet 215 communicate according to both the IPv4 and IPv6 protocols. Ex. 1005, ¶¶ 73, 78-79. Ananda discloses that the CPIs 215E, 220E transport IPv4 data packets across the IPv6 communication backbone 210 using IPv6 data packets, thereby allowing the IPv4 hosts 215B, 220B in the subnets 215, 220 to communicate with each other, and to allow multi-protocol IPv4/IPv6 communications between IPv4- and IPv6-configured devices in the subnets 215, 220. Ex. 1005, ¶¶ 76-79, Abstract; Ex. 1003, ¶¶ 99-101.



Ex. 1003, ¶ 102; Ex. 1005, Fig. 2*

Annotated Figure 2 above shows an IPv6 packet being transmitted from CPI 215E in the subnet 215 (“*IPv4 domain*”) across the IPv6 backbone 210 (“*IPv6 domain*”) to CPI 220E of subnet 220. Ex. 1005, ¶¶ 79, 139-142; Ex. 1003, ¶ 102.

With reference to annotated Figure 10 below, Ananda describes the sequence of actions when an IPv4 data packet sent by the IPv4 host 215B in subnet 215 (“*IPv4 domain*”) is routed via the IPv6 communication backbone 210 (“*IPv6 domain*”) and CPI 220E to the destination IPv4 host 220B in subnet 220. Ex. 1005, ¶ 136; Ex. 1003, ¶ 103.



Ex. 1003, ¶ 103; Ex. 1005, Fig. 10*

In step 1002 of Figure 10, the IPv4 host 215B in subnet 215 (“*IPv4 domain*”) sends an IPv4 data packet to the IPv4 host 220B. Ex. 1005, ¶ 137. In step 1004, CPI 215E encapsulates the IPv4 packet into an IPv6 packet and transmits it through IPv6 backbone 210 (“*IPv6 domain*”) via routers 210B and 210C (steps 1006 and 1008). Ex. 1005, ¶¶ 138-140. In step 1010, the CPI 220E receives the IPv6 packet from router 210C in IPv6 backbone 210 (“*IPv6 domain*”). In step 1012, CPI 220E decapsulates the header of the IPv6 packet to produce the original IPv4 packet transmitted by IPv4 host 215B, and transmits the IPv4 packet to IPv4 host 220B. Ex. 1005, ¶¶ 142-143. Ex. 1003, ¶ 104.

While Requester does not believe that “*IPv4 domain*” or “*IPv6 domain*” requires claim construction, it would have been obvious to a POSITA that Ananda’s subnets 215 and 220 are each a separate *IPv4 domain* and the IPv6 backbone 210 is an *IPv6 domain* under any reasonable construction of those terms. Ex. 1003, ¶ 105. For example, to the extent that *IPv4 domain* is interpreted to refer to an administrative partition of a network that employs the IPv4 protocol, Ananda’s Figures 2 and 8 show subnets 215 and 220 in dotted-line boxes, indicating that they are separate partitions of the overall network. Ananda states that each subnet is a “physically independent network segment.” Ex. 1005, ¶ 66. Subnets 215 and 220 also include devices

communicating via the IPv4 protocol. Ex. 1005, ¶¶ 76, 78-79, 82 (“a subnet can comprise purely IPv4 hosts”). Ex. 1003, ¶¶ 105-106.

To the extent that IPv4 domain is interpreted to mean a part of a computer network that uses the IPv4 protocol and in which the data processing resources are under common control, Ananda further explains how each subnet includes a CPI and an IPv6 router that control network traffic flowing to and from the individual subnet. Ex. 1005, ¶¶ 78-79. In particular, the IPv4 devices in subnet 215 are under the common control of CPI 215E, which enables multi-protocol communication within the subnet 215 (Ex. 1005, ¶ 78), and CPI 215E and IPv6 router 210B exchange IPv6 packets with the IPv6 network 210 on behalf of IPv4 devices in the subnet 215 (e.g., IPv4 host 215B). Ex. 1005, ¶ 137 (step 1002 in Fig. 10), ¶ 150 (step 1028 in Fig. 10). Communications to and from subnet 220 are facilitated separately by IPv6 router 210C and CPE 220E. *See* Ex. 1005, ¶¶ 143-144 (steps 1014 and 1016 in Fig. 10). Therefore, a POSITA would have understood that the collection of IPv4 hosts in subnet 215 are under common control and that the common control of subnet 215 is separate from the common control of subnet 220. Thus, subnet 215 and subnet 220 are each a separate *IPv4 domain*. Ex. 1003, ¶ 107.

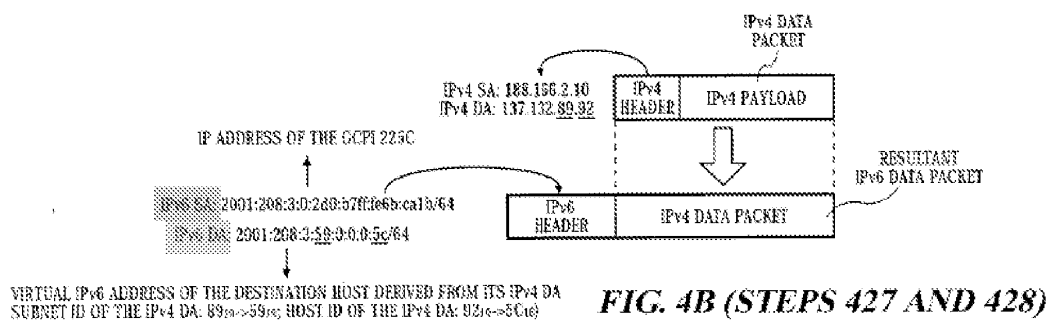
Finally, to the extent that *IPv4 domain* is interpreted to mean a collection of IPv4 hosts, IPv4 routers, and interconnecting network(s) managed by a single administrative authority, Ananda teaches that subnets 215 and 220 are each a “physically independent network segment.” Ex. 1005, ¶ 66. Because each subnet is physically independent, it would have been obvious to a POSITA for each subnet to be managed by an independent administrative authority. Ex. 1003, ¶¶ 85, 108. Ananda teaches that each subnet has a separate computer configured to provide dynamic host control protocol (DHCP) services (e.g., servers 215A and 220A). Ex. 1003, ¶¶ 108, 68, 78. DHCP was a well-known protocol used to administer IPv4 network addresses. *See, e.g.*, Ex. 1036, 1 (providing “automatic allocation of reusable network addresses”), 3, 12; Ex. 1003, ¶¶ 85, 108. Therefore, a POSITA would have understood that the hosts in subnet 215 are managed separately from the hosts in subnet 220. Thus, subnet 215 and subnet 220 are each a separate *IPv4 domain*. Ex. 1003, ¶ 108.

Accordingly, Ananda renders obvious element [1.pre.1]. Ex. 1003, ¶ 109.

- b. [1.pre.2] “said packet comprising an IPv6 destination address and an IPv6 source address,”**

Ananda renders obvious element [1.pre.2]. Ex. 1003, ¶ 110.

As described above with respect to element [1.pre.1], CPI 215E in subnet 215 encapsulates an IPv4 packet received from IPv4 host 215B to produce an IPv6 packet. Ex. 1005, ¶ 138. As shown in annotated Figure 4B below, Ananda discloses that the packet transmitted from CPI 215E to CPI 220E includes an IPv6 destination address (“*IPv6 DA*”) and an IPv6 source address (“*IPv6 SA*”) in the header of the IPv6 packet (“*resultant IPv6 data packet*”). Ex. 1005, ¶ 138 (IPv6 packet has IPv6 source address of CPI 215E and constructed IPv6 destination address of IPv4 host 220B). Ex. 1003, ¶ 111.



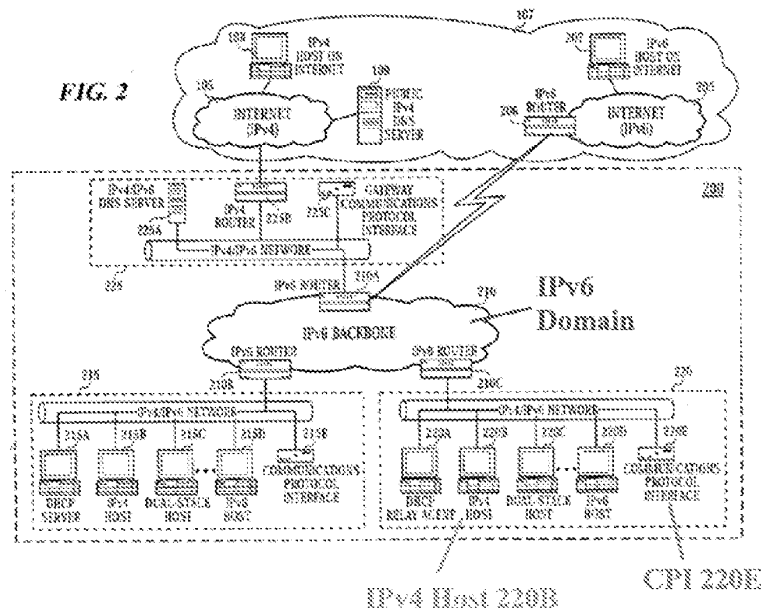
Ex. 1003, ¶ 111; Ex. 1005, Fig. 4B*

Thus, Ananda renders obvious element [1.pre.2]. Ex. 1003, ¶ 112.

- c. [1.pre.3] “*said method being executed in a home gateway adapted to connect a user terminal to said IPv6 domain, and comprising the following steps:*”

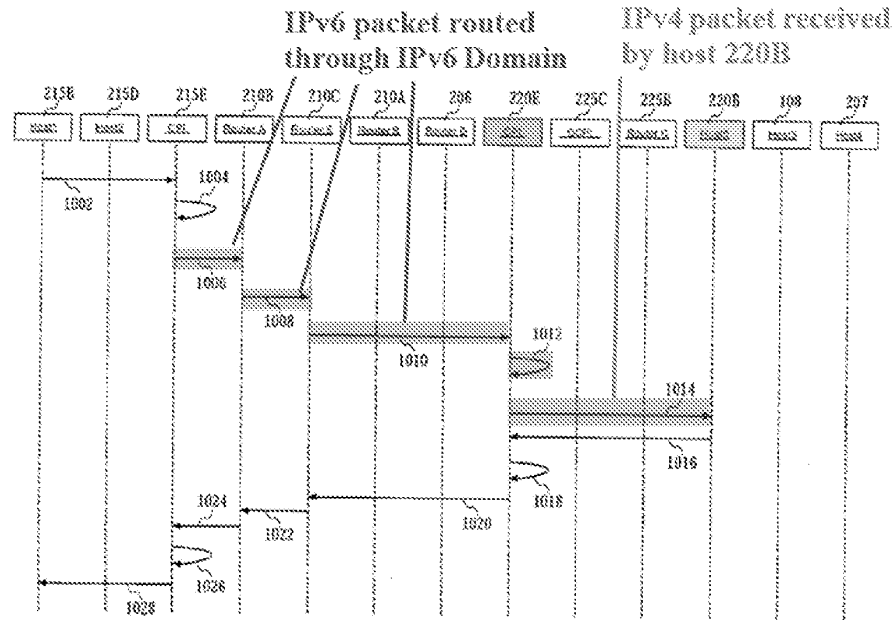
Ananda renders obvious element [1.pre.3]. Ex. 1003, ¶¶ 113.

With reference to annotated Figure 2 below, Ananda discloses that the method is executed in CPI 220E (“*home gateway*”) adapted to connect IPv4 host 220B (“*user terminal*”) to the IPv6 backbone 210 (“*IPv6 domain*”). Ananda discloses that the CPI 220E (“*home gateway*”) enables IPv4 host 220B (“*user terminal*”) to communicate with IPv6 backbone 210 (“*IPv6 domain*”). Ex. 1005, ¶¶ 79, 77-78; *see also* Ex. 1005, ¶ 63 (CPIs in multi-protocol subnets transport IPv4 data packets across an IPv6 communication backbone encapsulated in IPv6 data packets between the subnets). Ex. 1003, ¶ 114.



Ex. 1003, ¶ 114; Ex. 1005, Fig. 2*

As shown in annotated Figure 10 below, Ananda discloses that **CPI 220E** (“*home gateway*”), upon receiving an IPv6 packet from CPI 215E through IPv6 backbone 210 (via IPv6 routers 210B and 210C), extracts (decapsulates) the IPv4 packet in the transmitted IPv6 packet and routes the IPv4 packet to **IPv4 host 220B** (“*user terminal*”). Ex. 1005, ¶¶ 138-143; Ex. 1003, ¶ 115.



Ex. 1003, ¶ 115; Ex. 1005, Fig. 10*

Thus, Ananda renders obvious element [1.pre.3]. Ex. 1003, ¶ 116.

- d. [1.1] “identifying an IPv6 destination address constructed by concatenating an IPv6 prefix, an IPv4 destination address and a destination port number;”

Ananda in view of Wetterwald and RFC 4380 renders obvious element [1.1]. Ex. 1003, ¶¶

117.

With reference to annotated Figure 4 below, Ananda discloses that the IPv6 packet transmitted from CPI 215E includes *an IPv6 destination address* that is *constructed*. Ex. 1005, Fig. 4, step 440, ¶ 138 (construction of IPv6 destination address). Ex. 1003, ¶ 118. As described above with respect to elements [1.pre.1] and [1.pre.3], the CPI 220E receives the IPv6 packet from the CPI 215E after it is routed through IPv6 backbone 210. Ex. 1005, ¶¶ 138-142. Ananda states that a “determination 651 is made as to whether the IPv6 data packet contains an IPv4 data packet in its payload.” Ex. 1005, ¶ 113. Ananda also describes determining “whether the destination address of the IPv6 data packet is within the multi- protocol subnet.” Ex. 1005, ¶ 114. Consistent with these teachings and Ananda’s use of virtual IPv6 destination addresses (Ex. 1005, ¶¶ 82, 87), it would have been obvious for CPI 220E to identify that the IPv6 destination address included in the IPv6 header is a *constructed* address to facilitate directing the packet to the packet decapsulation process. Ex. 1005, ¶ 109, 142; Ex. 1003, ¶ 119.

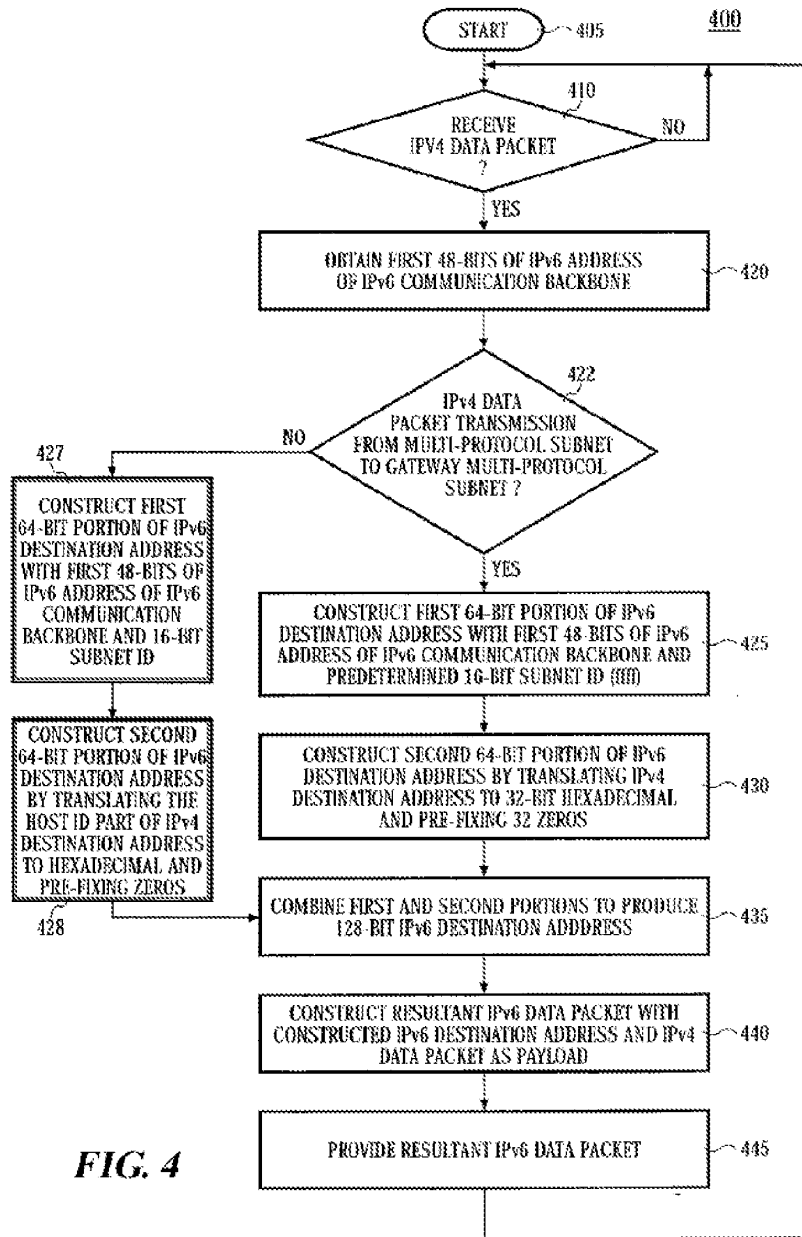


FIG. 4

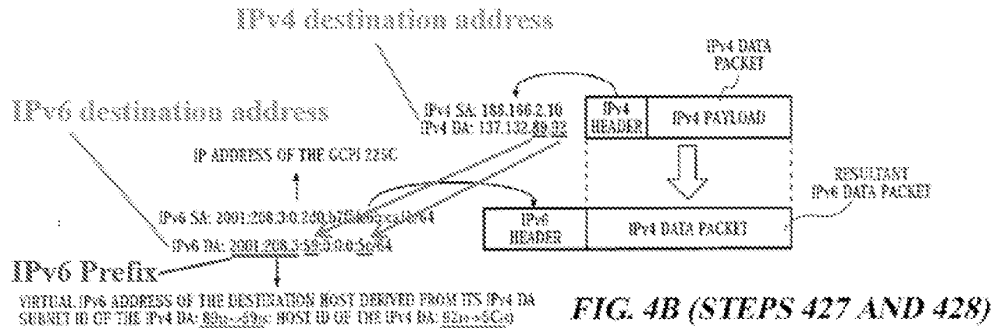
Ex. 1003, ¶ 118; Ex. 1005, Fig. 4*

Annotated Figure 4 above illustrates an encapsulation process 400 that is performed by the encapsulator 352 of CPI 215E (see Figure 3) of subnet 215, when the CPI 215E receives an IPv4 packet from IPv4 host 215B (step 410). Ex. 1005, ¶¶ 103, 99 (describing encapsulator 352). Ananda discloses that the CPI 215E supports dynamic encapsulation of IPv4 data packets, resulting in IPv6 data packets, where “the resultant IPv6 or outer encapsulation layer destination address is advantageously derived and **constructed from the destination address of the encapsulated IPv4 data packet**, dynamically.” Ex. 1005, ¶ 116; Ex. 1003, ¶¶ 120- 122.

As shown in annotated Figure 4 above, the encapsulation process 400 of the encapsulator 352 of CPI 215E (see Fig. 3) includes steps 427, 428, and 435 in which an *IPv6 destination address* is *constructed*. Steps 427 and 428 correspond to the situation where CPI 215E and CPI 220E communicate IPv6 packets with each other over IPv6 backbone 210, as discussed above (“No” in step 422). Ex. 1005, ¶¶ 103, 106. The IPv6 packet produced by CPI 215E is constructed to include an IPv6 destination address, as discussed below. Ex. 1003, ¶ 123.

Ananda explains that in step 427, a first portion of the constructed IPv6 address is formed from a 48-bit IPv6 prefix associated with the IPv6 communication backbone (“*IPv6 prefix*”). Ex. 1005, ¶ 106. Ananda explains that in step 428, a second portion of the constructed IPv6 address is formed from the original IPv4 destination address (“*an IPv4 destination address*”) of the IPv4 packet received from IPv4 host 215E. Ex. 1005, ¶ 107; Ex. 1003, ¶¶ 124-125.

As shown below with reference to annotated Figure 4B, Ananda discloses that the IPv6 destination address (DA) is constructed by combining (1) a 48-bit **IPv6 prefix**,⁸ and (2) the address of the IPv4 destination address. Ex. 1005, ¶ 106, Fig. 4B. As shown in annotated Figure 4B below, “89.92” is part of the IPv4 destination address of IPv4 host 220B (“**IPv6 destination address**”). Ex. 1005, ¶¶ 83-84. The IPv6 destination address constructed in Figure 4B includes (1) the IPv6 prefix concatenated with (2) two IPv6 address segments that are hexadecimal translations of the decimal values “89” and “92” taken from the IPv4 address of the IPv4 host 220B (i.e., **89 decimal = 59 hexadecimal**, and **92 decimal = 5c hexadecimal**). Ex. 1005, ¶¶ 107-108, 138; Ex. 1003, ¶¶ 126-127.



Ex. 1003, ¶ 127; Ex. 1005, Fig. 4B*

To the extent that Patent Owner argues that claim 1 requires incorporating all 32 bits of the IPv4 destination address into the constructed IPv6 destination address, Ananda renders obvious such an interpretation. Ex. 1003, ¶ 128. Ananda teaches an alternative approach to constructing an IPv6 address as shown in Figure 4A, in which the constructed IPv6 address includes all 32 bits of the original IPv4 destination address. Ex. 1005, ¶¶ 104 (steps 425 and 430 in Figs. 4 and 4A), 127; *see also* Ex. 1003, ¶ 129 (explaining alternative approach).

Thus, Ananda discloses that *an IPv6 destination address is constructed by concatenating an IPv6 prefix and an IPv4 destination address*, as shown in Figure 4B above. Ex. 1003, ¶ 130.

As discussed above, the CPI 215E encapsulates an IPv4 packet received from IPv4 host 215B into an IPv6 packet that is transmitted to CPI 220E over the IPv6 backbone 210. Ex. 1005, ¶¶ 138-141; Ex. 1003, ¶ 131.

Wetterwald teaches that it was known, when encapsulating IPv4 packets in IPv6 packets, to include IPv6 extension headers to identify the source and destination port numbers.

If desired, the translator 42 also **may add IPv6 option headers related to the layer 4 TCP/UDP source and destination ports**, as described in detail in RFC 2460.

Ex. 1008, 8:24-28; Ex. 1003, ¶¶ 134, 132-133, 135-137.

RFC 4380 discloses that it was known to include a destination port number in a constructed IPv6 address. The Teredo protocol, for example, was used to provide connectivity and integration between IPv4 and IPv6 networks, like Ananda and Wetterwald. The Teredo protocol defines a

“Teredo IPv6 address” that includes “the UDP port and IPv4 address at which a client can be reached.” Ex. 1009 at 5 (§ 2.13); Ex. 1003, ¶ 138.

A POSITA would have been motivated to modify Ananda’s IPv6 address construction technique to incorporate a destination port number in the IPv6 address, as taught by Wetterwald and RFC 4380, for the same reasons that the Teredo protocol incorporates a port number in an IPv6 address. For example, Teredo protocol uses the port number included in the IPv6 address to detect potentially fraudulent (“spoofed”) traffic:

3) If the source IPv6 address is a Teredo address, the **client compares the mapped IPv4 address and mapped port** in the source address **with the source IPv4 address and source port of the packet....**

5) If the source IPv6 address is a Teredo address, and the mapped IPv4 address and mapped port in the source address do not match the source IPv4 address and source port of the packet, the client checks whether there is an existing "local" entry for that IPv6 address....

6) **In the other cases**, the packet may be accepted, but **the client should be conscious that the source address may be spoofed**; before processing the packet, the client should perform the "direct IPv6 connectivity test" described in Section 5.2.9.

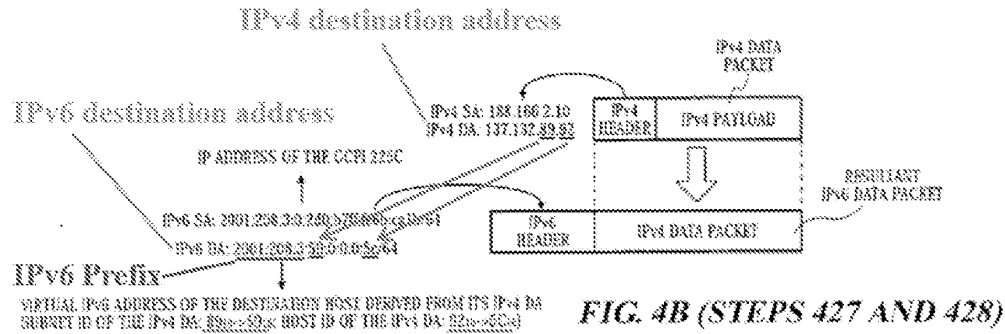
Ex. 1009 at 22-23; Ex. 1003, ¶ 142.

By comparing the destination port number identified in an IPv4 packet with the destination port number incorporated into an IPv6 destination address, the recipient of Ananda’s encapsulated IPv6 packet (e.g., CPI 220E) could potentially detect spoofed traffic. It was well-known in the art that spoofed packets were potentially dangerous and problematic. *See, e.g.*, Ex. 1016, ¶ 3 (describing the use of “spoofing” in network attacks); Ex. 1017, ¶ 3 (“Attackers hide their location using incorrect or ‘spoofed’ IP source addresses”); Ex. 1018, 1:57-61 (“unauthorized access is associated with ‘IP spoofing’”). Thus, the desire to provide some level of protection in Ananda’s mixed IPv4 and IPv6 network against spoofed traffic would have motivated a POSITA to incorporate an IPv4 destination port in Ananda’s constructed IPv6 destination address. Ex. 1003, ¶ 139.

A POSITA would have had an expectation of success in combining the technical features of Ananda, Wetterwald, and RFC 4380, as discussed above, because there would have been ample room (e.g., available bits) in Ananda's constructed IPv6 address to include a 16-bit destination port number. As noted above, Ananda teaches prefixing zeros to the IPv4 address (which is 32 bits in length) to create a 64-bit "second portion" of the constructed IPv6 address. Ex. 1005, ¶ 107 (step 428 in Fig. 4). Hence, Ananda's constructed IPv6 address includes 32 unallocated bits that Ananda suggests filling with zeros. A destination port number is just 16 bits (Ex. 1009, at 12 ("16-bit port number")) and therefore a POSITA would have expected to have had success in including the port number as a substitute for some of the zero bits in Ananda's constructed IPv6 address. Ex. 1003, ¶¶ 140-142.

Accordingly, a POSITA would have been motivated to modify Ananda's technique of constructing an IPv6 destination address by concatenating an IPv6 prefix with an IPv4 destination address, as taught by Ananda, and with a destination port number, as taught by Wetterwald and RFC 4380. Ex. 1003, ¶ 143. This modification represents the combination of prior art elements (constructing an IPv6 prefix with an IPv4 destination address, as taught by Ananda, and with a destination port number, as taught by Wetterwald and RFC 4380) according to known methods (concatenating the IPv6 prefix with the IPv4 destination address and destination port number) to yield the predictable result of including IPv6 extension headers in an IPv6 destination address to identify destination port numbers, as taught by Wetterwald and RFC 4380. *See KSR Int'l Co. v. Teleflex Inc.*, 550 U.S. 398, 401 (2007); Ex. 1003, ¶ 143.

Ananda's CPI 220E identifies an IPv6 destination address that is constructed as described above. In particular, Ananda discloses that the CPI 220E, upon receiving an IPv6 packet, determines whether the received IPv6 packet contains an IPv6 prefix. The CPI 220E identifies the *IPv6 destination address* included in the IPv6 header because it decapsulates IPv6 packet to produce the original IPv4 packet. Ex. 1005, ¶ 142. As shown in annotated Figure 4B below, the IPv6 destination address of the IPv6 packet received by the CPI 220E is constructed to include an IPv6 prefix ("2001:208:3"), which is the IPv6 source address of the CPI 215E and the IPv6 destination address of IPv4 host 220B. Ex. 1005, ¶ 106 (step 427 in Figure 4), 138 (step 1004 in Figure 10); Ex. 1003, ¶ 144.



Ex. 1003, ¶ 144; Ex. 1005, Fig. 4B*

The CPI 220E thus identifies the IPv6 prefix in the IPv6 header, as the IPv6 prefix is the first part of the IPv6 destination address that the CPI 220E strips from the received IPv6 packet. Ex. 1005, ¶ 142, 102; Ex. 1003, ¶ 145. Further, the CPI 220E determines based on the IPv6 prefix whether the IPv6 packet is a virtual IPv6 address constructed as described above (i.e., “[constructed] IPv6 destination address”). In particular, Ananda discloses that the CPI 220E’s IPv6 input handler 332 determines how to process the IPv6 packet based on the IPv6 destination address. Ex. 1005, ¶ 94. Further, Ananda explains that the CPI 220E determines whether the received IPv6 packet encapsulates an IPv4 data packet by “accessing an IPv6 address proxy 340 to **determine whether the destination address of the IPv6 data packet is a virtual IPv6 address of an IPv4 host 215B...**” Ex. 1005, ¶ 96; Ex. 1003, ¶¶ 146-147.

Thus, Ananda, Wetterwald and RFC 4380 render obvious element [1.1]. Ex. 1003, ¶ 148.

- e. [1.2.1] “*if necessary, regularizing at least one of the IPv6 source address and the IPv6 destination address of the data packet by replacing said at least one IPv6 address,*”

Ananda in view of Wetterwald renders obvious element [1.2.1] for the following reasons. Ex. 1003, ¶ 149.

As described above in Section I.E.2, element [1.2.1] requires either (1) regularizing the IPv6 source address by replacing the IPv6 source address, or (2) regularizing the IPv6 destination address by replacing the IPv6 destination address. Ex. 1003, ¶ 151.

Ananda discloses that the CPI 220E, upon receiving the IPv6 packet, replaces the IPv6 destination address in the packet with an IPv4 destination address. Ex. 1005, ¶ 142. The CPI 220E performs this replacement of the IPv6 destination address of the IPv6 packet by decapsulating the IPv6 packet to produce the original IPv4 packet transmitted from IPv4 host 215B. Ex. 1005, ¶ 142 (step 1012 in Fig. 10). The decapsulation of the IPv6 packet received by the CPI 220E results in the destination address of the received IPv6 packet being *replaced* with (i.e., translated to) an IPv4 destination address. Ex. 1005, ¶¶ 142-143, 137-138; Ex. 1003, ¶ 153. In this way, the CPI 220E regularizes the destination address of the packet forwarded to the IPv4 host 220B. Ex. 1003, ¶¶ 152-153.

With reference to annotated Figure 3 below, Ananda discloses that when the CPI 220E decapsulates the IPv6 packet to produce the IPv4 packet, an IPv4 input handler 322 of the CPI 220E accesses the CPI 220E's IPv4 routing table 324 to determine where to route the IPv4 packet. Ex. 1005, ¶ 142, 90. Thus, Ananda discloses that the CPI 220E accesses its routing table 324 based on the IPv4 destination address of the decapsulated IPv4 packet, before transmitting the decapsulated IPv4 packet to its destination. Ex. 1003, ¶ 154.

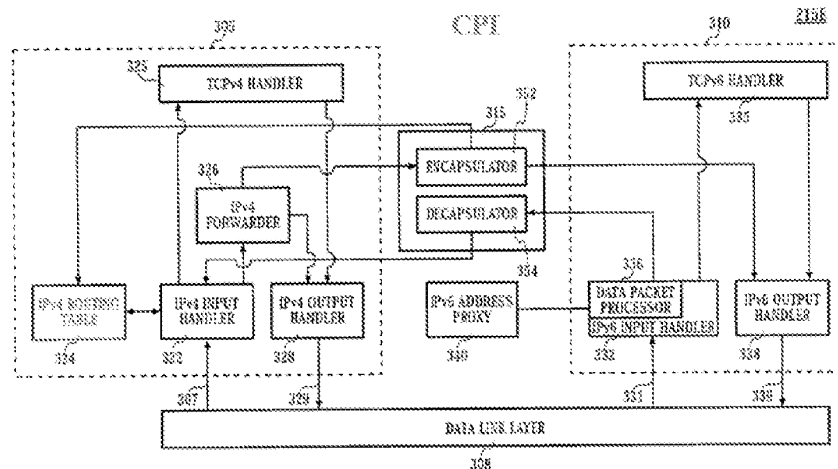


FIG. 3

Ex. 1003, ¶ 154; Ex. 1005, Fig. 3*

Like the '845 patent and Ananda, Wetterwald is directed to providing interoperability between IPv4 nodes and IPv6 nodes across an IPv6 network. Ex. 1008, 1:8-14. With reference to annotated Figure 2 below, Wetterwald discloses that an IPv4 terminal 18a in a first **IPv4 domain 12** transmits an IPv4 packet to another IPv4 terminal 22 in a second **IPv4 domain 16**. Ex. 1008, 4:60- 64, 5:9-13, 6:3-9. Like the '845 patent and Ananda, Wetterwald discloses that the IPv4 packet is translated into an IPv6 packet by an access point 30, the IPv6 packet is transmitted across an **IPv6 domain 14**, and a **translator 50** in clusterhead 32 translates the IPv6 packet to an IPv4 packet before it is routed to the destination IPv4 terminal 22. Ex. 1008, 6:6:1-9, 4:28-56 (describing network configuration), 5:9-21 (assignment of IPv4 and IPv6 addresses), 6:38-60 and 7:37-41 (**translator 50** translates IPv6 packet to IPv4 packet); Ex. 1003, ¶ 155. A **translation resource 42** of the access point 30 of Wetterwald likewise performs IPv4->IPv6 translation, similar to Ananda's CPI 215E. Ex. 1005, ¶ 138; Ex. 1008, 6:1-9, 6:16-22. Thus, like the CPI 220E of Ananda, the clusterhead 32 of Wetterwald receives an IPv6 packet in an **IPv6 domain 14** connected to an **IPv4 domain 16**. Ex. 1003, ¶¶ 155-158.

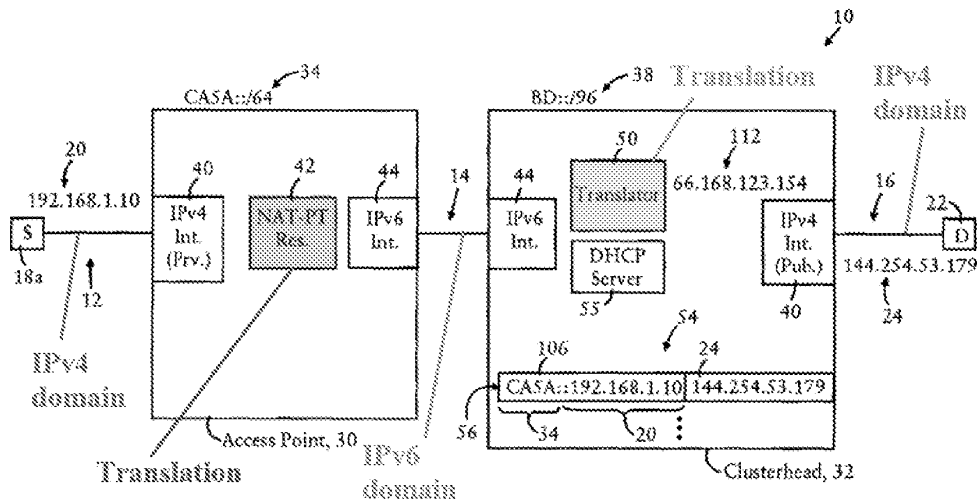


Figure 2

Ex. 1003, ¶ 155; Ex. 1008, Fig. 2*

In addition to performing IPv6->IPv4 translation as shown in annotated Figure 2 above, the **translator 50** also includes an **IPv4 NAT/PAT translation resource 53** that translates between a public IPv4 address and a private IPv4 address, as shown in annotated Figure 5 below. Ex. 1008, 7:1-21. Thus, after the translation resource 51 of the **translator 50** translates a received IPv6 packet into an IPv4 packet (Ex. 1008, 6:38-56, 7:37-41), the **translation resource 53** uses a NAT table 54 to translate between public and private IPv4 addresses, as described in RFC1918 (Ex. 1007). Ex. 1008, 7:1-21, 1:37-40; Ex. 1003, ¶ 159.

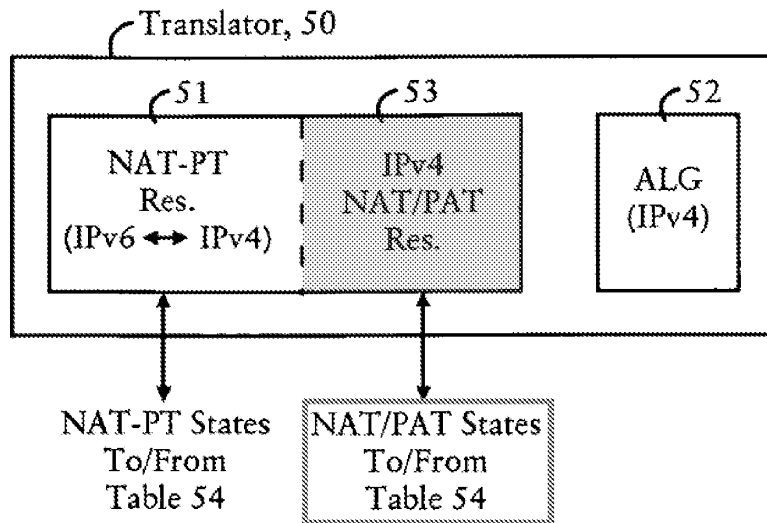
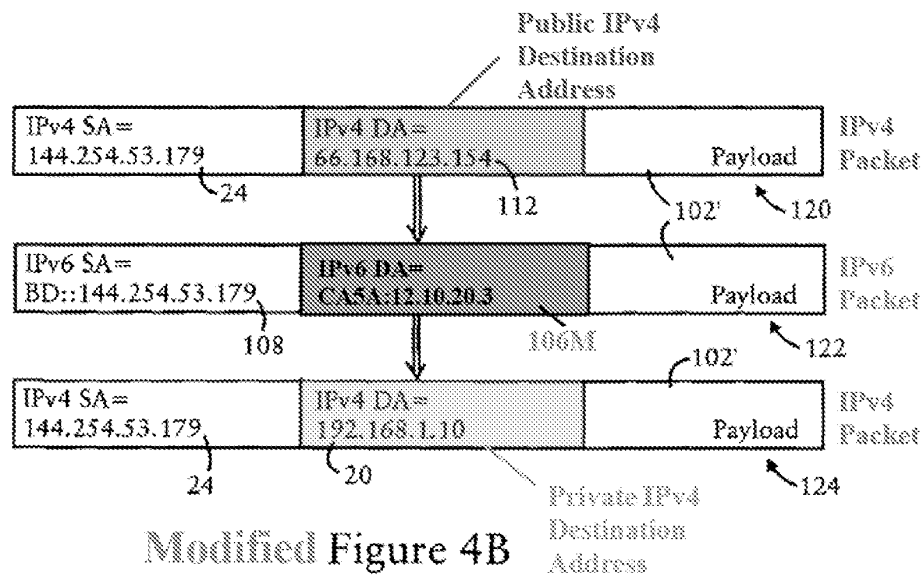


Figure 5

Ex. 1003, ¶ 159; Ex. 1008, Fig. 5*

Using the NAT table 54, Wetterwald's **translator 50** also performs port address translation (PAT). Ex. 1008, 1:7-14. Wetterwald explains that the **translation resource 53** includes PAT functionality that enables the clusterhead 32 to "reuse an assigned public address 112 for multiple connections on the IPv4 network 16," by translating between the assigned public address 112 and a private IPv4 address. Ex. 1008, 7:1-7; Ex. 1003, ¶ 160.

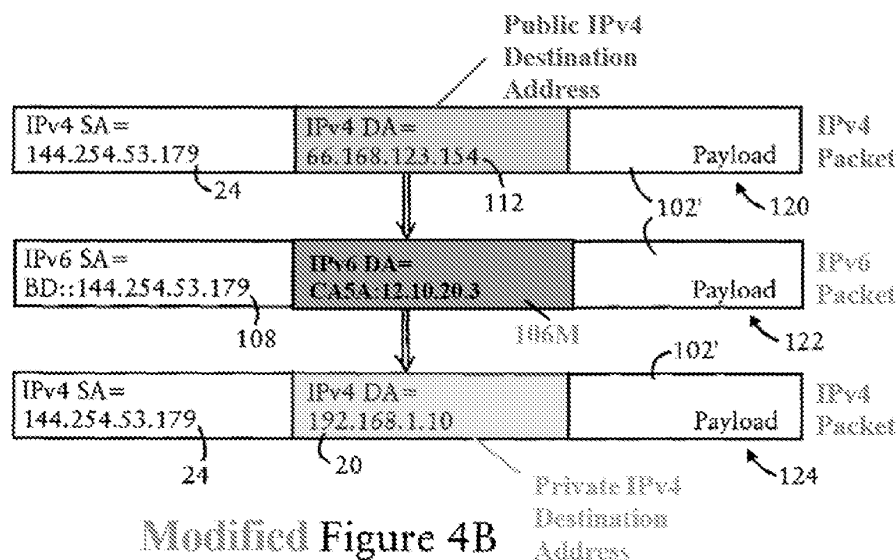
Annotated Figure 4B of Wetterwald below discloses an example of a **public IPv4 destination address 112** (66.168.123.154) contained in the **IPv4 packet 120** that is translated to the **IPv6 packet 122**, and a **private IPv4 destination address 20** contained in the **IPv4 packet 124** that is translated from the **IPv6 packet 122**. Ex. 1008, 6:61-67, 7:30-36 (translation resource 53 has “reverse NAT functionality” corresponding to response packet illustrated in Figure 4B); Ex. 1003, ¶ 160.



Ex. 1003, ¶ 160; Ex. 1008, Fig. 4B*

Wetterwald discloses that the NAT/PAT table entry 54 specifies an association between (1) the translated IPv4 private and public addresses (e.g., an association between **private IPv4 address 20** (192.168.1.10) and **public IPv4 address 112** (66.168.123.154) 112 in Figure 4B), and (2) the originally-specified and translated TCP/UDP port values (e.g., “80” and “1”). Ex. 1008, 7:7-19. Thus, the PAT functionality in the **translation resource 53** enables the gateway to use the same public IPv4 address for 2^{16} (i.e., 65,536) distinct connections. Ex. 1008, 7:7-21; Ex. 1003, ¶ 161.

In Wetterwald's Figure 4B above, which shows the response packet, the **private IPv4 address 20** (192.168.1.10) is concatenated with the IPv6 prefix (CA5A::/64) 34 of the access point 30 (see Figure 2) in the IPv6 destination address 106 of the IPv6 packet 122. Wetterwald explains that the clusterhead 32 previously stored the association between the **private IPv4 address 20** and the IPv6 prefix of the access point 30 when the clusterhead received the IPv6 packet 104 shown in Figure 4A. Ex. 1008, 8:51-62. A POSITA would have been motivated to follow the guidance in RFC1918 that packets with private addresses should not be forwarded over public links (e.g., IPv6 domain 14 in Wetterwald). As discussed above, Wetterwald's **translation resource 53** performs public-to- private IPv4 translations as described in RFC1918. Ex. 1008, 7:1-3; Ex. 1003, ¶ 162. Thus, a POSITA would have been motivated to modify Figure 4B as shown below, where the **modified IPv6 destination address 106M** includes the IPv6 prefix of the access node 30 (CA5A::/64) concatenated with a public IPv4 address (12.10.20.3) assigned to the access node 30,¹⁰ rather than the **private IPv4 address 20** (192.168.1.10) assigned to the IPv4 host 18a, as illustrated in Figure 2; Ex. 1003, ¶¶ 162, 164-165.



Ex. 1003, ¶ 162; Ex. 1008, Fig. 4B*

The public-to-private IPv4 translation of the destination address in the IPv4 packet 124 would then be performed, as shown in modified Figure 4B, and as taught by Wetterwald (i.e., replacement of the public IPv4 destination address (12.10.20.3) assigned to the access node 30

with the private IPv4 address (192.168.1.10) assigned to the IPv4 host 18a). Ex. 1008, 7:1-21; Ex. 1003, ¶ 163.

Ananda is concerned with the shortage of IPv4 addresses. Ex. 1005, ¶ 6. Further, Ananda discloses that the IPv4 host 215B in subnet 215 and IPv4 host 220B in subnet 220 may be assigned private IPv4 addresses. Ex. 1005, ¶¶ 84 (IPv4 hosts may be assigned private IPv4 addresses), 82 (IPv4 host 215B and IPv4 host 220B are “purely IPv4.”). Utilizing the public-to-private IPv4 address translation technique of Wetterwald after the CPI 220E has decapsulated a received IPv6 packet into an IPv4 packet would help mitigate the overuse of public IPv4 addresses, because one public IPv4 address can be used to provide communications for 2^{16} (i.e., 65,536) distinct private IPv4 addresses (“*native address[es]*”). Ex. 1003, ¶¶ 166-169.

A POSITA would have been motivated to modify Ananda to include a NAT table (like Wetterwald’s NAT 54) to associate public and private IPv4 addresses so that the CPI 220E would perform public-to-private IPv4 translation, just like the **translation resource 53** of Wetterwald. This modification would permit IPv4 host 215B and IPv4 host 220B to use private IPv4 addresses (Ex. 1005, ¶ 82 (IPv4 hosts may be assigned private addresses)), and not reveal their private addresses when transmitting packets outside their respective subnet 215, 220, as disclosed by RFC1918. Ex. 1007, § 3, 5 (packets with private source or destination addresses should not be forwarded over public links). Wetterwald’s **translation resource 53** performs public-to-private IPv4 translations as described in RFC1918. Ex. 1008, 7:1-3; Ex. 1003, ¶ 170.

For example, when the IPv4 host 215B transmits an IPv4 packet to IPv4 host 220B (Ex. 1005, ¶ 137), the IPv4 host 215B would transmit the IPv4 packet using a public IPv4 address of the host 220B as the destination address if the IPv4 host 220B is using a private address that is unknown to devices outside the subnet 220 (see Ex. 1007, § 2, 2-3) (private addresses are ambiguous between different enterprises). In this case, the public IPv4 address of the IPv4 host 220B would be the IPv4 address of the CPI 220E, so that the private IPv4 address of the IPv4 host 220B remains private. Ex. 1007, § 3, 5 (packets with private source or destination addresses should not be forwarded over public links); Ex. 1003, ¶ 171.

The CPI 215E, when constructing the IPv6 destination address as described above with respect to element [1.1] (*see, e.g.*, Ex. 1005, ¶¶ 106-107, 105), would then utilize the public IPv4 address of the IPv4 host 220B (i.e., the IPv4 address of the CPI 220E) as the IPv4 destination

address when constructing the IPv6 packet. After decapsulating the received IPv6 packet to produce the IPv4 packet transmitted by the CPI 215E, the CPI 220E would perform public-to-private IPv4 translation, like the **translation resource 53** of Wetterwald, to translate the public IPv4 destination address of the decapsulated IPv4 packet (e.g., the public IPv4 address of the CPI 220E) to the private IPv4 address of the IPv4 host 220B. Ex. 1008, 7:1-21; Ex. 1003, ¶ 172.

Likewise, as explained in Dr. Min's declaration, if IPv4 host 215B is using a private IPv4 address, the CPI 215E may perform a similar private-to-public IPv4 translation, as disclosed by Wetterwald, using the public IPv4 address of the CPI 215E as the source address of the IPv4 packet that the CPI 215E encapsulates into an IPv6 packet. Ex. 1005, ¶ 138; Ex. 1003, ¶ 173. This modification of Ananda's CPI 215E and CPI 220E using their public IPv4 addresses instead of the private IPv4 address of their respective IPv4 hosts 215B, 220B is consistent with RFC1918. Ex. 1007, § 3 (p. 4) (private hosts have access to external services via mediating gateways). Ex. 1003, ¶ 174.

As described above, RFC1918 advises that private addresses should not be forwarded over public links. Ex. 1007, § 3, 5. Based on this guidance, a POSITA would have been motivated to modify the IPv4 routing technique of Ananda to utilize Wetterwald's public-to-private network address translation so that private IPv4 addresses are not utilized in IPv6 packets transmitted over Wetterwald's IPv6 network 14. Ex. 1003, ¶ 175. Further, a POSITA would have been motivated to utilize Wetterwald's public-to-private IPv4 translation in Ananda, because one public IPv4 address in an IPv4 packet received by Ananda's CPI 220E (e.g., an IPv6 packet that the CPI 220E translates into an IPv4 packet) can be used for 2^{16} private addresses for IPv4 hosts in the subnet 220. Ex. 1003, ¶ 176. Ananda explains that the IPv4 hosts (e.g., IPv4 host 220B) are "purely IPv4 hosts" that may be assigned private IPv4 addresses. Ex. 1005, ¶¶ 82, 84. This would have involved the combination of prior art elements (Ananda's use of an IPv4 routing table with Wetterwald's NAT/PAT translation resource 53) according to known methods (translating an IPv4 address in the header of an IPv4 packet decapsulated by the CPI 220E into one of 2^{16} private IPv4 addresses) to yield the predictable result of translating a public IPv4 address into a private IPv4 address, as taught by Wetterwald. *See KSR*, 550 U.S. at 401; Ex. 1003, ¶ 176.

Furthermore, the obvious combination of Ananda and Wetterwald would not have required any modification to Ananda's network architecture because Ananda discloses that IPv4 host 215B

in subnet 215 and IPv4 host 220B in subnets 220 may be assigned private addresses. Ex. 1005, ¶ 84; Ex. 1003, ¶ 177.

As described above with respect to element [1.1], Wetterwald teaches that it was known, when encapsulating IPv4 packets in IPv6 packets, to include IPv6 extension headers to identify the source and destination port numbers (Ex. 1008, 8:24-28), and RFC 4380 discloses that it was known to include a destination port number in a constructed IPv6 address. Ex. 1009, 22-23. Wetterwald discloses that the translation resource 53 translates between public and private IPv4 addresses based on port values specified in the IPv4 packet. Ex. 1008, 7:7-19. Therefore, a POSITA would have had an expectation of success in modifying Ananda's CPI 220E to perform a public-to-private IPv4 translation after decapsulating a received IPv6 packet, because Wetterwald discloses that the public-to-private IPv4 translations are performed based on port values specified in the IPv4 packet. Ex. 1008, 7:7-19; Ex. 1003, ¶ 178. As described above with respect to element [1.1], it would have been obvious to modify Ananda to utilize an IPv6 destination address that is constructed by concatenating an IPv6 prefix, an IPv4 destination address, and a destination port number, as taught by Wetterwald and RFC 4380.

Thus, a POSITA would have had a reasonable expectation of success in making the above modifications to Ananda because the public-to-private IPv4 translation would be performed based on the port values included in the packet, as taught by Wetterwald. Ex. 1008, 7:7-19; Ex. 1003, ¶ 179.

Thus, the prior art renders obvious element [1.2.1]. Ex. 1003, ¶ 180.

f. [1.2.2] “the replacement process belonging to a group comprising replacement of a constructed address with a native address and replacement of a native address with a constructed address, and”

Ananda in view of Wetterwald renders obvious element [1.2.2]. Ex. 1003, ¶ 181.

Initially, it is noted that element [1.2.2] recites two features in the alternative by reciting the phrase “belonging to a group comprising...” Thus, element [1.2.2] requires that the replacement process of element [1.2.1] comprises (1) replacement of a constructed address with a native address, or (2) replacement of a native address with a constructed address. As described above, Ananda and Wetterwald render obvious element [1.2.1]. Ex. 1003, ¶¶ 182.

As described above with respect to element [1.2.1], Ananda discloses that when the CPI

220E decapsulates the IPv6 packet (step 1012 in Fig. 10) received from the CPI 215E, the CPI 220E replaces the IPv6 destination address with the IPv4 destination address of the IPv4 host 220B. Ex. 1005, ¶ 142; Ex. 1003, ¶¶ 183.

The IPv6 destination address contained in the IPv6 packet received from the CPI 215E is a constructed address. Ex. 1003, ¶¶ 184. In particular, the IPv6 destination address in the IPv6 packet is an IPv6 virtual address of IPv4 host 220B (Ex. 1005, ¶ 138) that is constructed in steps 427, 428, 435 and 440 of annotated Figure 4 below. Ex. 1005, ¶¶ 106-107, 105; Ex. 1003, ¶¶ 184-185.

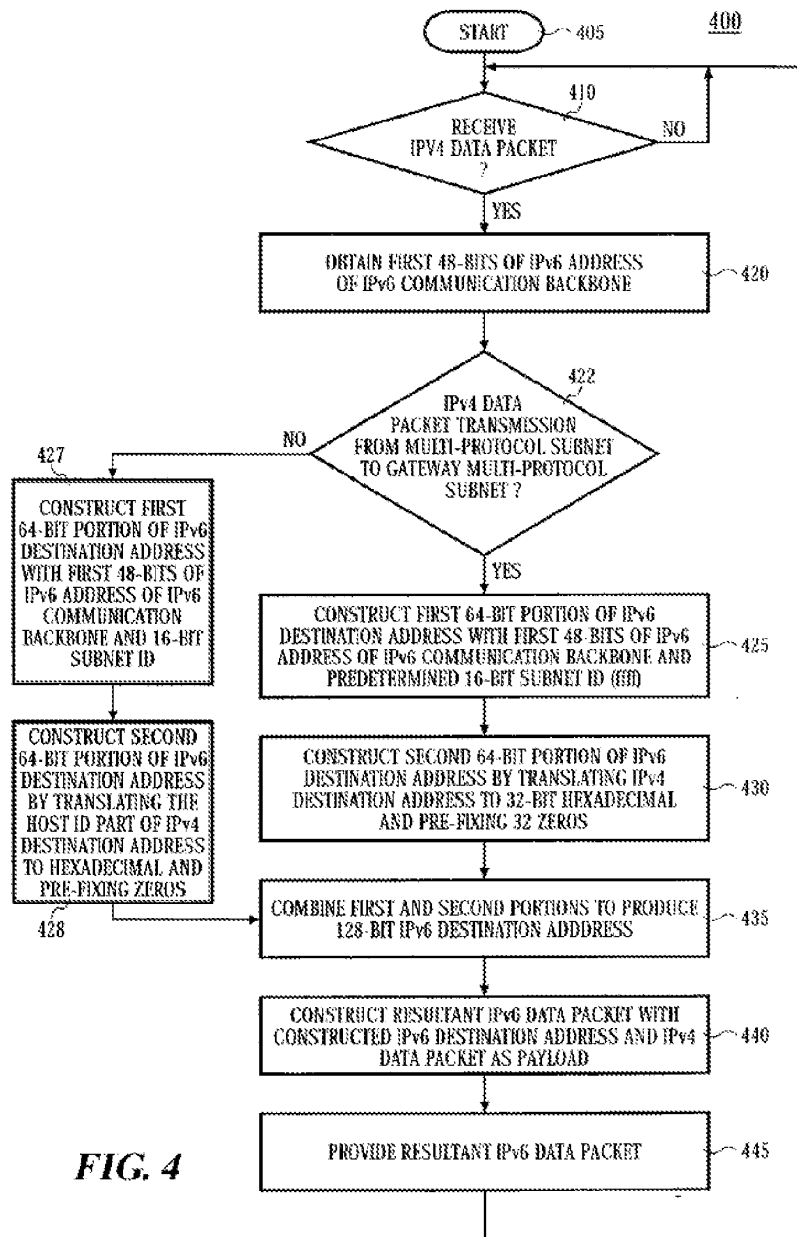


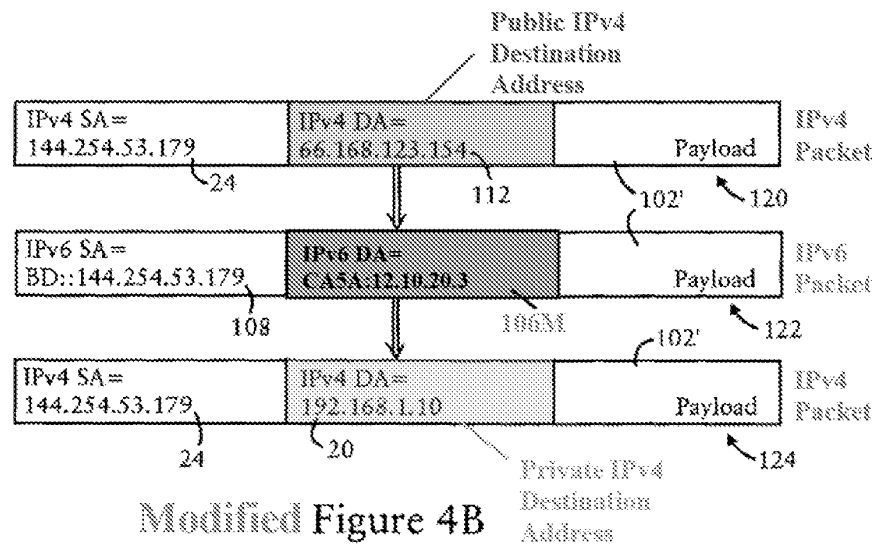
FIG. 4

Ex. 1003, ¶ 185; Ex. 1005, Fig. 4*

As described above with respect to element [1.2.1], a POSITA would have been motivated to modify Ananda such that the CPI 220E, after decapsulating the received IPv6 packet to produce the original IPv4 packet (Ex. 1005, ¶ 142) would use a translation resource, like Wetterwald's translation resource 53, to translate the public IPv4 destination address in the decapsulated IPv4

packet to a private IPv4 address of the IPv4 host 220B. Ex. 1003, ¶ 186.

This obvious modification would involve replacement of the constructed IPv6 address (“*constructed address*”) contained in the IPv6 packet received by the CPI 220E with the private IPv4 address (“*native address*”) of the IPv4 host 220B. Annotated Figure 4B of Wetterwald below, as modified as described above with respect to element [1.2.1], illustrates replacement of a **constructed IPv6 destination address 106M** with a **private IPv4 destination address 20** in the **IPv4 packet 124** that is translated from the **IPv6 packet 122**. Ex. 1008, 6:50-54, 6:61-67, 9:11-32. As shown in annotated Figure 4B, the **private IPv4 destination address 20** in the **IPv4 packet 124** differs from the **public IPv4 destination address 112** contained in the **original IPv4 packet 120** that was translated into the **IPv6 packet 122**. Ex. 1008, 6:61-67; Ex. 1003, ¶ 187.



Ex. 1003, ¶ 187; Ex. 1008, Fig. 4B*

Accordingly, Ananda in view of Wetterwald renders obvious the claimed feature of the replacement process comprising *replacement of a constructed address with a native address*, as recited in element [1.2.2]. Ex. 1003, ¶ 188.

g. [1.2.3] “*modifying the data packet using the regularized address; and*”

Ananda and Wetterwald render obvious element [1.2.3]. Ex. 1003, ¶ 189.

As described above with respect to element [1.2.1], a POSITA would have been motivated

to modify Ananda's CPI 220E to use a translation resource to translate the public IPv4 destination address in the decapsulated IPv4 packet to a private IPv4 address of the IPv4 host 220B. Ex. 1003, ¶ 190.

Ananda discloses that when the CPI 220E receives the IPv6 packet transmitted by the CPI 215E, the decapsulator 354 of the CPI 220E decapsulates the IPv6 header to produce the original IPv4 data packet. Ex. 1005, ¶ 142. Annotated Figure 3 below shows the decapsulator 354 of the CPI 220E. Ex. 1005, ¶¶ 142, 102; Ex. 1003, ¶ 191.

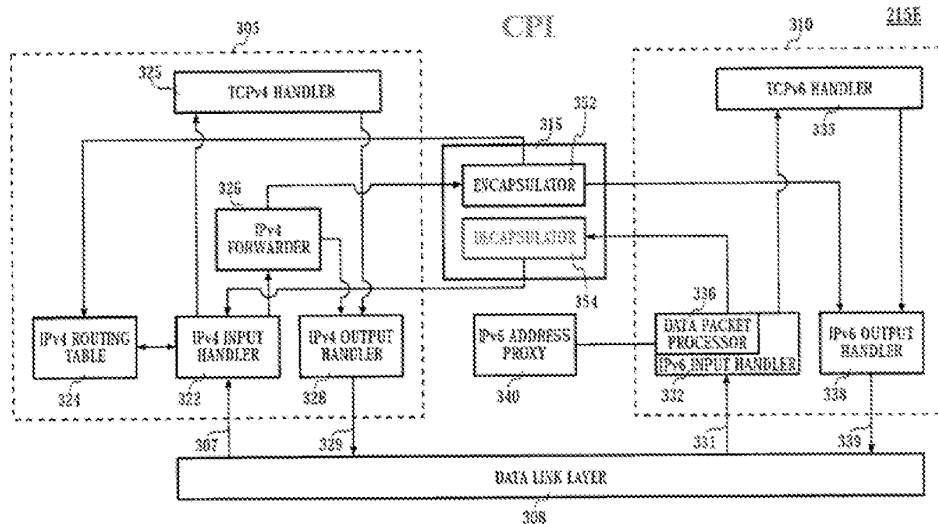


FIG. 3

Ex. 1003, ¶ 191; Ex. 1005, Fig. 3*

Annotated Figure 5 of Ananda below shows the decapsulation process performed by the decapsulator 354. Ananda explains that extracting (step 515) an IPv4 data packet from the payload of an IPv6 data packet involves stripping the IPv6 header from the IPv6 data packet. Ex. 1005, ¶ 109. Stripping the IPv6 header from the packet results in modifying the packet. Ex. 1003, ¶ 192.

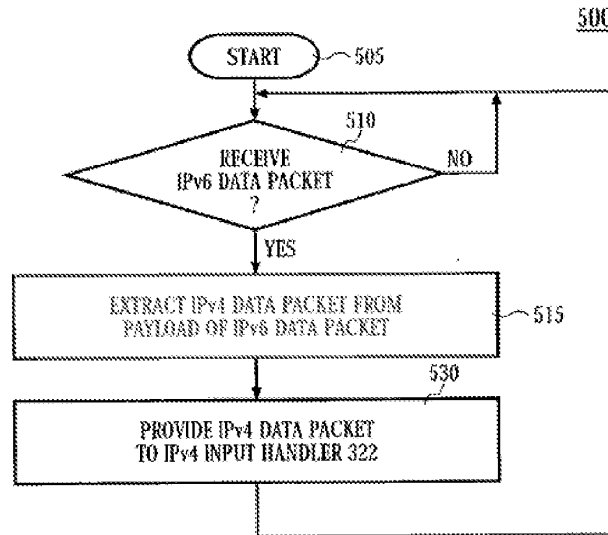
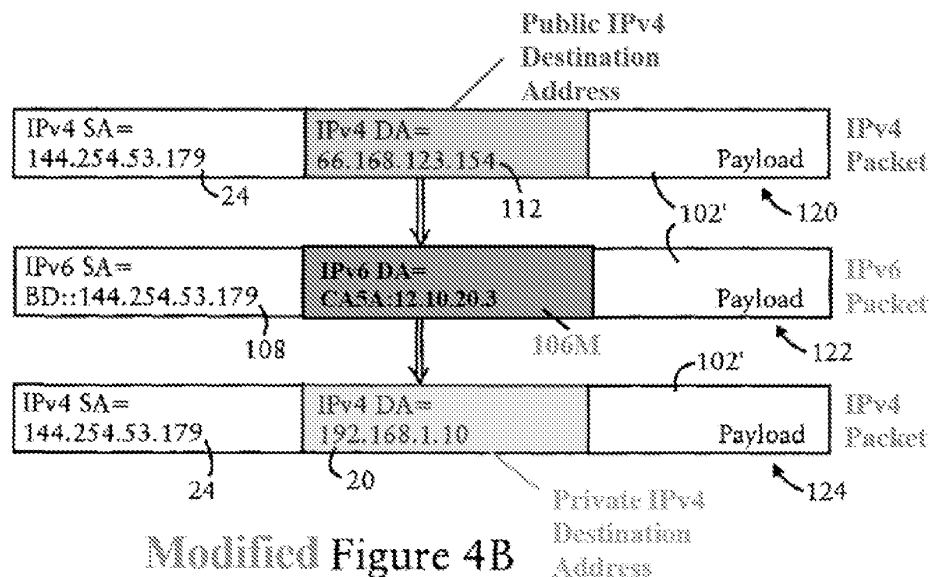


FIG. 5

Ex. 1003, ¶ 192; Ex. 1005, Fig. 4*

Replacing the IPv4 destination address in the decapsulated IPv4 packet with the private IPv4 address of the IPv4 host 220B involves modifying the IPv6 packet with the private IPv4 address of the IPv4 host (“*regularized address*”) because the destination address of the IPv6 packet would be replaced with the private IPv4 address of the IPv4 host 220B, as explained above with respect to element [1.2.2]. For example, as shown in annotated Figure 4B below, which is modified as described above with respect to element [1.2.1], utilizing Wetterwald’s translation resource 53 in Ananda’s CPI 220E would involve (1) stripping the IPv6 header, as taught by Ananda, and (2) replacing the IPv4 destination address in the decapsulated IPv4 packet with the private IPv4 destination address of the destination terminal (e.g., IPv4 host 220B), as taught by Wetterwald. Ex. 1005, ¶¶ 142, 109; Ex. 1008, 6:61-67, 9:11-32; Ex. 1003, ¶ 193.



Ex. 1003, ¶ 193; Ex. 1008, Fig. 4B*

As shown in annotated Figure 4B of Wetterwald above, the **private IPv4 destination address 20** (e.g., the private IPv4 address of Ananda's IPv4 host 220B) in the **IPv4 packet 124** differs from the **public IPv4 destination address 112** (e.g., the public IPv4 address of Ananda's CPI 220E) contained in the **original IPv4 packet 120** that was translated into the **IPv6 packet 122**. Ex. 1008, 6:61-67; Ex. 1003, ¶ 194.

Thus, Ananda in view of Wetterwald renders obvious element [1.2.3]. Ex. 1003, ¶ 195.

h. [1.3] "routing the IPv6 data packet to its destination."

Ananda renders obvious element [1.3]. Ex. 1003, ¶ 196.

As shown in annotated Figure 10 below, Ananda discloses that the CPI 220E receives the IPv6 packet transmitted from CPI 215E as it is routed through the IPv6 backbone 210 by routers 210A and 210B. Ex. 1005, ¶¶ 138-142 (steps 1006, 1008 and 1010 in Fig. 10). Then, the CPI 220E decapsulates the IPv6 header to produce the IPv4 packet that was encapsulated in the IPv6 packet. Ex. 1005, ¶ 142 (step 1012 in Fig. 10). The CPI 220E then routes the IPv4 packet to the IPv4 host 220B, which is the *destination* of the packet. Ex. 1005, ¶ 143 (step 1014); Ex. 1003, ¶ 197.

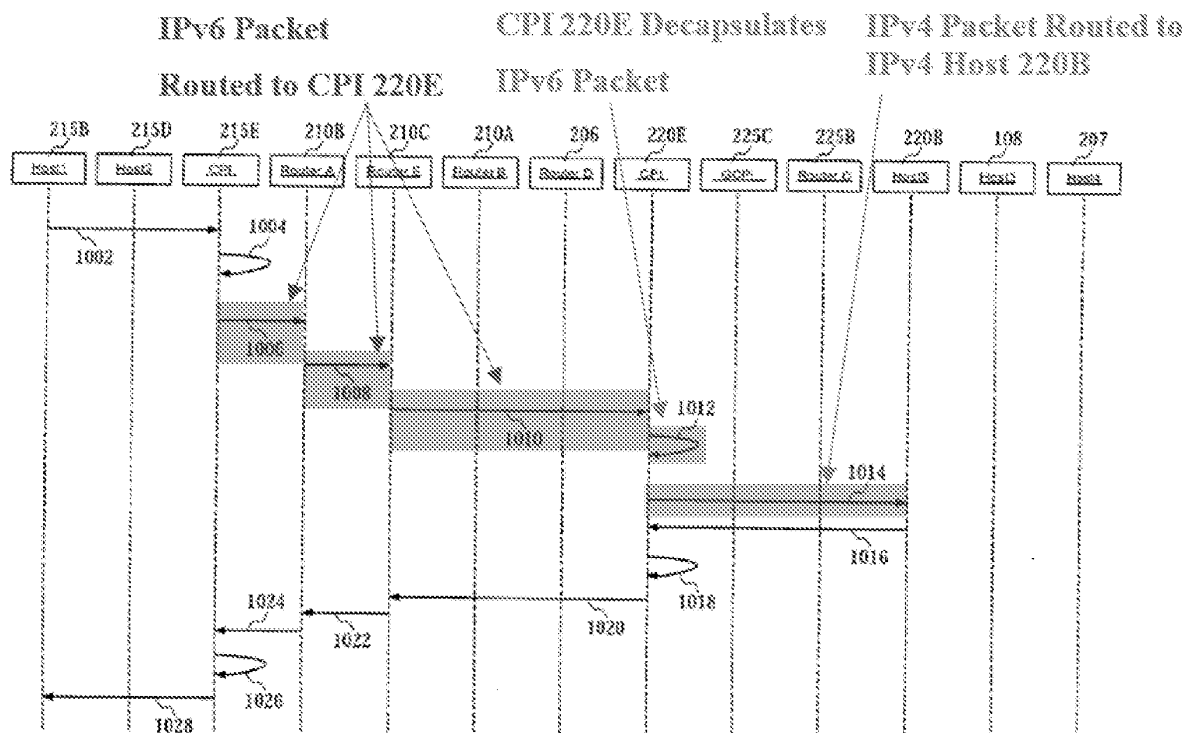


FIG. 10

Ex. 1003, ¶ 197; Ex. 1005, Fig. 10*

By receiving the IPv6 packet, translating the IPv6 packet into an IPv4 packet, and routing the translated IPv4 packet to its destination (the IPv4 host 220B), the CPI 220E routes the IPv6 data packet to its destination. Therefore, Ananda renders obvious element [1.3]. Ex. 1003, ¶ 198.

2. Claim 5

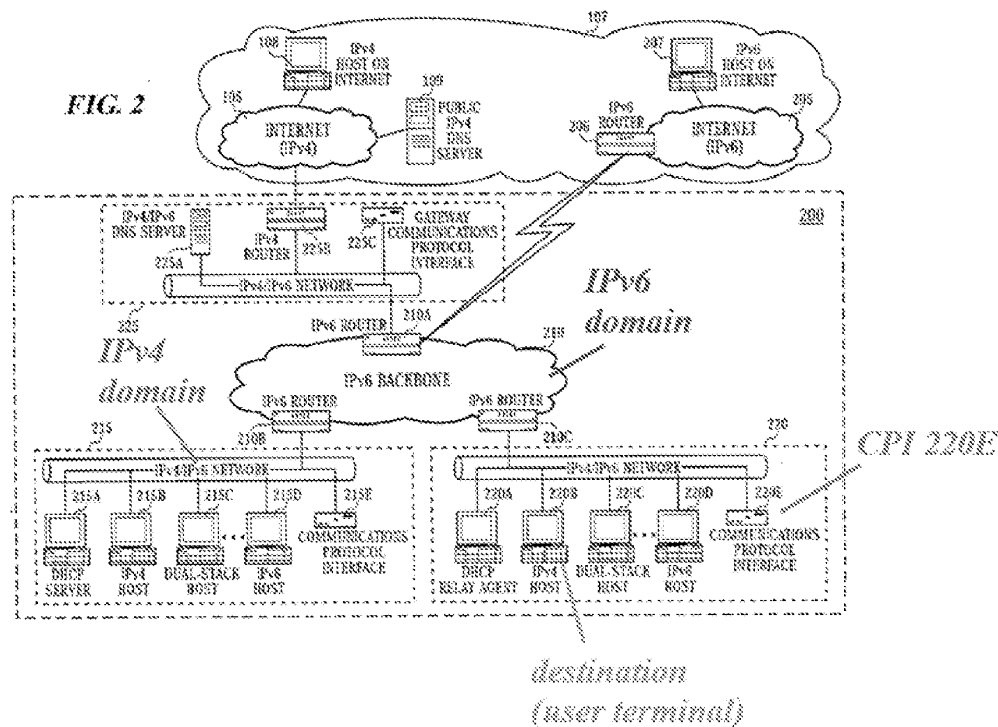
- a. [5.0] “The method according to claim 1, wherein when the data packet comes from the IPv4 domain and goes to said user terminal, the step of regularizing the destination address comprises the following steps:”

Ananda in view of Wetterwald renders obvious element [5.0]. Ex. 1003, ¶199.

With reference to annotated Figure 2 below, Ananda discloses that the CPI 220E of multi-protocol IPv4/IPv6 subnet 220 receives an IPv6 packet in an IPv6 backbone 210 (“IPv6 domain”), which is connected to a multi-protocol IPv4/IPv6 subnet 215 (“IPv4 domain”), where the IPv6 packet was originally transmitted as an IPv4 packet from the IPv4 host 215B in the subnet 215

("IPv4 domain"). Ex. 1005, ¶¶ 137-142, 79. When the CPI 220E receives an IPv6 packet from CPI 215E, the CPI 220E translates the IPv6 packet into an IPv4 packet and transmits the IPv4 packet to IPv4 host 220B ("user terminal"), the destination of the IPv4 packet transmitted by the CPI 215B. Ex. 1005, ¶¶ 137, 142; Ex. 1003, ¶¶ 200-202.

Thus, Ananda renders obvious element [5.0]. Ex. 1003, ¶ 203.



Ex. 1003, ¶ 200; Ex. 1005, Fig. 2*

- b. [5.1] "searching an address translation table of the gateway for an entry associating with said constructed destination address a native destination address;"

As described above with respect to elements [1.2.1]-[1.2.3], a POSITA would have been motivated to modify Ananda so that the CPI 220E, after decapsulating the received IPv6 packet, utilizes a translation resource to replace a public IPv4 destination address in the decapsulated IPv4 packet with a private IPv4 destination address (e.g., the private IPv4 address of the IPv4 host 220B). Ex. 1003, ¶ 207. Wetterwald describes that the translation resource 53 uses a NAT table 54 (*address translation table of the gateway*) to translate between a public IPv4 address and a private

IPv4 address, as shown in annotated Figure 5 below. Ex. 1008, 7:1-21. Ex. 1003, ¶ 208.

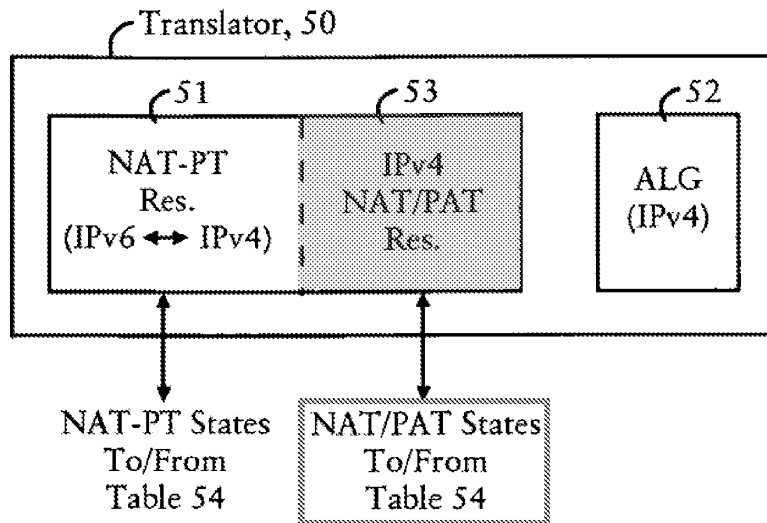


Figure 5

Ex. 1003, ¶ 208; Ex. 1008, Fig. 5*

In the combination, an address translation table (like Wetterwald’s translation table 54) would be incorporated into Ananda’s CPI 220E. The translation table would include an association between private IPv4 addresses (“*native destination address*”) and public IPv4 addresses contained in constructed IPv6 addresses, allowing Ananda’s CPI 220E to perform the public-to-private translation of IPv4 addresses as taught by Wetterwald’s **translation resource 53**. Ex. 1008, 7:1-21. In the combination, the constructed IPv6 addresses received by CPI 220E correspond to the claimed “*said constructed destination address*.” Ex. 1003, ¶¶ 209, 204-206.

Accordingly, Ananda in view of Wetterwald renders obvious element [5.1]. Ex. 1003, ¶ 210.

- c. [5.2] “*if an entry is found, replacing the constructed destination address of the data packet with the native destination address from that entry.*”

Ananda and Wetterwald render obvious element [5.2]. Ex. 1003, ¶ 211.

As described above with respect to elements [1.2.1] and [5.2], Wetterwald discloses that the **translation resource 53** uses its NAT table 54 to translate between public and private IPv4 addresses. Ex. 1008, 7:1-21; Ex. 1003, ¶ 212.

As described above with respect to elements [1.1] and [1.2.2], Ananda discloses that the IPv6 packet is constructed to include hexadecimal translations of the IPv4 destination address contained in the original IPv4 packet. Ex. 1005, ¶¶ 106-108, 105 (above-described construction shown in Figure 4B). Further, as described above with respect to element [1.2.1], Ananda's CPI 220E would utilize a translation table, similar to Wetterwald's NAT 54 shown in annotated Figure 5 below, to perform the public-to-private IPv4 translation after the CPI 220E has decapsulated the IPv6 packet to produce the IPv4 packet. Ex. 1005, ¶ 142; Ex. 1008, 6:38-67; Ex. 1003, ¶ 213.

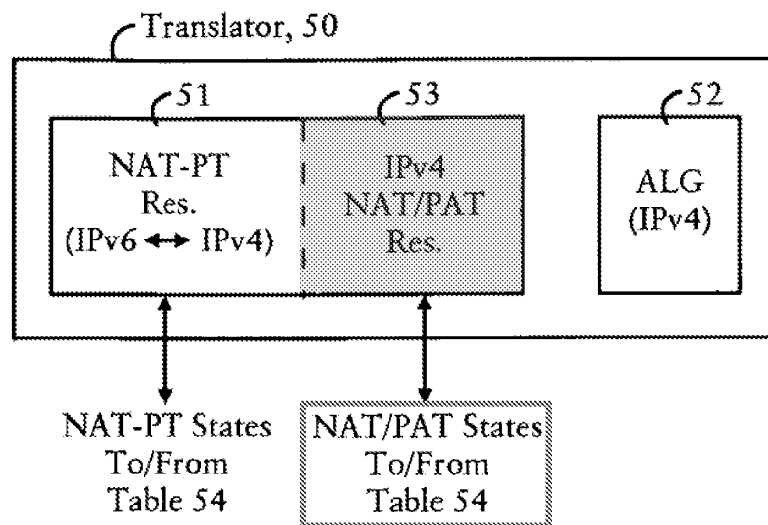


Figure 5

Ex. 1003, ¶ 213; Ex. 1008, Fig. 5*

Therefore, in the obvious modification of Ananda's CPI220E using a translation resource, Ananda's CPI 220E would replace the public IPv4 address contained in the constructed IPv6 packet with a private IPv4 address ("native address") of the destination terminal (e.g., IPv4 host 220B), as taught by Wetterwald's public-to-private IPv4 translation. Ex. 1003, ¶¶ 214-215.

Thus, Ananda in view of Wetterwald renders obvious element [5.2]. Ex. 1003, ¶ 216.

3. Claim 6

- a. [6.0] *“The method according to claim 5, further comprising a step of verifying that the destination port number of the data packet and the destination port number contained in the constructed destination address match.”*

Ananda in view of Wetterwald and RFC 4380 renders obvious element [6.0]. Ex. 1003, ¶ 217.

As discussed with respect to element [1.2.1], it would have been obvious to a POSITA to compare the destination address and port number values within a constructed IPv6 destination address to the corresponding values in the header of an encapsulated IPv4 packet. Ex. 1003, ¶ 218. Such a comparison would be useful, for example, in detecting “spoofed” packets. Ex. 1009, 22-23; Ex. 1003, ¶ 219. By teaching to compare destination address and port number values of an encapsulated IPv4 packet with the corresponding values in a constructed IPv6 address, the prior art renders obvious the claimed “verifying” step. Ex. 1003, ¶ 220.

4. Claim 7

- a. [7.0] *“A non-transitory computer program product that is at least one of downloadable from a communications network and stored on a computer-readable medium and executable by a microprocessor, comprising program code instructions for executing the method according to claim 1 of receiving a data packet in an IPv6 domain.”*

As described above in Section V.A.1.a (element [1.pre.3]), the CPI 220E performs the features of the method of claim 1. Ex. 1003, ¶¶ 221-222. The CPI 220E “comprises an IPv4 processor for processing IPv4 specific communication, an IPv6 processor for processing IPv6 specific communication, and a protocol converter for processing communication between the IPv4 and IPv6 processors.” Ex. 1005, ¶¶ 19, 63. Therefore, the CPI comprises a *microprocessor*. Ex. 1003, ¶¶ 223-224.

Further, Ananda discloses that the CPI comprises various memories. Ex. 1005, claims 6, 9 and 13. The memories of Ananda correspond to a *computer-readable medium*. Ex. 1003, ¶ 225.

Ananda discloses that the CPIs may be “implemented as software modules within a destination host, a stand-alone unit coupled to a subnet and an IPv6 network, ...or be incorporated in a network interface card (NIC).” Ex. 1005, ¶ 118. Thus, Ananda discloses executing software (“*program code instructions*”) to perform the functions of the CPI 220E. Ex. 1003, ¶¶ 226-227.

As discussed generally regarding claim 1, Ananda, Wetterwald, and RFC 4380 render obvious *a method for receiving an IPv6 data packet in an IPv6 domain connected to an IPv4 domain*. See *supra* Section X.E.5; Ex. 1003, ¶ 228.

Wetterwald discloses that the steps for routing data packets in an IPv6 network can be implemented as “executable code stored on a computer readable medium (e.g., floppy disk, hard disk, EEPROM, CD-ROM, etc.)” Ex. 1008, 7:55-65. It would have been obvious to a POSITA to implement the teachings of the prior art in this way so that the executable code would be executed by the processors of Ananda’s CPI 220E and thus perform the described packet routing techniques. Ex. 1003, ¶¶ 229-230. Accordingly, in view of the disclosure of Wetterwald, a POSITA would have been motivated to modify Ananda to provide a computer-readable medium that stores a computer program product executed by the processors of the CPI 220E, to implement the functions of Ananda’s CPI 220E to receive a data packet in an IPv6 domain. Ex. 1003, ¶ 231.

Therefore, Ananda, Wetterwald, and RFC 4380 render obvious element [7.0]. Ex. 1003, ¶ 232.

5. Claim 8

a. [8.pre.1] “A device for receiving an IPv6 data packet in an IPv6 domain connected to an IPv4 domain,

Ananda renders obvious element [8.pre.1]. Ex. 1003, ¶ 233.

Limitation [8.pre.1] is identical to element [1.pre.1] above, except that element [8.pre.1] recites a “device” instead of a “method.” Ananda discloses a device (CPI 220E) for receiving an IPv6 data packet, where the CPI “comprises an IPv4 processor for processing IPv4 specific communication, an IPv6 processor for processing IPv6 specific communication, and a protocol converter for processing communication between the IPv4 and IPv6 processors.” Ex. 1005, ¶¶ 19, 63. As described above with respect to element [1.pre.1], Ananda’s CPI 220E receives an IPv6 data packet in an IPv6 domain connected to an IPv4 domain. Ananda’s CPI 220E is a device. Ex. 1003, ¶ 234.

Ananda describes the structural components of the CPI 215E of subnet 215 in Figure 3 below. Ex. 1005, ¶¶ 89-102. However, the description of the structural elements of the CPI 215E applies equally to the CPI 220E of subnet 220. Ex. 1003, ¶ 235. For instance, Ananda explains that the CPI 220E uses “its decapsulator 354” (Ex. 1005, ¶ 142) and “its encapsulator 352” (Ex.

1005, ¶ 144), which refers to the structural components of the CPI 215E illustrated in Figure 3. Dr. Min's declaration explains in more detail why the CPI 220E has the same structural configuration as the CPI 215E illustrated in Figure 3. Ex. 1003, ¶ 235.

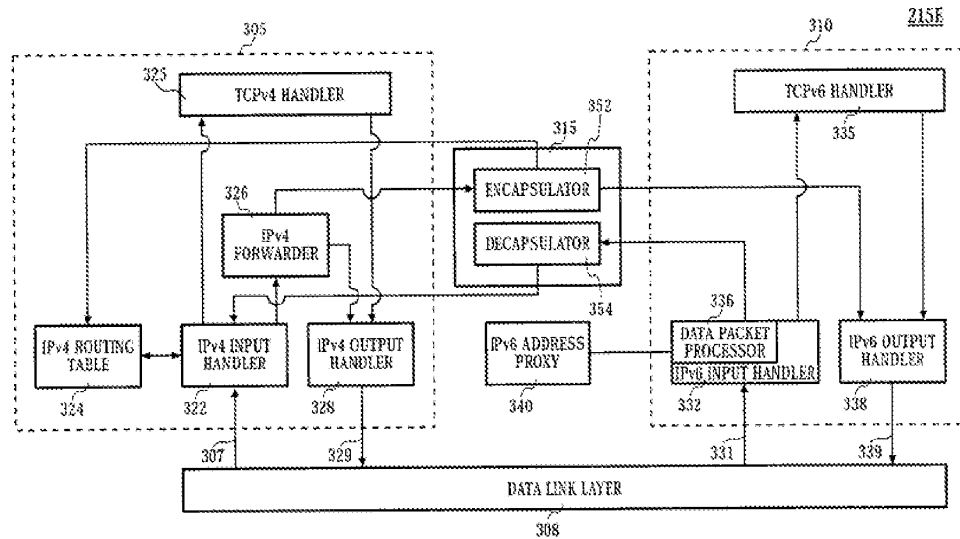
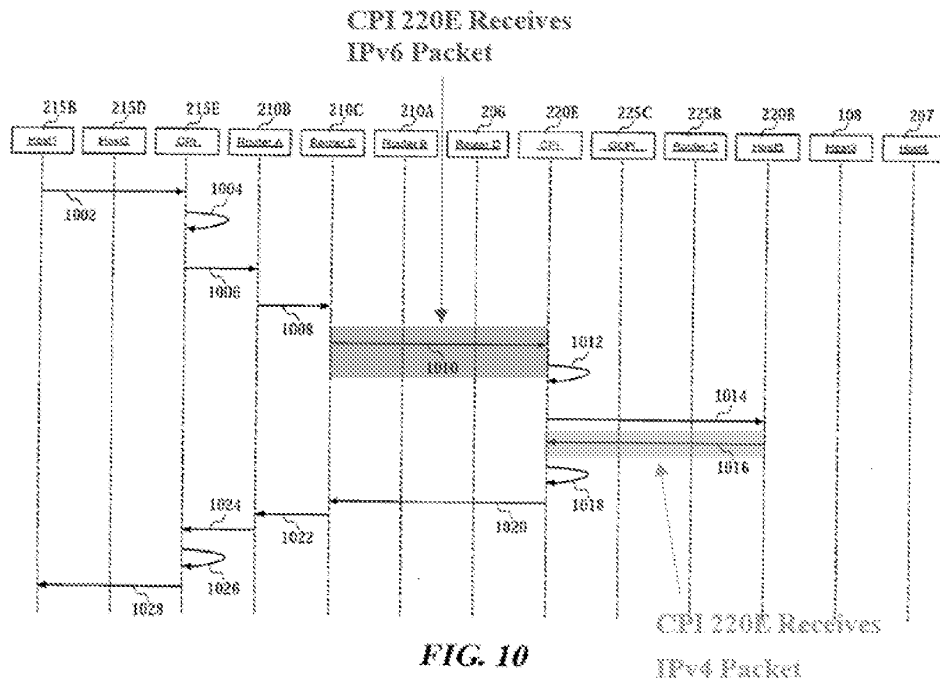


FIG. 3

Ex. 1005, Fig. 3

Because CPI 220E receives IPv6 packets in an IPv6 network (step 1010 in annotated Figure 10 below) and receives IPv4 packets in an IPv4 network (step 1016 in annotated Figure 10 below), CPI 220E is assigned both IPv4 and IPv6 addresses, which provides further confirmation that CPI 220E is a device. For example, Ananda explains that CPI 220E is assigned an IPv6 address of 2001:208:3:58:202:2dff:fe0D:4561. Ex. 1005, ¶ 145; Ex. 1003, ¶ 236.



Ex. 1003, ¶ 236; Ex. 1005, Fig. 4*

Thus, Ananda renders obvious element [8.pre.1]. Ex. 1003, ¶ 237.

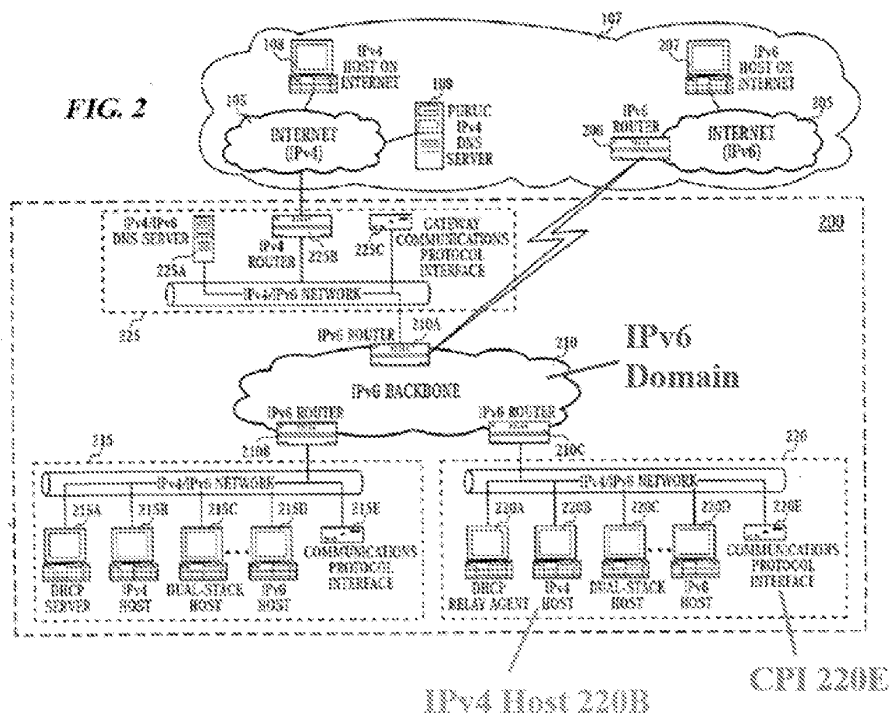
- b. [8.pre.2] “said packet comprising an IPv6 destination address and an IPv6 source address,”**

See analysis of element [1.pre.2], which recites an identical element. Ex. 1003, ¶ 238.

- c. [8.pre.3] “said device for use in a home gateway adapted to connect a user terminal to said IPv6 domain, comprising:”**

Limitation [8.pre.3] is identical to element [1.pre.3] above, except that element [1.pre.3] recites that the method is executed in a home gateway, whereas element [8.pre.3] recites “said device for use in a home gateway.” Ex. 1003, ¶ 239.

As described above with respect to element [8.pre.1], the CPI 220E of Ananda is a *device*. Further, as described above with respect to element [1.pre.3], Ananda’s **CPI 220E** is a home gateway adapted to connect **IPv4 host 220B** (“*user terminal*”) to communicate with **IPv6 backbone 210** (“*IPv6 domain*”), as shown in annotated Figure 2 below. Ex. 1005 ¶¶ 79, 77-78, 63; Ex. 1003, ¶ 240.



Ex. 1003, ¶ 240; Ex. 1005, Fig. 2*

Therefore, Ananda renders obvious element [8.pre.3]. Ex. 1003, ¶ 241.

- d. [8.1] “means for identifying an IPv6 destination address constructed by concatenating an IPv6 prefix, an IPv4 destination address, and a destination port number;”

Element [8.1] is identical to element [1.1], except that element [8.1] recites “means for identifying an IPv6 destination address....,” whereas element [1.1] recites “identifying an IPv6 destination address....” Ananda in view of Wetterwald and RFC 4380 render obvious element [1.1]. Ex. 1003, ¶ 242.

Ananda’s CPI 220E is a receiver device because it includes an **IPv4 processor 305** having an IPv4 input handler 322, and an **IPv6 processor 310** having an IPv6 input handler 332, as shown in annotated Figure 3 below. Ex. 1005, ¶¶ 89-90, 94. Ananda discloses that the CPI comprises various memories. Ex. 1005, claims 6, 9 and 13. The CPI 220E has the same structure for identifying an IPv6 destination address (i.e., a receiver device having a processor and memory) for element [8.1], as discussed above in Section VIII.C.1. Ex. 1003, ¶¶ 243-244.

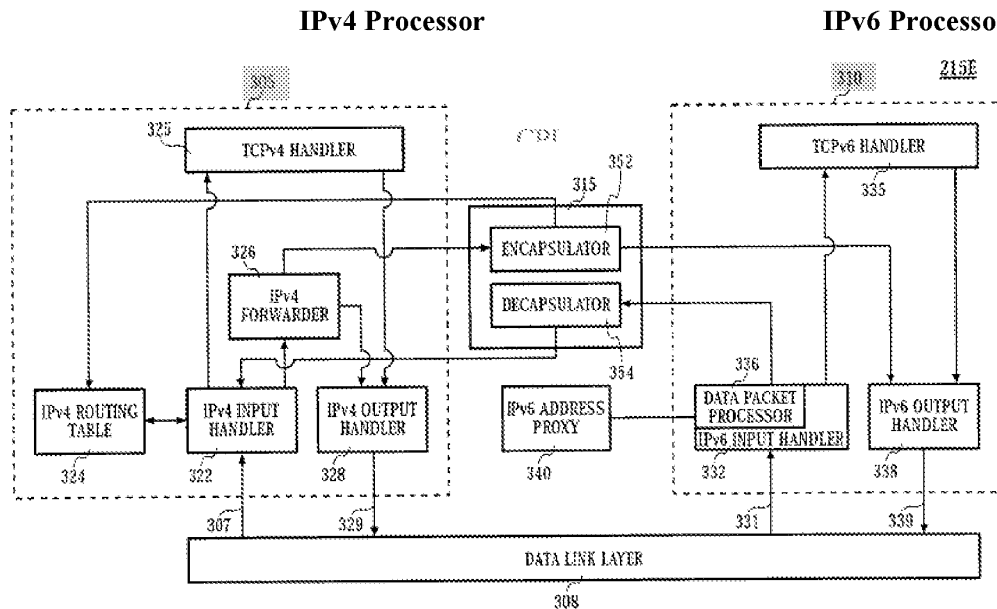


FIG. 3

Ex. 1003, ¶ 244; Ex. 1008, Fig. 3*

Thus, Ananda in view of Wetterwald and RFC 4380 renders obvious element [8.1]. Ex. 1003, ¶ 245.

- e. [8.2.1] “means for regularizing, if necessary, at least one of the IPv6 source address and the IPv6 destination address of the data packet by replacing said at least one IPv6 address,”

Ananda in view of Wetterwald renders obvious element [8.2.1]. Ex. 1003, ¶246.

Element [8.2.1] is identical to element [1.2.1], except that element [8.2.1] recites “means for regularizing ...,” whereas element [1.2.1] recites the method step of “regularizing.” Ex. 1003, ¶ 247.

As discussed above with respect to element [1.2.1], Ananda’s CPI 220E, modified to include NAT features as taught by Wetterwald, regularizes the IPv6 destination address in the received IPv6 packet by replacing the IPv6 destination address with a private IPv4 destination address. The modified CPI 220E would perform a public-to-private IPv4 address translation, like Wetterwald’s translation resource 53, and thus the CPI 220E would include a NAT table stored in the modified CPI 220E’s memory similar to the translation resource 53 as shown in annotated Figure 5 of Wetterwald below. Ex. 1008, 6:38-49; Ex. 1003, ¶ 248.

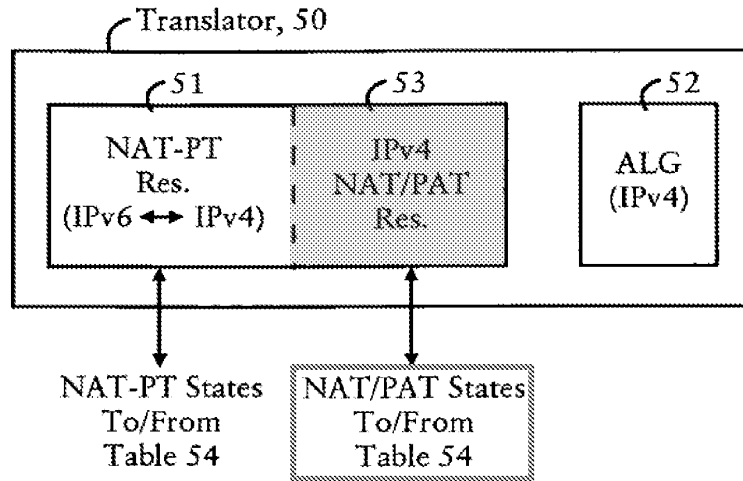


Figure 5

Ex. 1003, ¶ 248; Ex. 1008, Fig. 5*

The modified CPI 220E thus has the same structure as the “means for regularizing” element (i.e., a receiver device having a processor and memory that stores a NAT for translating between addresses), as discussed above in Section VIII.C.2. Ex. 1003, ¶¶ 249-251.

Thus, Ananda in view of Wetterwald renders obvious element [8.2.1]. Ex. 1003, ¶ 252.

- f. [8.2.2] “the replacement process belonging to a group comprising replacement of a constructed address with a native address and replacement of a native address with a constructed address, and”

See analysis of element [1.2.2] above, which recites an identical element. Ex. 1003, ¶ 253.

- g. [8.2.3] “[means for...] modifying the data packet using the regularized address; and”

Element [8.2.3] is identical to element [1.2.3], except that element [8.2.3] is part of the “means for regularizing” element. Ex. 1003, ¶ 254. As described above with respect to element [8.2.1], the modified CPI 220E of Ananda has the same structure as the “means for regularizing” element. Ex. 1003, ¶ 255.

Thus, Ananda in view of Wetterwald renders obvious element [8.2.3]. Ex. 1003, ¶ 256.

- h. [8.3] “means for routing the IPv6 data packet to its destination.”

Element [8.3] is identical to element [1.3], except that element [8.3] recites “means for

routing,” whereas element [1.3] recites “routing....” As described above, Ananda renders obvious element [1.3]. Ex. 1003, ¶¶ 257-258. Further, Ananda’s CPI 220E has the same structure as the “means for routing” element as discussed above in Section VIII.C.3 (i.e., telecommunications means 254 of the device 25). Ex. 1003, ¶ 259. The CPI 220E includes IPv4 and IPv6 processors 305, 310 which communicate with the data link layer 308 to route IPv4 and IPv6 packets. Ex. 1005, ¶¶ 90, 94-95, Fig. 3. Further, Ananda discloses that the CPIs may be incorporated in a network interface card (NIC). Ex. 1005, ¶ 118. The CPI 220E includes a telecommunications means. Ex. 1003, ¶ 259.

Thus, Ananda renders obvious element [8.3]. Ex. 1003, ¶ 260.

VI. Secondary Considerations

This Request demonstrates that the Challenged Claims of the ’845 Patent are unpatentable as either being anticipated or having been obvious in view of the prior art references. Ex. 1026, ¶ 218; Ex. 1003 ¶ 381. The Applicant did not identify any evidence of secondary considerations during prosecution. *Id.* Further, the clear teachings in the prior art cannot be overcome by any supposed “secondary considerations.” *Graham v. John Deere Co. of Kansas City*, 383 U.S. 1, 36 (1966); *see Ohio Willow Wood Co. v. Alps South, LLC*, 735 F.3d 1333, 1344 (Fed. Cir. 2013) (“[W]here a claimed invention represents no more than the predictable use of prior art elements according to established functions, as here, evidence of secondary indicia are frequently deemed inadequate to establish non-obviousness.”); *see also* Ex. 1026, ¶ 218; Ex. 1003 ¶ 381.

VII. Disclosure of Concurrent Litigation, Reexamination, and Related Proceedings

Based on information available to Requester, the ’845 Patent is at issue in the following district court matters:

Title	Case No.	Court	Filed	Status
<i>Monarch Networking Solutions LLC v. Juniper Networks, Inc.</i>	1-23-cv-00670	E.D. Va.	2023-05-23	Pending
<i>Monarch Networking Solutions LLC v. Cisco Systems, Inc. et al.</i>	2-20-cv-00015	E.D. Tex.	2020-01-21	Dismissed

VIII. Conclusion

Requester requests the issuance of an Office Action rejecting the Challenged Claims based

Request for *Ex Parte* Reexamination
U.S. Patent 8,451,845

on one or more of Grounds 1-5 detailed above. The Commissioner is hereby authorized to charge Deposit Account 12-0769 under Docket No. US022-0010RE the *Ex Parte* Reexamination fee of \$12,600 under 37 C.F.R. § 1.20(c)(2). Requester believes no other fee is due with this submission; however, the Commissioner is hereby authorized to charge any fee deficiency or credit any overpayment to Deposit Account 12-0769.

Please direct all correspondence in this matter to the undersigned.

Respectfully submitted,

Dated: September 8, 2023

By: /James D. Stein/

James D. Stein (Reg. No. 63,782)
Lee & Hayes, P.C.
75 14th Street NE, Suite 2500
Atlanta, Georgia 30309
james.stein@leehayes.com
P: 404-736-1918

Counsel for Requester Unified Patents, LLC