



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.				
95/000,659	02/13/2012	6629163	159291-0025(163)	6219				
55959	7590	04/03/2012	<table border="1"> <tr><td colspan="2">EXAMINER</td></tr> <tr><td colspan="2">AHMED, SALMAN</td></tr> </table>		EXAMINER		AHMED, SALMAN	
EXAMINER								
AHMED, SALMAN								
Newman Du Wors LLP 1201 Third Avenue, Suite 1600 SEATTLE, WA 98101			<table border="1"> <tr> <th>ART UNIT</th> <th>PAPER NUMBER</th> </tr> <tr> <td>3992</td> <td></td> </tr> </table>		ART UNIT	PAPER NUMBER	3992	
ART UNIT	PAPER NUMBER							
3992								
			<table border="1"> <tr> <th>MAIL DATE</th> <th>DELIVERY MODE</th> </tr> <tr> <td>04/03/2012</td> <td>PAPER</td> </tr> </table>		MAIL DATE	DELIVERY MODE	04/03/2012	PAPER
MAIL DATE	DELIVERY MODE							
04/03/2012	PAPER							

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.



DO NOT USE IN PALM PRINTER

THIRD PARTY REQUESTER'S CORRESPONDENCE ADDRESS
IRELL & MANELLA, LLP
DAVID MCPHIE
840 NEWPORT CENTER DR., STE 400
NEWPORT BEACH, CA 92660

Date: 4-3-12

**Transmittal of Communication to Third Party Requester
Inter Partes Reexamination**

REEXAMINATION CONTROL NO. : 95000659
PATENT NO. : 6629163
TECHNOLOGY CENTER : 3999
ART UNIT : 3992

Enclosed is a copy of the latest communication from the United States Patent and Trademark Office in the above identified Reexamination proceeding. 37 CFR 1.903.

Prior to the filing of a Notice of Appeal, each time the patent owner responds to this communication, the third party requester of the inter partes reexamination may once file written comments within a period of 30 days from the date of service of the patent owner's response. This 30-day time period is statutory (35 U.S.C. 314(b)(2)), and, as such, it cannot be extended. See also 37 CFR 1.947.

If an ex parte reexamination has been merged with the inter partes reexamination, no responsive submission by any ex parte third party requester is permitted.

All correspondence relating to this inter partes reexamination proceeding should be directed to the Central Reexamination Unit at the mail, FAX, or hand-carry addresses given at the end of the communication enclosed with this transmittal.

PTOL-2070(Rev.07-04)

ORDER GRANTING/DENYING REQUEST FOR INTER PARTES REEXAMINATION	Control No.	Patent Under Reexamination
	95/000,659	6629163
	Examiner	Art Unit
	SALMAN AHMED	3992

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address. --

The request for *inter partes* reexamination has been considered. Identification of the claims, the references relied on, and the rationale supporting the determination are attached.

Attachment(s): PTO-892 PTO/SB/08 Other: _____

1. The request for *inter partes* reexamination is GRANTED.

An Office action is attached with this order.

An Office action will follow in due course.

2. The request for *inter partes* reexamination is DENIED.

This decision is not appealable. 35 U.S.C. 312(c). Requester may seek review of a denial by petition to the Director of the USPTO within ONE MONTH from the mailing date hereof. 37 CFR 1.927. EXTENSIONS OF TIME ONLY UNDER 37 CFR 1.183. In due course, a refund under 37 CFR 1.26(c) will be made to requester.

All correspondence relating to this *inter partes* reexamination proceeding should be directed to the **Central Reexamination Unit** at the mail, FAX, or hand-carry addresses given at the end of this Order.

Art Unit: 3992

DECISION GRANTING *INTER PARTES* REEXAMINATION

1. The present request for *inter partes* reexamination establishes a reasonable likelihood that requester will prevail with respect to claims 1, 15 and 35 of United States Patent Number 6,629,163 (Balassanlan, Edward).
2. Extensions of time under 37 CFR 1.136(a) will not be permitted in *inter partes* reexamination proceedings because the provisions of 37 CFR 1.136 apply only to "an applicant" and not to the patent owner in a reexamination proceeding. Additionally, 35 U.S.C. 314(c) requires that *inter partes* reexamination proceedings "will be conducted with special dispatch" (37 CFR 1.937). Patent owner extensions of time in *inter partes* reexamination proceedings are provided for in 37 CFR 1.956. Extensions of time are not available for third party requester comments, because a comment period of 30 days from service of patent owner's response is set by statute. 35 U.S.C. 314(b)(3).

References Cited in the Request

3. The Request identifies the following printed publications as providing teachings relevant to the claims of the '163 patent.

US- 5,298,674 A	03-29-1994	Yun
US- 6,104,500 A	08-15-2000	Alam
US- 6,243,667 B1	06-05-2001	Kerr
US- 5,835,726 A	11-10-1998	Shwed
US- 6,651,099 B1	11-18-2003	Dietz

Art Unit: 3992

PFEIFER et al., Generic Conversion of Communication Media for Supporting Personal Mobility, Multimedia Telecommunication and Applications, COST 237 Workshop, Nov. 25-27, 1996
NORTHERN TELECOM, Digital Switching Systems, ISDN Primary Rate User-Network Interface Specification, NA011, Std 08.01, Aug. 1998
NELSON et al., The Data Compression Book, 2nd Edition; Nov. 6, 1995, M&T Books, New York, NY
COX, Superdistribution: objects as property on the electronic frontier; June 4, 1996, Addison-Wesley Publishing, Reading, MA
FRANZ, Job and Stream Control In Heterogeneous Hardware and Software Architectures, April 22, 1998, Berlin, DE
van der MEER, Dynamic Configuration Management of the Equipment in Distributed Communication Environments, Oct. 6, 1996, Technische Universitat Berlin, DE
Information Sciences Institute, RFC:793, Transmission Control Protocol, DARPA Internet Program Protocol Specification, Sept. 1981, Marina Del Rey, California
MILLS et al., Principles of Information Systems Analysis and Design, copyright 1986, Academic Press, Inc., San Diego, CA
ARBANOWSKI, Generic Description of Telecommunication Services and Dynamic Resource Selection in Intelligent Communication Environments, Oct. 9, 1996, Berlin, DE

Art Unit: 3992

LAWSON, Cisco NetFlow Switching speeds traffic routing, InfoWorld, July 7, 1997, ProQuest Center, pg. 19
BELLARE et al., A Concrete Security Treatment of Symmetric Encryption: Analysis of the DES Modes of Operation, IEEE, Aug. 15, 1997
BELLARE, XOR MACS: New Methods for Message Authentication Using Finite Pseudorandom Functions, CRYPTO '95, LNCS 963, pp. 15-28, 1995, Berlin Heidelberg DE
IBM Raleigh Center, Local Area Network Concepts and Products: Routers and Gateways, 1st Ed., May 1996, Research Triangle Park, NC
NATIONAL INST. OF STDS AND TECH., CheckPoint FireWall-1 White Paper, Version 2.0, Sept. 1995, Germany
BELLISSARD et al., Dynamic Reconfiguration of Agent-Based Applications, Proceedings of ACM European SIGOPS Workshop, Sintra, Sept. 1998
FRASER et al., DTE Firewalls Phase Two Measurement and Evaluation Report, TIS Report #0682, July 22, 1997, Glenwood, MD
DECASPER et al., Router Plugins A Software Architecture of Next Generation Routers, Proceedings of ACM SIGCOMM '98, Sept. 1998, Vancouver B.C.
ATKINSON, Security Architecture for the Internet Protocol, RFC: 1825, Standard Track, Naval Research Lab., Aug. 1995
KARN et al, RFC: 1883: The ESP DES-CBC Transform, Aug. 1995
DEERING & HINDEN, Internet Protocol, Version 6 (IPv6) Specification, RFC: 1883, Standards Track, Dec. 1995
HUITEMA, IPv6: The New Internet Protocol, Oct. 28, 1997, Prentice-Hall, Upper Saddle River, NJ
DECASPER, Crossbow A Toolkit for Integrated Services over Cell Switched IPv6, IEEE, 1997, Zurich, CH/St. Louis, MO
MOSBERGER, Scout: A Path-Based Operating System, Dissertation submitted to Dept of Computer Science, 1997, University of Arizona
KRUPCZAK et al., Implementing Communication Protocols in Java, IEEE Communication Magazine, October 1998
FIUCZYNSKI et al., An Extensible Protocol Architecture for Application-Specific Networking, Department of Computer Science and Engineering, Seattle, WA
MUHUGUSA et al., COMSCRIPT*: An Environment for the Implementation of Protocol Stacks and their Dynamic Reconfiguration, 1994

Issues Raised in the Request

1. The following issues for rejection were proposed in the Request for *inter partes* reexamination:

Issue 1:

Claims 1, 15 and 35 are anticipated by Pfeifer et al. (Generic Conversion of Communication Media for Supporting Personal Mobility), hereinafter Pfeifer96.

Issue 2:

Claims 1, 15 and 35 are obvious over Pfeifer et al. (Generic Conversion of Communication Media for Supporting Personal Mobility), hereinafter Pfeifer96.

Issue 3:

Claims 1, 15 and 35 are obvious over Pfeifer et al. (Generic Conversion of Communication Media for Supporting Personal Mobility), hereinafter Pfeifer96 in view of Specification entitled "ISDN Primary Rate User-Network Interface Specification" from Northern Telecom ("ISDN98") and Book entitled "The Data Compression Book" by Mark Nelson and Jean-Loup Gailly ("Nelson").

Issue 4:

Claims 1, 15 and 35 are obvious over Pfeifer et al. (Generic Conversion of Communication Media for Supporting Personal Mobility), hereinafter Pfeifer96 in view of

Art Unit: 3992

Thesis entitled "Genetic Description of Telecommunication Services and Dynamic Resource Selection in Intelligent Communication Environments" by Stefan Arbanowski ("Arbanowski96").

Issue 5:

Claims 1, 15 and 35 are obvious over Pfeifer et al. (Generic Conversion of Communication Media for Supporting Personal Mobility), hereinafter Pfeifer96 in view of Article entitled "Resource Selection in Heterogeneous Communication Environments using the Teleservice Descriptor" by Tom Pfeifer, Stefan Arbanowski, and Radu Popescu-Zeletin ("Pfeifer97").

Issue 6:

Claims 1, 15 and 35 are obvious over Pfeifer et al. (Generic Conversion of Communication Media for Supporting Personal Mobility), hereinafter Pfeifer96 in view of Book entitled "Superdistribution: Objects as Property on the Electronic Frontier" by Brad Cox ("Cox").

Issue 7:

Claims 1, 15 and 35 are obvious over Pfeifer et al. (Generic Conversion of Communication Media for Supporting Personal Mobility), hereinafter Pfeifer96 in view of Thesis entitled "Dynamic Configuration Management of the Equipment in Distributed Communication Environments" by Sven van der Meer ("Meer96").

Art Unit: 3992

Issue 8:

Claims 1, 15 and 35 are obvious over Pfeifer et al. (Generic Conversion of Communication Media for Supporting Personal Mobility), hereinafter Pfeifer96 in view of Thesis entitled "Dynamic Configuration Management of the Equipment in Distributed Communication Environments" by Sven van der Meer ("Meer96") and Specification entitled RFC 793: "Transmission Control Protocol" by Information Sciences Institute ("RFC 793").

Issue 9:

Claims 1, 15 and 35 are obvious over Pfeifer et al. (Generic Conversion of Communication Media for Supporting Personal Mobility), hereinafter Pfeifer96 in view of Thesis entitled "Job and Stream Control in Heterogeneous Hardware and Software Architectures" by Stefan Franz ("Franz98").

Issue 10:

Claims 1, 15 and 35 are obvious over Pfeifer et al. (Generic Conversion of Communication Media for Supporting Personal Mobility), hereinafter Pfeifer96 in view of Specification entitled "ISDN Primary Rate User-Network Interface Specification" from Northern Telecom ("ISDN98"), Book entitled "The Data Compression Book" by Mark Nelson and Jean-Loup Gailly ("Nelson"), Book entitled "Superdistribution: Objects as Property on the Electronic Frontier" by Brad Cox ("Cox"), Thesis entitled "Dynamic

Art Unit: 3992

Configuration Management of the Equipment in Distributed Communication Environments" by Sven van der Meer ("Meer96"), Specification entitled RFC 793: "Transmission Control Protocol" by Information Sciences Institute ("RFC 793") and Thesis entitled "Job and Stream Control in Heterogeneous Hardware and Software Architectures" by Stefan Franz ("Franz98").

Issue 11:

Claims 1, 15 and 35 are obvious over Pfeifer et al. (Generic Conversion of Communication Media for Supporting Personal Mobility), hereinafter Pfeifer96 in view of Article entitled "Resource Selection in Heterogeneous Communication Environments using the Teleservice Descriptor" by Tom Pfeifer, Stefan Arbanowski, and Radu Popescu-Zeletin ("Pfeifer97") and U.S. Patent No. 6,104,500 entitled "Networked Fax Routing Via Email" by Hassam Alam, Horace Dediu, and Scot Yupaj ("Alam")

Issue 12:

Claims 1, 15 and 35 are obvious over Pfeifer et al. (Generic Conversion of Communication Media for Supporting Personal Mobility), hereinafter Pfeifer96 in view of Article entitled "Resource Selection in Heterogeneous Communication Environments using the Teleservice Descriptor" by Tom Pfeifer, Stefan Arbanowski, and Radu Popescu-Zeletin ("Pfeifer97") and U.S. Patent No. 5,298,674 entitled "Apparatus for Discriminating an Audio Signal as an Ordinary Vocal Sound or Musical Sound" by Sang-Lak Yun ("Yun").

Art Unit: 3992

Issue 13:

Claims 1, 15 and 35 are obvious over Pfeifer et al. (Generic Conversion of Communication Media for Supporting Personal Mobility), hereinafter Pfeifer96 in view of Thesis entitled "Job and Stream Control in Heterogeneous Hardware and Software Architectures" by Stefan Franz ("Franz98"), Thesis entitled "Dynamic Configuration Management of the Equipment in Distributed Communication Environments" by Sven van der Meer ("Meer96"), Thesis entitled "Genetic Description of Telecommunication Services and Dynamic Resource Selection in Intelligent Communication Environments" by Stefan Arbanowski ("Arbanowski96") and Article entitled "Resource Selection in Heterogeneous Communication Environments using the Teleservice Descriptor" by Tom Pfeifer, Stefan Arbanowski, and Radu Popescu-Zeletin ("Pfeifer97").

Issue 14:

Claims 1, 15 and 35 are obvious over Pfeifer et al. (Generic Conversion of Communication Media for Supporting Personal Mobility), hereinafter Pfeifer96 in view of Thesis entitled "Genetic Description of Telecommunication Services and Dynamic Resource Selection in Intelligent Communication Environments" by Stefan Arbanowski ("Arbanowski96"), Article entitled "Resource Selection in Heterogeneous Communication Environments using the Teleservice Descriptor" by Tom Pfeifer, Stefan Arbanowski, and Radu Popescu-Zeletin ("Pfeifer97"), Specification entitled "ISDN Primary Rate User-Network Interface Specification" from Northern Telecom ("ISDN98"),

Art Unit: 3992

Book entitled "The Data Compression Book" by Mark Nelson and Jean-Loup Gailly ("Nelson"), Thesis entitled "Dynamic Configuration Management of the Equipment in Distributed

Communication Environments" by Sven van der Meer ("Meer96"), Specification entitled RFC 793: "Transmission Control Protocol" by Information Sciences Institute ("RFC 793"), Thesis entitled "Job and Stream Control in Heterogeneous Hardware and Software Architectures" by Stefan Franz ("Franz98"), U.S. Patent No. 6,104,500 entitled "Networked Fax Routing Via Email" by Hassam Alam, Horace Dediu, and Scot Yupaj ("Alam") and U.S. Patent No. 5,298,674 entitled "Apparatus for Discriminating an Audio Signal as an Ordinary Vocal Sound or Musical Sound" by Sang-Lak Yun ("Yun").

Issue 15:

Claims 1, 15 and 35 are anticipated over U.S. Pat. No. 6,243,667 entitled "Network Flow Switching and Flow Data Export," by Darren R. Kerr and Barry L. Bruins ("Kerr").

Issue 16:

Claims 1, 15 and 35 are obvious over U.S. Pat. No. 6,243,667 entitled "Network Flow Switching and Flow Data Export," by Darren R. Kerr and Barry L. Bruins ("Kerr").

Issue 17:

Art Unit: 3992

Claims 1, 15 and 35 are obvious over U.S. Pat. No. 6,243,667 entitled "Network Flow Switching and Flow Data Export," by Darren R. Kerr and Barry L. Bruins ("Kerr") in view of Article entitled "Cisco NetFlow Switching speeds traffic routing," InfoWorld Magazine ("NetFlow").

Issue 18:

Claims 1, 15 and 35 are obvious over U.S. Pat. No. 6,243,667 entitled "Network Flow Switching and Flow Data Export," by Darren R. Kerr and Barry L. Bruins ("Kerr") in view of Specification entitled RFC 1825: "Security Architecture for the Internet Protocol" by R. Atkinson ("RFC 1825") and Specification entitled RFC 1829: "The ESP DES-CBC Transform" by P. Kam et al. ("RFC 1829").

Issue 19:

Claims 1, 15 and 35 are obvious over U.S. Pat. No. 6,243,667 entitled "Network Flow Switching and Flow Data Export," by Darren R. Kerr and Barry L. Bruins ("Kerr") in view of Article entitled "A Concrete Security Treatment of Symmetric Encryption" by M. Bellare et al. ("Bellare97") and Article entitled "XOR MACs: New Methods for Message Authentication Using Finite Pseudorandom Functions" by Mihir Bellare, Roch Guerin, and Phillip Rogaway ("Bellare95").

Issue 20:

Claims 1, 15 and 35 are obvious over U.S. Pat. No. 6,243,667 entitled "Network Flow Switching and Flow Data Export," by Darren R. Kerr and Barry L. Bruins ("Kerr") in

Art Unit: 3992

view of Book entitled "Local Area Network Concepts and Products: Routers and Gateways" from IBM ("IBM96").

Issue 21:

Claims 1, 15 and 35 are obvious over U.S. Pat. No. 6,243,667 entitled "Network Flow Switching and Flow Data Export," by Darren R. Kerr and Barry L. Bruins ("Kerr") in view of Book entitled "Local Area Network Concepts and Products: Routers and Gateways" from IBM ("IBM96") and Nelson.

Issue 22:

Claims 1, 15 and 35 are obvious over U.S. Pat. No. 6,243,667 entitled "Network Flow Switching and Flow Data Export," by Darren R. Kerr and Barry L. Bruins ("Kerr") in view of Specification entitled RFC 1825: "Security Architecture for the Internet Protocol" by R. Atkinson ("RFC 1825"), Specification entitled RFC 1829: "The ESP DES-CBC Transform" by P. Kam et al. ("RFC 1829"), Article entitled "A Concrete Security Treatment of Symmetric Encryption" by M. Bellare et al. ("Bellare97") and Article entitled "XOR MACs: New Methods for Message Authentication Using Finite Pseudorandom Functions" by Mihir Bellare, Roch Guerin, and Phillip Rogaway ("Bellare95"), Book entitled "Local Area Network Concepts and Products: Routers and Gateways" from IBM ("IBM96") and Book entitled "The Data Compression Book" by Mark Nelson and Jean-Loup Gailly ("Nelson").

Issue 23:

Claims 1, 15 and 35 are obvious over U.S. Pat. No. 6,243,667 entitled "Network Flow Switching and Flow Data Export," by Darren R. Kerr and Barry L. Bruins ("Kerr") in view of Article entitled "Dynamic Reconfiguration of Agent-Based Applications" by Luc Bellisard, Noel de Palma, and Michel Riveill ("Bellisard").

Issue 24:

Claims 1, 15 and 35 are obvious over U.S. Pat. No. 6,243,667 entitled "Network Flow Switching and Flow Data Export," by Darren R. Kerr and Barry L. Bruins ("Kerr") in view of Publication entitled "DTE Firewalls Phase Two Measurement and Evaluation Report" by Timothy L. Fraser et al. of Trusted Information Systems ("Fraser").

Issue 25:

Claims 1, 15 and 35 are obvious over U.S. Pat. No. 6,243,667 entitled "Network Flow Switching and Flow Data Export," by Darren R. Kerr and Barry L. Bruins ("Kerr") in view of Specification entitled RFC 1825: "Security Architecture for the Internet Protocol" by R. Atkinson ("RFC 1825"), Specification entitled RFC 1829: "The ESP DES-CBC Transform" by P. Kam et al. ("RFC 1829"), Article entitled "A Concrete Security Treatment of Symmetric Encryption" by M. Bellare et al. ("Bellare97") and Article entitled "XOR MACs: New Methods for Message Authentication Using Finite Pseudorandom Functions" by Mihir Bellare, Roch Guerin, and Phillip Rogaway ("Bellare95"), Book entitled "Local Area Network Concepts and Products: Routers and Gateways" from IBM ("IBM96") and Book entitled "The Data Compression Book" by Mark Nelson and Jean-

Art Unit: 3992

Loup Gailly ("Nelson"), Article entitled "Dynamic Reconfiguration of Agent-Based Applications") by Luc Bellisard, Noel de Palma, and Michel Riveill ("Bellisard") and Publication entitled "DTE Firewalls Phase Two Measurement and Evaluation Report" by Timothy L. Fraser et al. of Trusted Information Systems ("Fraser").

Issue 26:

Claims 1, 15 and 35 are obvious over U.S. Pat. No. 6,243,667 entitled "Network Flow Switching and Flow Data Export," by Darren R. Kerr and Barry L. Bruins ("Kerr") in view of Article entitled "Checkpoint Firewall-I White Paper, Version 2.0" ("Checkpoint") and U.S. Pat. No. 5,835,726 entitled "System for securing the flow of and selectively modifying packets in a computer network," by Shwed et al. ("Shwed").

Issue 27:

Claims 1, 15 and 35 are obvious over U.S. Pat. No. 6,243,667 entitled "Network Flow Switching and Flow Data Export," by Darren R. Kerr and Barry L. Bruins ("Kerr") in view of U.S. Pat. No. 6,651,099 entitled "Method and Apparatus for Monitoring Traffic in a Network" by Russell S. Dietz et al. ("Dietz").

Issue 28:

Claims 1, 15 and 35 are obvious over U.S. Pat. No. 6,243,667 entitled "Network Flow Switching and Flow Data Export," by Darren R. Kerr and Barry L. Bruins ("Kerr") in view of Pfeifer et al. (Generic Conversion of Communication Media for Supporting Personal Mobility), hereinafter Pfeifer96.

Issue 29:

Claims 1, 15 and 35 are anticipated by Article entitled "Router Plugins: A Software Architecture for Next Generation Routers" by Dan Decasper et al. ("Decasper98").

Issue 30:

Claims 1, 15 and 35 are obvious over Article entitled "Router Plugins: A Software Architecture for Next Generation Routers" by Dan Decasper et al. ("Decasper98").

Issue 31:

Claims 1, 15 and 35 are obvious over Article entitled "Router Plugins: A Software Architecture for Next Generation Routers" by Dan Decasper et al. ("Decasper98") in view of Specification entitled RFC 1825: "Security Architecture for the Internet Protocol" by R. Atkinson ("RFC 1825"), Specification entitled RFC 1829: "The ESP DES-CBC Transform" by P. Kam et al. ("RFC 1829").

Issue 32:

Claims 1, 15 and 35 are obvious over Article entitled "Router Plugins: A Software Architecture for Next Generation Routers" by Dan Decasper et al. ("Decasper98") in view of Specification entitled RFC 1883: "Internet Protocol, Version 6 (IPv6)

Art Unit: 3992

Specification" by S. Deering and R. Hinden ("RFC 1883") and Book entitled "IPv6: The New Internet Protocol" by Christian Huitema ("Huitema").

Issue 33:

Claims 1, 15 and 35 are obvious over Article entitled "Router Plugins: A Software Architecture for Next Generation Routers" by Dan Decasper et al. ("Decasper98") in view of Article entitled "Crossbow: A Toolkit for Integrated Services over Cell Switched IPv6" by Dan Decasper et al. ("Decasper97").

Issue 34:

Claims 1, 15 and 35 are obvious over Article entitled "Router Plugins: A Software Architecture for Next Generation Routers" by Dan Decasper et al. ("Decasper98") in view of Article entitled "Crossbow: A Toolkit for Integrated Services over Cell Switched IPv6" by Dan Decasper et al. ("Decasper97"), Article entitled "A Concrete Security Treatment of Symmetric Encryption" by M. Bellare et al. ("Bellare97") and Article entitled "XOR MACs: New Methods for Message Authentication Using Finite Pseudorandom Functions" by Mihir Bellare, Roch Guerin, and Phillip Rogaway ("Bellare95").

Issue 35:

Claims 1, 15 and 35 are obvious over Article entitled "Router Plugins: A Software Architecture for Next Generation Routers" by Dan Decasper et al. ("Decasper98") in

Art Unit: 3992

view of Book entitled "Local Area Network Concepts and Products: Routers and Gateways" from IBM ("IBM96").

Issue 36:

Claims 1, 15 and 35 are obvious over Article entitled "Router Plugins: A Software Architecture for Next Generation Routers" by Dan Decasper et al. ("Decasper98") in view of Book entitled "Local Area Network Concepts and Products: Routers and Gateways" from IBM ("IBM96") and Book entitled "The Data Compression Book" by Mark Nelson and Jean-Loup Gailly ("Nelson").

Issue 37:

Claims 1, 15 and 35 are obvious over Article entitled "Router Plugins: A Software Architecture for Next Generation Routers" by Dan Decasper et al. ("Decasper98") in view of Specification entitled RFC 1825: "Security Architecture for the Internet Protocol" by R. Atkinson ("RFC 1825"), Specification entitled RFC 1829, "Crossbow: A Toolkit for Integrated Services over Cell Switched IPv6" by Dan Decasper et al. ("Decasper97"), Article entitled "A Concrete Security Treatment of Symmetric Encryption" by M. Bellare et al. ("Bellare97") and Article entitled "XOR MACs: New Methods for Message Authentication Using Finite Pseudorandom Functions" by Mihir Bellare, Roch Guerin, and Phillip Rogaway ("Bellare95"), Book entitled "Local Area Network Concepts and Products: Routers and Gateways" from IBM ("IBM96") and Book entitled "The Data Compression Book" by Mark Nelson and Jean-Loup Gailly ("Nelson").

Art Unit: 3992

Issue 38:

Claims 1, 15 and 35 are obvious over Article entitled "Router Plugins: A Software Architecture for Next Generation Routers" by Dan Decasper et al. ("Decasper98") in view of Article entitled "Dynamic Reconfiguration of Agent-Based Applications" by Luc Bellisard, Noel de Palma, and Michel Riveill ("Bellisard").

Issue 39:

Claims 1, 15 and 35 are obvious over Article entitled "Router Plugins: A Software Architecture for Next Generation Routers" by Dan Decasper et al. ("Decasper98") in view of Publication entitled "DTE Firewalls Phase Two Measurement and Evaluation Report" by Timothy L. Fraser et al. of Trusted Information Systems ("Fraser").

Issue 40:

Claims 1, 15 and 35 are obvious over Article entitled "Router Plugins: A Software Architecture for Next Generation Routers" by Dan Decasper et al. ("Decasper98") in view of Specification entitled RFC 1825: "Security Architecture for the Internet Protocol" by R. Atkinson ("RFC 1825"), Specification entitled RFC 1829: "The ESP DES-CBC Transform" by P. Kam et al. ("RFC 1829"), Specification entitled RFC 1883: "Internet Protocol, Version 6 (IPv6) Specification" by S. Deering and R. Hinden ("RFC 1883"), Book entitled "IPv6: The New Internet Protocol" by Christian Huitema ("Huitema"), Article entitled "Crossbow: A Toolkit for Integrated Services over Cell Switched IPv6" by Dan Decasper et al. ("Decasper97"), Article entitled "A Concrete Security Treatment of

Art Unit: 3992

Symmetric Encryption" by M. Bellare et al. ("Bellare97"), Article entitled "XOR MACs: New Methods for Message Authentication Using Finite Pseudorandom Functions" by Mihir Bellare, Roch Guerin, and Phillip Rogaway ("Bellare95"), Book entitled "Local Area Network Concepts and Products: Routers and Gateways" from IBM ("IBM96"), Book entitled "The Data Compression Book" by Mark Nelson and Jean-Loup Gailly ("Nelson"), Article entitled "Dynamic Reconfiguration of Agent-Based Applications" by Luc Bellisard, Noel de Palma, and Michel Riveill ("Bellisard") and Publication entitled "DTE Firewalls Phase Two Measurement and Evaluation Report" by Timothy L. Fraser et al. of Trusted Information Systems ("Fraser").

Issue 41:

Claims 1, 15 and 35 are anticipated by Article entitled "Implementing Communication Protocols in Java" by Bobby Krupczak et. al ("HotLava").

Issue 42:

Claims 1, 15 and 35 are anticipated by Dissertation entitled "Scout: A Path-Based Operating System" by David Mosberger ("Mosberger").

Issue 43:

Claims 1, 15 and 35 are obvious over Dissertation entitled "Scout: A Path-Based Operating System" by David Mosberger ("Mosberger").

Art Unit: 3992

Issue 44:

Claims 1, 15 and 35 are obvious over Dissertation entitled "Scout: A Path-Based Operating System" by David Mosberger ("Mosberger") and Article entitled "Implementing Communication Protocols in Java" by Bobby Krupczak et. al ("HotLava").

Issue 45:

Claims 1, 15 and 35 are obvious over Dissertation entitled "Scout: A Path-Based Operating System" by David Mosberger ("Mosberger") and Article entitled "An Extensible Protocol Architecture for Application-Specific Networking" by Marc Fiuczynski et. al ("Plexus").

Issue 46:

Claims 1, 15 and 35 are obvious over Dissertation entitled "Scout: A Path-Based Operating System" by David Mosberger ("Mosberger") and Article entitled "ComScript: An Environment for the Implementation of Protocol Stacks and their Dynamic Reconfiguration" by Murhimanya Muhugusa et. al (ComScript).

Reasonable Likelihood that Requester will prevail Statement

As noted on pages 24-271 of the Request, the Requester submits that **Pfeifer96**, **Kerr**, **Decasper98**, **Mosberger** and **HotLava** in combination with other prior art raise RLP based on proposed teachings.

ISSUE 1

Pfeifer96 teaches the Intelligent Personal Communication Support System is introduced as an application for multiple media conversion tools, embedded in a context of personal mobility, service personalization and service interoperability support.



Fig. 1. Media converter system

After discussing models for conversion in theory, the current conversion technology is evaluated. The necessity of an integrated framework of flexible converters and a generic converter model are derived and automatic management of conversion quality is discussed (Abstract).

Pfeifer96 further teaches

Art Unit: 3992

For converting one format of the same medium into another, tools exist for many platforms for text, bitmap images and audio. They are mostly in the public domain, and perform well as software solutions [32]. Converting video formats requires the appropriate encoding/decoding hardware and software for the compression methods involved [28, 29, 30, 31].

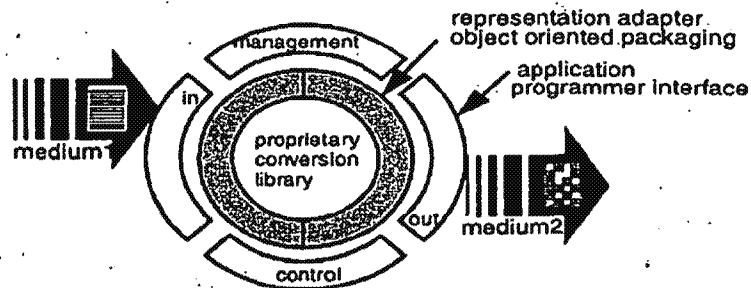


Fig. 7. Generic converter model

Finally, Pfeifer96 teaches:

In order to illustrate the benefits of the PCS concept, Figure 10 depicts a simplified intelligent call processing model to be performed by an advanced Personal Communication platform. It is characterized by a *four-stage mapping process* that translates a logical user name used as the called party address (i.e. a personal ID) into an appropriate network address (i.e. a terminal ID). This temporary physical address is passed back to the requesting communication service. The mapping process looks as follows:

- *1st*, the evaluation of a user's "Personal Call Logic" provides the *control of his reachability*. The result may be a forwarding to another user, a call rejection, a call redirection to an asynchronous service, e.g. an answering machine, or an acceptance.
- *2nd*, the exact recipient of the communication invitation has been settled and no further call management will be performed. A *mapping of the user to his location* is made based on user registration data.

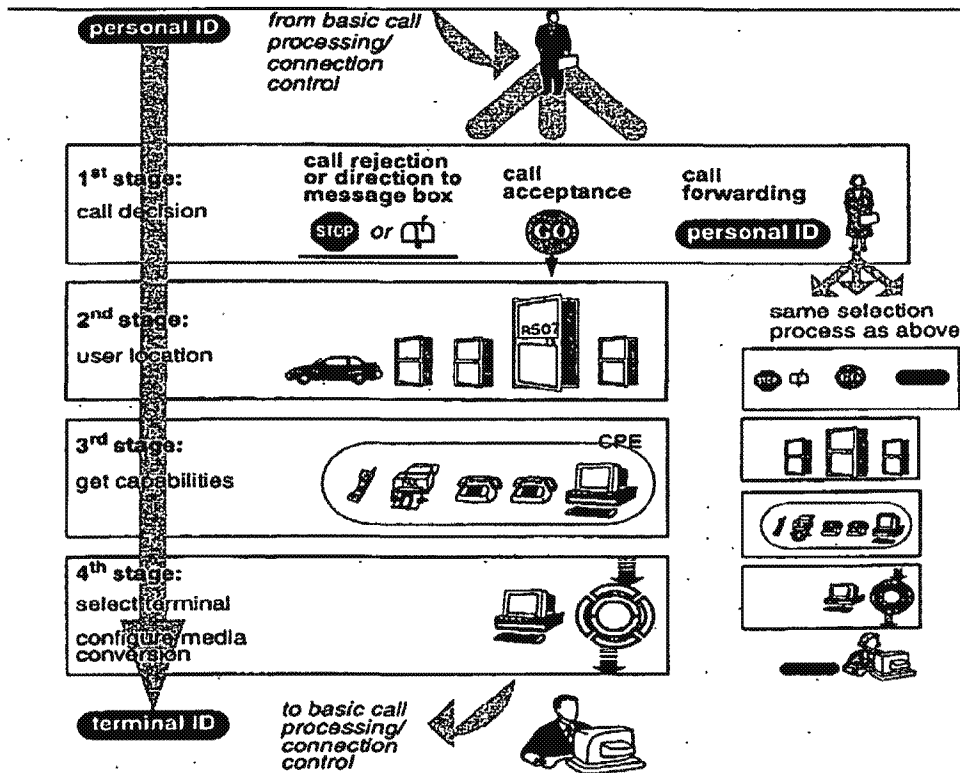


Fig. 10. PCS-based Intelligent Call Processing

- 3rd, it maps a location to a virtual communication endpoint corresponding to a terminal group representing the set of all access devices in the user's current vicinity. An object-oriented modelling of virtual communication endpoints encompasses the knowledge on terminal capabilities; supported services, and selection mechanisms.
- 4th, an appropriate terminal ID from the group of devices is selected and parameterized by a service type, used communication media, and optionally by user preferences. Within this stage, two cases can be distinguished:
 - a) In case there exist at least one device of the virtual communication end-point supporting the desired medium of the call, the most appropriate device is selected.
 - b) In case no device for the desired medium can be found, further rules of the Personal Call Logic determine whether a conversion into another medium is allowed/restricted. Then, the necessary converters are configured and a now appropriate device is selected.

However, claim 1 states:

...for the first packet of the message, identifying a sequence of components for processing the packets of the message such that the output format of the components of the sequence match the input format of the next call component in the sequence; and

storing an indication of each of the identified components so that the sequence does not need to be re-identified for subsequent packets of the message...

Similarly, claim 15 states:

...identifying a sequence of components for processing each message based on the **first packet** of the message so that subsequent packets of the message can be processed without re-identifying the components, ...

Similarly, claim 35 states:

...identifying a message-specific sequence of components for processing the packets of each message upon receiving the **first packet** of the message wherein subsequent packets of the message can use the message-specific sequence identified when the **first packet** was received...

In regards to these limitations relating to “**first packet**”, Requester submits in page 32:

whereabouts for obtaining access to their service.” *Id.* at 118. Because users are mobile and move in and out of range of various “terminal equipment” with varying “capabilities,” it is not possible to determine the specific media conversions that will be needed to achieve a connection to the user until the first packet of the message to that user has been received by the iPCSS. *See id.* at 119. This is why Pfeifer96 teaches a multi-stage call connection procedure, wherein the

Requester submits in page 34:

Art Unit: 3992

Processing in the iPCSS proceeds in this order for an obvious reason: it is not possible to put together a suitable "chain of converters" between devices until the source and destination devices are *known*, and the devices will not be known before the first packet of the call has arrived. *See id.* at 114, 118-19. The iPCSS will not know the *source* device and its medium until the device initiates the call. *Id.* at 119-20. For example, is it a voice call, a fax, or an email? *Id.* at 119-20. And likewise the iPCSS will not know "the set" of possible *destination* devices in the called "user's current vicinity" until the call is initiated, because the user's vicinity (and hence the devices in that vicinity) can change from moment to moment. *Id.* For example, is there a fax machine or computer nearby, or merely a telephone? *See id.* As explained by Pfeifer96: "The iPCSS architecture . . . aim[s] . . . to increase the nomadic user's reachability by introducing . . . the dynamic selection of terminals." *Id.* at 122 (emphasis added).

Requester submits in page 38:

Id. at 124, 116 (emphasis added). Of course, as demonstrated above, this elaborate analysis cannot even begin until the first packet of the message has been received by the iPCSS. Among

Requester submits in page 39:

After receiving the first packet of the message, Pfeifer96 concatenates individual converters to form many possible chains that might be used to connect the message's two endpoints. *See id.* at 114, 124. The possible chains are compared to determine which would

Examiner submits that Pfeifer96 does not appear to teach "first packet" initiating the "identifying" step of sequence of components. Pfeifer96 talks about initiating calls or sessions; however, initiating calls or sessions does not necessarily equate to "first packet" of a message triggering the "identifying" steps. Although Examiner agrees with Requester's comment that connection cannot be established until the "first packet" is received, however, nowhere in Pfeifer96 it is stated that the first

Art Unit: 3992

packet initiates "identifying" step of sequence of components and all other subsequent "retrieving" step of state information relating to performing processing of previous packet; and the "storing" step of the state information, , as required by the claim limitations.

Therefore, Examiner submits that Pfeifer96 does not appear to teach ...for the ***first packet*** of the message, ***identifying*** a sequence of components for processing the packets of the message... as in claim 1, ...***identifying*** a sequence of components for processing each message based on the ***first packet*** of the message so that subsequent packets of the message can be processed without re-identifying the components, ... as in claim 15 and ...***identifying*** a message-specific sequence of components for processing the packets of each message upon receiving the ***first packet*** of the message wherein subsequent packets of the message can use the message-specific sequence identified when the ***first packet*** was received... as in claim 35.

Hence, for the reasons cited above, it is found that the requester has not shown a reasonable likelihood of prevail with respect to claims 1, 15 and 35.

ISSUE 2

Claim 1 states:

...for the ***first packet*** of the message, ***identifying*** a sequence of components for processing the packets of the message such that the output format of the components of the sequence match the input format of the next component in the sequence; and

Art Unit: 3992

storing an indication of each of the identified components so that the sequence does not need to be re-identified for subsequent packets of the message...

Similarly, claim 15 states:

...identifying a sequence of components for processing each message based on the first packet of the message so that subsequent packets of the message can be processed without re-identifying the components, ...

Similarly, claim 35 states:

...identifying a message-specific sequence of components for processing the packets of each message upon receiving the first packet of the message wherein subsequent packets of the message can use the message-specific sequence identified when the first packet was received...

In regards to these limitations relating to “first packet”, Requester submits in page 53:

received.” As explained above, after receiving the first packet of the message, Pfeifer96 concatenates individual converters to form *many* possible chains that might be used to connect the message’s two endpoints. See Section V.A.1 (Pfeifer 102) at Claim 1(iii) above.

Requester submits in page 63:

claim constructions, Pfeifer96 renders obvious this element. See Claim 1(iii) (showing “for the first packet of the message, dynamically identifying a non-predefined sequence of components for processing the packets of the message”) and Claim 1(iv) (showing “storing an indication of

Requester submits in page 68:

Art Unit: 3992

element. *See* Claim 1(iii) (showing “for the first packet of the message, dynamically identifying a non-predefined sequence of components for processing the packets of the message”) above.

As Examiner has shown above:

Requester submits in page 32:

whereabouts for obtaining access to their service.” *Id.* at 118. Because users are mobile and move in and out of range of various “terminal equipment” with varying “capabilities,” it is not possible to determine the specific media conversions that will be needed to achieve a connection to the user until the first packet of the message to that user has been received by the iPCSS. *See id.* at 119. This is why Pfeifer96 teaches a multi-stage call connection procedure, wherein the

Requester submits in page 34:

Processing in the iPCSS proceeds in this order for an obvious reason: it is not possible to put together a suitable “chain of converters” between devices until the source and destination devices are *known*, and the devices will not be known before the first packet of the call has arrived. *See id.* at 114, 118-19. The iPCSS will not know the *source* device and its medium until the device initiates the call. *Id.* at 119-20. For example, is it a voice call, a fax, or an email? *Id.* at 119-20. And likewise the iPCSS will not know “the set” of possible *destination* devices in the called “user’s current vicinity” until the call is initiated, because the user’s vicinity (and hence the devices in that vicinity) can change from moment to moment. *Id.* For example, is there a fax machine or computer nearby, or merely a telephone? *See id.* As explained by Pfeifer96: “The iPCSS architecture . . . aim[s] . . . to increase the nomadic user’s reachability by introducing . . . the **dynamic** selection of terminals.” *Id.* at 122 (emphasis added).

Requester submits in page 38:

Id. at 124, 116 (emphasis added). Of course, as demonstrated above, this elaborate analysis cannot even begin until the first packet of the message has been received by the iPCSS. Among

Requester submits in page 39:

After receiving the first packet of the message, Pfeifer96 concatenates individual converters to form many possible chains that might be used to connect the message's two endpoints. See id. at 114, 124. The possible chains are compared to determine which would

Examiner submits that Pfeifer96 does not appear to disclose **"first packet" initiating the "identifying" step of sequence of components.** Pfeifer96 talks about initiating calls or sessions; however, initiating calls or sessions does not necessarily equates to "first packet" of a message triggering the "identifying" steps. Although Examiner agrees with Requester's comment that connection cannot be established until the "first packet" is received, however, nowhere in Pfeifer96 it is stated that the first packet initiates "identifying" step of sequence of components and all other subsequent "retrieving" step of state information relating to performing processing of previous packet; and the "storing" step of the state information, as required by the claim limitations.

Therefore, Examiner submits that Pfeifer96 does not appear to disclose ...for the **first packet** of the message, **identifying** a sequence of components for processing the packets of the message... as in claim 1, ...**identifying** a sequence of components for processing each message based on the **first packet** of the message so that subsequent packets of the message can be processed without re-identifying the components, ... as in claim 15 and ...**identifying** a message-specific sequence of components for processing the packets of each message upon receiving the **first packet** of the message wherein subsequent packets of the message can use the

Art Unit: 3992

message-specific sequence identified when the *first packet* was received... as in claim 35.

Hence, for the reasons cited above, it is found that the requester has not shown a reasonable likelihood of prevail with respect to claims 1, 15 and 35.

ISSUE 3

Nelson discloses (as submitted by the Request, page 58) "Adaptive coding... lead[s] to vastly improved compression ratios," and that "compression research in the last 10 years has concentrated on adaptive models." Ex. 5 at 8, 18. Adaptive algorithms include such well-known algorithms as "Adaptive Huffman Coding" (chapter 4; id. at 75), "Adaptive [Statistical] Modeling" (chapter 6; id. at 155), "[Adaptive] Dictionary-Based Compression" (chapter 7: id. at 203), and "Sliding Window Compression" (chapter 8; id. at 215); and the prominent "LZ" family of compression algorithms (chapter 8 and 9, id. at 221,255). All of these adaptive techniques are lossless. See id. at 9 ("All of the compression techniques discussed through chapter 9 are 'lossless'").

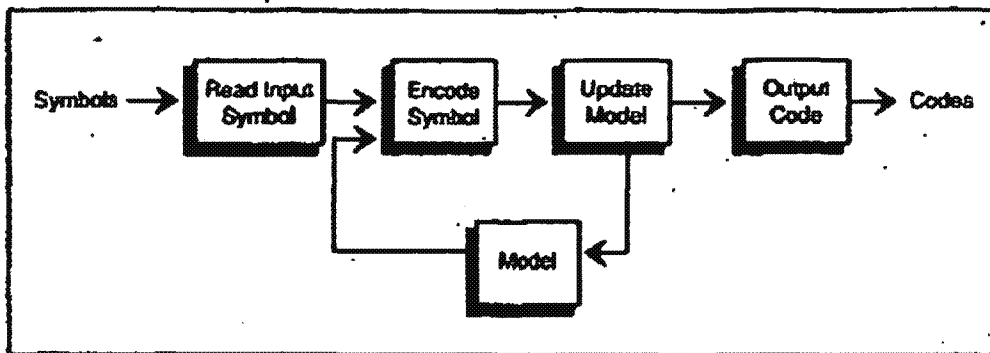


FIGURE 2.2 GENERAL ADAPTIVE COMPRESSION.

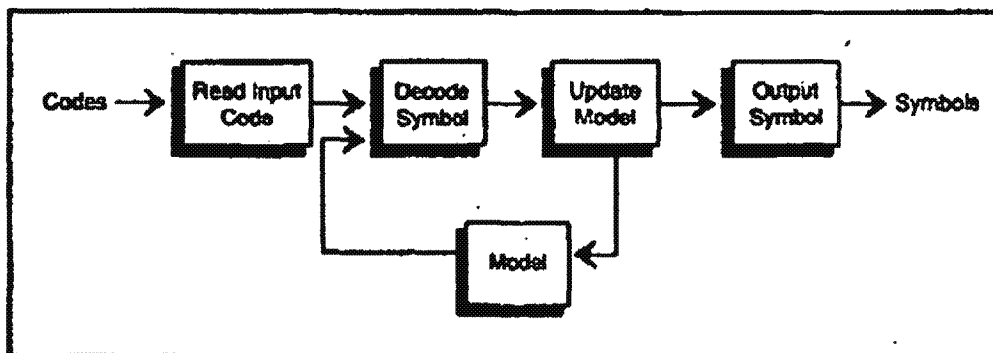


FIGURE 2.3 GENERAL ADAPTIVE DECOMPRESSION.

Nelson further explains the stateful manner in which adaptive coding operates: "When using an adaptive model, data does not have to be scanned once before coding in order to generate statistics [used to perform compression/decompression]. Instead, the statistics are continually modified as new characters are read in and coded. The general flow of a program using an adaptive model looks something like that shown in Figures 2.2 and 2.3." Id. At 18.

However, Claim 1 states:

...for the **first packet** of the message, **identifying** a sequence of components for processing the packets of the message such that the output format of the components of the sequence match the input format of the next component in the sequence; and **storing** an indication of each of the identified components so that the sequence does not need to be re-identified for subsequent packets of the message...

Similarly, claim 15 states:

...**identifying** a sequence of components for processing each message based on the **first packet** of the message so that subsequent packets of the message can be processed without re-identifying the components, ...

Similarly, claim 35 states:

...**identifying** a message-specific sequence of components for processing the packets of each message upon receiving the **first packet** of the message wherein subsequent packets of the message can use the message-specific sequence identified when the **first packet** was received...

In regards to these limitations relating to “**first packet**”, Requester submits in page 53:

If certain aspects recited in claims 1, 15, and 35 of the ‘163 patent are not deemed to be disclosed, inherent, or obvious over Pfeifer96 alone, then the inclusion of those aspects certainly would be obvious over Pfeifer96 in view of ISDN98 and Nelson, under 35 U.S.C. § 103.

ISDN98 and Nelson were cited above under MPEP § 2205 as confirming that certain information regarding ISDN (ISDN98) and compression (Nelson) would have been part of the standard background knowledge of those of ordinary skill in the art. See Section V.A.2 above.

Art Unit: 3992

Nelson fails to overcome the deficiencies of Pfeifer96, ...for the first packet of the message, identifying a sequence of components for processing the packets of the message... as in claim 1, ...identifying a sequence of components for processing each message based on the first packet of the message so that subsequent packets of the message can be processed without re-identifying the components, ... as in claim 15 and ...identifying a message-specific sequence of components for processing the packets of each message upon receiving the first packet of the message wherein subsequent packets of the message can use the message-specific sequence identified when the first packet was received... as in claim 35.

Hence, for the reasons cited above, it is found that the requester has not shown a reasonable likelihood of prevail with respect to claims 1, 15 and 35.

ISSUE 4

Arbanowksi96 discloses the PCSS functional model defining the main functional processes that are involved in the provision of personal communication capabilities. A major task is the transformation of PIDs to appropriate Customer Premises Equipment (CPE) in according to the user preferences and registration information. The PCSS performs a multi stage functional mapping from the given PID of the called party (the person the caller wants to communicate with) to a physical terminal ID at the location of the addressed person. This process involves the handling of parameters of the media formal and service type used by the calling party (the person which has initiated the call) and the communication of the called party:

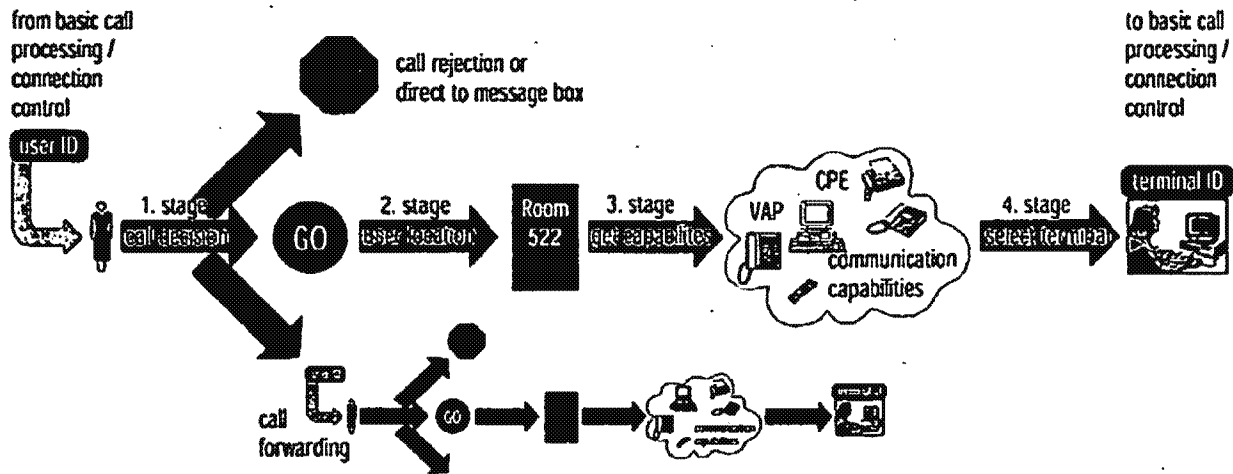


Figure 2-1 PCSS – Call Handling

In particular four stages of the mapping from a PID to a terminal ID are defined. Figure 2-1 shows the whole mapping scenario.

- ☛ The evaluation of a user’s personal call logic constitutes the 1st stage, providing the management of reachability (often a person to person mapping: call forwarding, call accepting, call blocking, announcement, voice box).
- ☛ The 2nd stage usually performs a person to location mapping based on user registration data. An electronic location system (infrared based Active Badges [Hopper94]) is used for automatic registration at locations. For the manual registration a user application was developed [Vetter95]. To use it, a user must himself identify and authenticate to the system.
- ☛ The 3rd stage performs a mapping from the location to a virtual communication endpoint corresponding to a group of terminals. Virtual Access Points (VAP) represent a set of terminals in the users current vicinity. A virtual access point encompasses knowledge on terminal capabilities, supported services, and on selection mechanisms.
- ☛ The 4th stage selects an appropriate terminal ID from the group of devices. The functionality will be performed by the VAP selected in the previous stage. The selection in this stage is parameterized by a service type (of the incoming call), by the used media, and optionally by user preferences.

Claim 1 states:

...for the *first packet* of the message, *identifying* a sequence of components for processing the packets of the message such that the output format of the components of the sequence match the input format of the next component in the sequence; and *storing* an indication of each of the identified components so that the sequence does not need to be re-identified for subsequent packets of the message...

Similarly, claim 15 states:

...**identifying** a sequence of components for processing each message based on the **first packet** of the message so that subsequent packets of the message can be processed without re-identifying the components, ...

Similarly, claim 35 states:

...**identifying** a message-specific sequence of components for processing the packets of each message upon receiving the **first packet** of the message wherein subsequent packets of the message can use the message-specific sequence identified when the **first packet** was received...

In regards to these limitations relating to “**first packet**”, Requester submits in page 76:

obvious these elements. See Claim 1(iii) (showing “for the first packet of the message, dynamically identifying a non-predefined sequence of components for processing the packets of the message”) and Claim 1(iv) (showing “storing an indication of each of the identified

Requester submits in page 77:

renders obvious this element. See Claim 1(iii) (showing “for the first packet of the message, dynamically identifying a non-predefined sequence of components for processing the packets of the message”) above.

However, Arbanowksi96 fails to overcome the deficiencies of Pfeifer96, ...for the **first packet** of the message, **identifying** a sequence of components for processing the packets of the message... as in claim 1, ...**identifying** a sequence of components for processing each message based on the **first packet** of the message so that subsequent packets of the message can be processed without re-identifying the components, ... as in claim 15 and ...**identifying** a message-specific sequence of

Art Unit: 3992

components for processing the packets of each message upon receiving the *first packet* of the message wherein subsequent packets of the message can use the message-specific sequence identified when the *first packet* was received... as in claim 35.

Hence, for the reasons cited above, it is found that the requester has not shown a reasonable likelihood of prevail with respect to claims 1, 15 and 35.

ISSUE 5

Pfeifer97 teaches (Abstract) automated processes in distributed communication environments require tools for unifying heterogeneous multimedia services. The Teleservice Descriptor is introduced for generic handling and integration of traditional and innovative forms of communication. The Intelligent Resource Selector applies this descriptor for dynamic selection of communication end points and combination of necessary converters for service interworking. The Intelligent Personal Communication Support System provides the test-bed for the implementation of the developed algorithms, applicable in CPE, TINA and IN solutions.

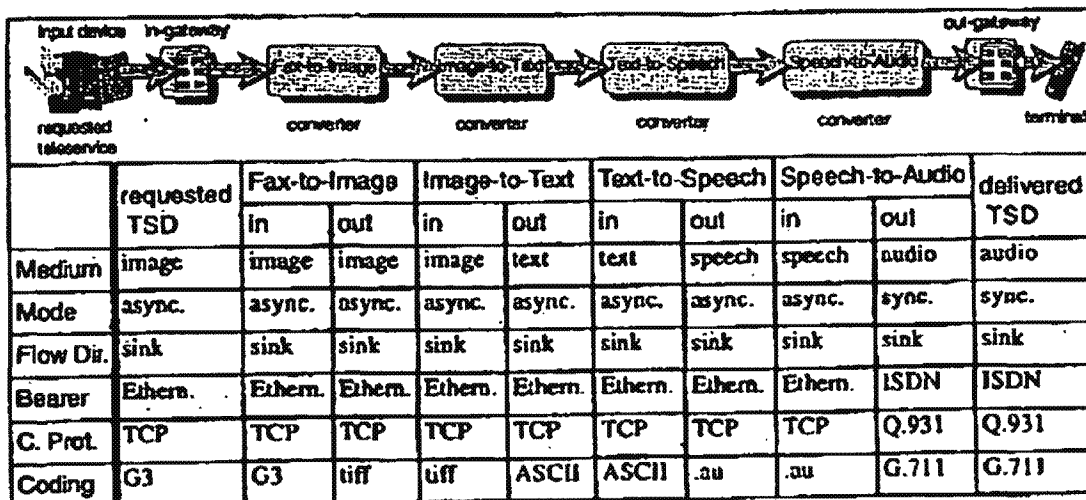


Fig. 5. Dynamic Resource Selection - Converter Chain with TSD mappings

Claim 1 states:

...for the **first packet** of the message, **identifying** a sequence of components for processing the packets of the message such that the output format of the components of the sequence match the input format of the next component in the sequence; and **storing** an indication of each of the identified components so that the sequence does not need to be re-identified for subsequent packets of the message...

Similarly, claim 15 states:

...**identifying** a sequence of components for processing each message based on the **first packet** of the message so that subsequent packets of the message can be processed without re-identifying the components, ...

Similarly, claim 35 states:

...**identifying** a message-specific sequence of components for processing the packets of each message upon receiving the **first packet** of the message wherein

Art Unit: 3992

subsequent packets of the message can use the message-specific sequence identified when the **first packet** was received...

In regards to these limitations relating to “**first packet**”, Requester submits in page 83:

Under Implicit’s apparent claim constructions, Pfeifer96 in view of Pfeifer97 renders obvious this element. See Claim 1(iii) (showing “for the first packet of the message, dynamically identifying a non-predefined sequence of components for processing the packets of the message”) and Claim 1(iv) (showing “storing an indication of each of the identified components

Requester submits in page 84:

obvious this element. See Claim 1(iii) (showing “for the first packet of the message, dynamically identifying a non-predefined sequence of components for processing the packets of the message”) above.

However, Pfeifer97 fails to overcome the deficiencies of Pfeifer96, ...for the **first packet** of the message, identifying a sequence of components for processing the packets of the message... as in claim 1, ...identifying a sequence of components for processing each message based on the **first packet** of the message so that subsequent packets of the message can be processed without re-identifying the components, ... as in claim 15 and ...identifying a message-specific sequence of components for processing the packets of each message upon receiving the **first packet** of the message wherein subsequent packets of the message can use the message-specific sequence identified when the **first packet** was received... as in claim 35.

Hence, for the reasons cited above, it is found that the requester has not shown a reasonable likelihood of prevail with respect to claims 1, 15 and 35.

ISSUE 6

Cox discloses (Request, pages 88-89) "an invocation-based metering" approach which Cox styles "Superdistribution." See, e.g., id. at 155, 169 ("invocation-based revenue collection") (emphasis in original). The goal of this "Superdistribution" approach is to "provide a meter that supports revenue collection for components of any granularity, Id. at 156. Assessing royalties based on actual usage of a component would solve a number of problems, including the problem of Vendor E:

Instead of paying a large fee up-front, all customers, large and small, get the component for free. Later, when they begin to sell their own products based on this component, they pay a negotiated) fee for using their subvendor's product. The subvendor now receives a continuing revenue stream that is directly proportional to the utility his component provides to his customers. Id. at 154.

In any event, upon reading Cox, one of ordinary skill in the art could not fail to see its relevance to the small, reusable converter components of Pfeifer96. Though the Cox approach could obviously be applied to metering the usage of any components in a large software system such as iPCSS, the converter components of Pfeifer96 in particular would stand out as especially likely candidates for this treatment, because Pfeifer96 expressly teaches they may be "proprietary" external components obtained

Art Unit: 3992

"from different manufacturers," rather than components developed purely internally. See Ex. A02 (Pfeifer96) at 108, 113-14.

However, Cox fails to overcome the deficiencies of Pfeifer96, ...for the **first packet** of the message, **identifying** a sequence of components for processing the packets of the message... as in claim 1, ...**identifying** a sequence of components for processing each message based on the **first packet** of the message so that subsequent packets of the message can be processed without re-identifying the components, ... as in claim 15 and ...**identifying** a message-specific sequence of components for processing the packets of each message upon receiving the **first packet** of the message wherein subsequent packets of the message can use the message-specific sequence identified when the **first packet** was received... as in claim 35.

Hence, for the reasons cited above, it is found that the requester has **not** shown a reasonable likelihood of prevail with respect to claims 1, 15 and 35.

ISSUE 7

Meer96 teaches the PCSS functional model defines the main functional processes that are involved in the dynamic terminal selection. Starting with a request from the teleservice that provides a personal ID of the called party, accompanied by parameters of the media formats and service type used by the calling party, a multi-stage functional mapping to a physical terminal ID will be performed. The result, i.e. the

selected terminal ID, will be replied to the teleservice which uses this parameter as input to the subsequent basic call processing. [Eckardt96a]

The VAP, major part in the 4th stage, selects an appropriate terminal and (if required and if possible) configures specialized resource functionality. The configuration and parametrization processes are based on a generic service description, including parameters such as (used) media, bearer, service type and service data format. Media/format conversions are not in the scope of the PCSS-VAP object and the generic service description is in a rather preliminary state.

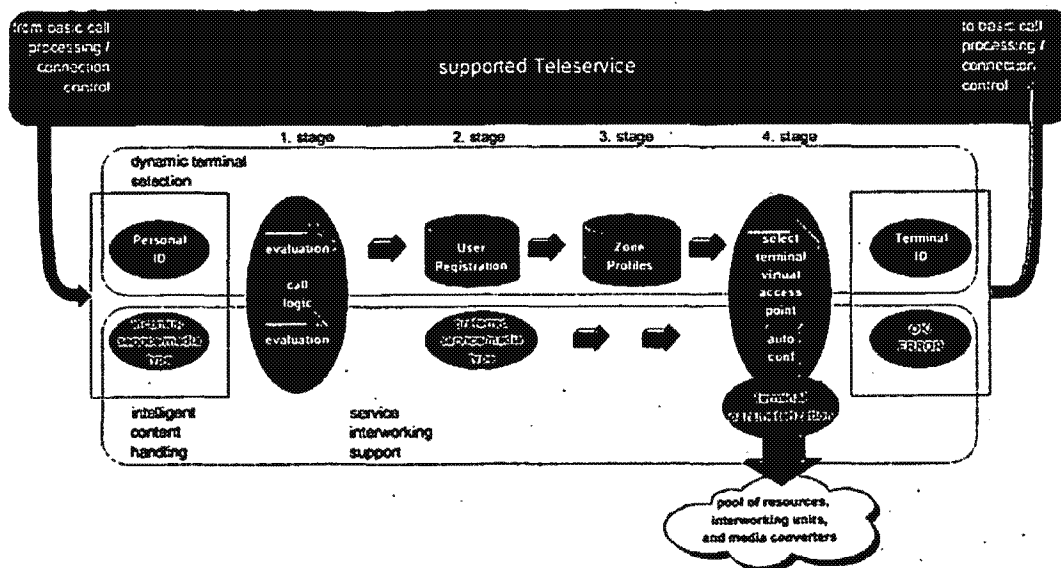


Figure 2-2: PCSS - Functional Model [Eckardt96a]

Requester submits in page 93, one particularly pertinent manner in which Meer96 presents significant additional information is regarding the "state information" element of claim 1 ("for each of a plurality of packets of the message in sequence, for each of a plurality of components in the identified non-predefined sequence, retrieving state information relating to performing the processing of the component with the

Art Unit: 3992

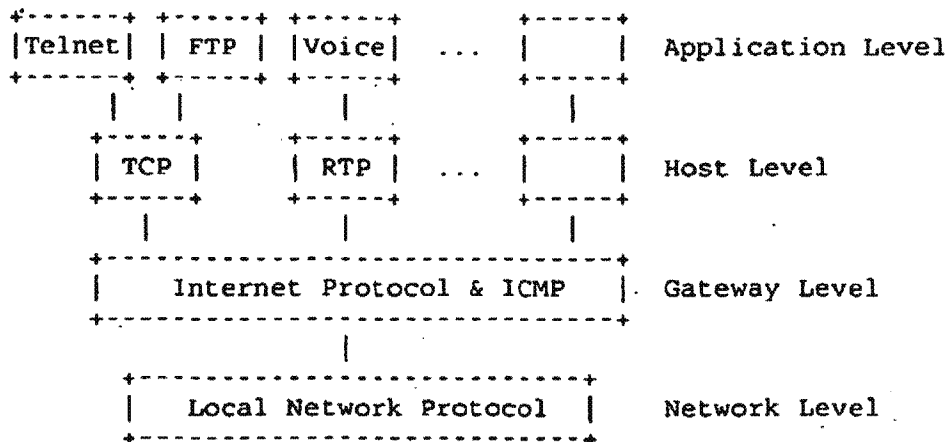
previous packet of the message; performing the processing of the identified component with the packet and the retrieved state information; and storing state information relating to the processing of the component with the packet for use when processing the next packet of the message").

However, Meer96 fails to overcome the deficiencies of Pfeifer96, ...for the **first packet** of the message, **identifying** a sequence of components for processing the packets of the message... as in claim 1, ...**identifying** a sequence of components for processing each message based on the **first packet** of the message so that subsequent packets of the message can be processed without re-identifying the components, ... as in claim 15 and ...**identifying** a message-specific sequence of components for processing the packets of each message upon receiving the **first packet** of the message wherein subsequent packets of the message can use the message-specific sequence identified when the **first packet** was received... as in claim 35.

Hence, for the reasons cited above, it is found that the requester has **not** shown a reasonable likelihood of prevail with respect to claims 1, 15 and 35.

ISSUE 8

RFC 793 discloses (Request, page 99) "RFC 793 merely confirms that certain information regarding the stateful operation of TCP would have been part of the standard background knowledge of those of ordinary skill in the art".



Protocol Relationships

Figure 2.

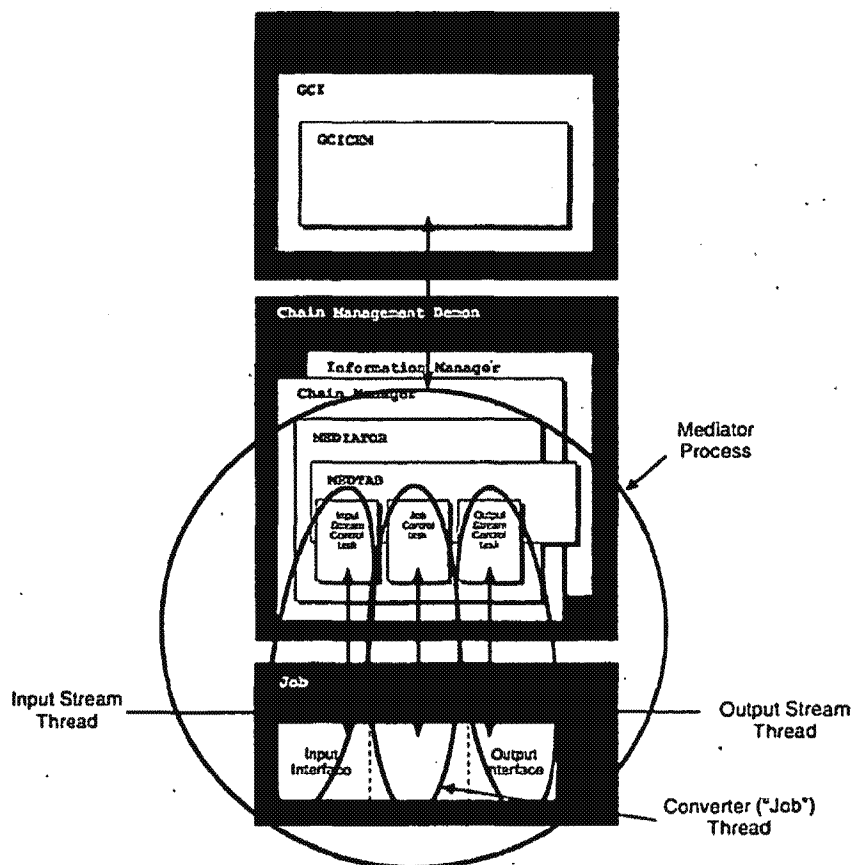
RFC 793 page 2 states The TCP is intended to provide a reliable process-to-process communication service in a multinet environment. The TCP is intended to be a host-to-host protocol in common use in multiple networks.

However, RFC 793 fails to overcome the deficiencies of Pfeifer96, ...for the **first packet** of the message, **identifying** a sequence of components for processing the packets of the message... as in claim 1, ...**identifying** a sequence of components for processing each message based on the **first packet** of the message so that subsequent packets of the message can be processed without re-identifying the components, ... as in claim 15 and ...**identifying** a message-specific sequence of components for processing the packets of each message upon receiving the **first packet** of the message wherein subsequent packets of the message can use the message-specific sequence identified when the **first packet** was received... as in claim 35.

Hence, for the reasons cited above, it is found that the requester has not shown a reasonable likelihood of prevail with respect to claims 1, 15 and 35.

ISSUE 9

Franz98 teaches (Request, page 102) an elaborate and systematic analysis of the various software building blocks that would be needed for "Job Control and Stream Control" in the iPCSS. See, e.g., id. At 19 ("Types of Operating Systems" section), 50 ("Programs and Jobs" section"), 51 ("Processes" section), 55 ("Threads" section). At the end of this lengthy analysis, Franz98 presents its conclusions about how the iPCSS should be structured, based on these building blocks. See Id. at 91-112 (chapter entitled "Realisation"). In this "Realisation" chapter, Franz98 recapitulates some iPCSS architectural concepts which would be familiar to readers of Pfeifer96.



Id. at 111 (Figure 6-9: "The final structure of the implementation") (showing structure for a

Franz98 teaches (Request, page 105) this state information (including at least "the program counter of the CPU" and the "used set of registers of the CPU") is clearly "information relating to performing the processing of the component": indeed, the program counter and registers would change in response to virtually instruction performed in the course of the converter's processing of a packet, so their state at the moment they were saved would clearly relate to that previous processing.

However, Franz98 fails to overcome the deficiencies of Pfeifer96, ...for the first packet of the message, identifying a sequence of components for processing the packets of the message... as in claim 1, ...identifying a sequence of components for

Art Unit: 3992

processing each message based on the *first packet* of the message so that subsequent packets of the message can be processed without re-identifying the components, ... as in claim 15 and ...*identifying* a message-specific sequence of components for processing the packets of each message upon receiving the *first packet* of the message wherein subsequent packets of the message can use the message-specific sequence identified when the *first packet* was received... as in claim 35.

Hence, for the reasons cited above, it is found that the requester has not shown a reasonable likelihood of prevail with respect to claims 1, 15 and 35.

ISSUE 10

Requester submits that (pages 107-108) Pfeifer96 teaches an "iPCSS" System wherein a converter chain is "dynamically generated" only after the first packet of a message is received. It also discloses and renders obvious that the converter components would maintain "state information" in the manners recited by claims 1, 15, and 35. It does so in several manners, including through use of ISDN connection converter components (which would maintain state information in order to execute the stateful ISDN protocol), and through use of components which perform compression or decompression. ISDN98 confirms ISDN connections are stateful. Nelson confirms that obvious implementations of compression/decompression algorithms for use with Pfeifer96 would be stateful. Cox teaches an "invocation-based metering" approach to software revenue collection which would be obvious to apply to Pfeifer96, and

Art Unit: 3992

maintaining a cumulative invocation count would entail maintaining "state information." Meet97 explains that in the iPCSS system, a portion of every converter component could be located across a stateful network connection (e.g., a TCP connection), which would require maintaining "state information" for each. RFC 793 confirms such a TCP connection would be stateful. Franz98 explains that in the iPCSS system, every converter component would maintain state information across packets because of the operating system "threading" structure used for the convener component jobs.

However, ISDN98, Nelson, Cox, Meer96, RFC 793, and Franz98 fail to overcome the deficiencies of Pfeifer96, ...for the first packet of the message, identifying a sequence of components for processing the packets of the message... as in claim 1, ...identifying a sequence of components for processing each message based on the first packet of the message so that subsequent packets of the message can be processed without re-identifying the components, ... as in claim 15 and ...identifying a message-specific sequence of components for processing the packets of each message upon receiving the first packet of the message wherein subsequent packets of the message can use the message-specific sequence identified when the first packet was received... as in claim 35.

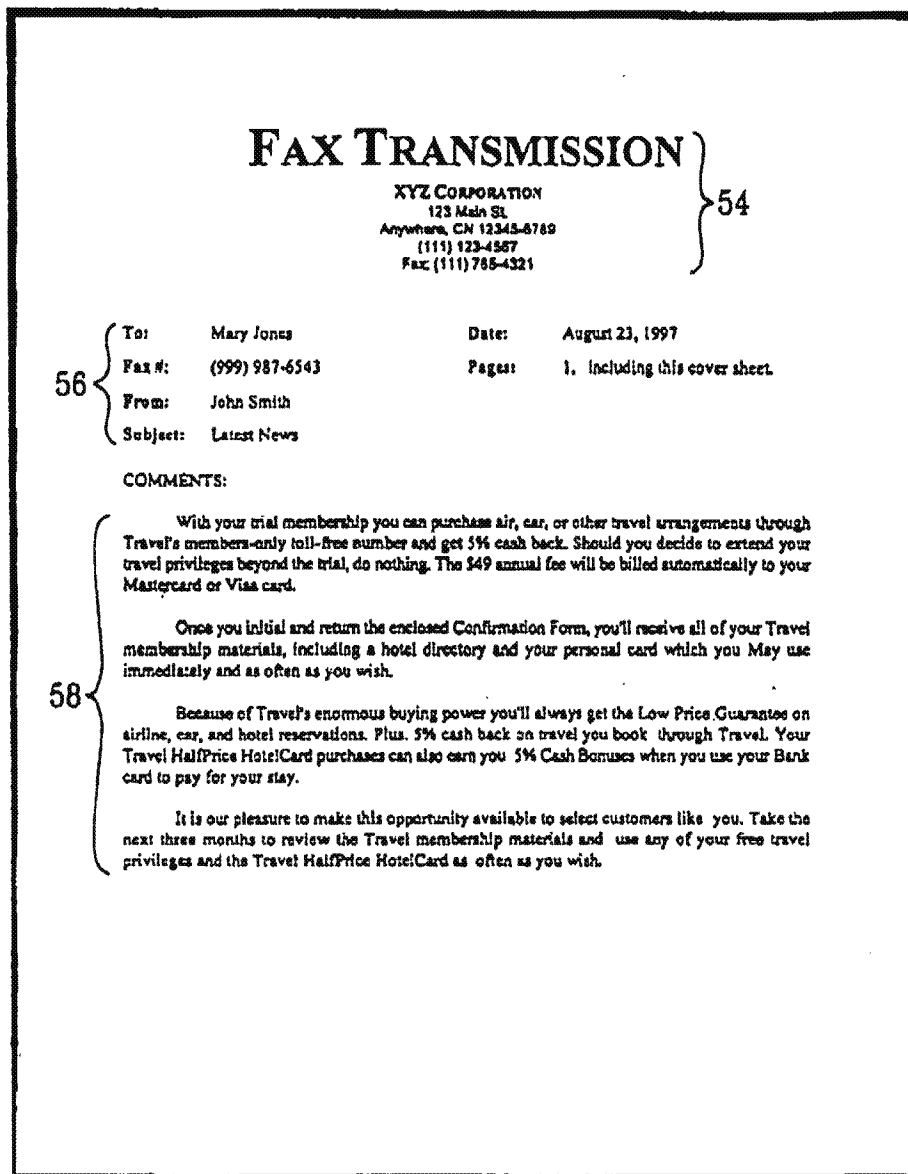
Hence, for the reasons cited above, it is found that the requester has not shown a reasonable likelihood of prevail with respect to claims 1, 15 and 35.

ISSUE 11

Art Unit: 3992

Alam discloses FIG. 4 depicts an exemplary fax 52 such as that an image of which may be received and processed by the fax server 20 depicted in FIG. 1. A sender identifier header 54 appears near the top of the fax 52. Immediately beneath the sender identifier header 54 is an addressing block 56 that includes names of both of the addressee and of the sender of the fax 52. Beneath the addressing block 56 is a text area 58 which occupies the remainder of the fax 52.

Art Unit: 3992



52

FIG. 4

The techniques that locate the address in the image data of the fax 52 includes rules for image cleanup (de-skewing, shade removal) geometric analysis (line identification, block bounding box detection), and feature pattern analysis (attribute-

Art Unit: 3992

value pair detection). A rules engine combines the result of this feature extraction process and removes blocks of the image from further analysis.

The Request submits (pages 108-109) Alam confirms such routing would indeed be enabled, and supplies additional detail on how the called party of such an incoming fax could be obtained. Specifically, Alam teaches that the fax image may be scanned, e.g., "to locate name fields..., based upon their nearness to and relationship with keywords. Keywords associated with the addressee's name such as 'To,' 'Recipient,' 'Attn' or 'Dear' point to the addressee name." Ex. 13 at 9:15-21. Once the destination party is determined, the iPCSS is clearly capable of determining that user's location and routing the communication to a terminal in the user's vicinity. E.g., Ex. A02 at 119, 123-24.

However, Alam fails to overcome the deficiencies of Pfeifer96, ...for the ***first packet*** of the message, ***identifying*** a sequence of components for processing the packets of the message... as in claim 1, ...***identifying*** a sequence of components for processing each message based on the ***first packet*** of the message so that subsequent packets of the message can be processed without re-identifying the components, ... as in claim 15 and ...***identifying*** a message-specific sequence of components for processing the packets of each message upon receiving the ***first packet*** of the message wherein subsequent packets of the message can use the message-specific sequence identified when the ***first packet*** was received... as in claim 35.

Hence, for the reasons cited above, it is found that the requester has not shown a reasonable likelihood of prevail with respect to claims 1, 15 and 35.

ISSUE 12

In regards to Yun, Request submits in page 109, [B]y teaching an "apparatus for discriminating a received audio signal as vocal sound or musical sound," Yun suggests how that specific conversion would be implemented and applied in practice. See Ex. 14 at Abstract. For example, example, incoming audio communications could be routed in one manner if they contain music (e.g., to a screen for viewing "music notes on video"), and in another if they contain voice (e.g., to a "speech recognition" component). E.g., Ex. 12 at 6 ("display of music notes"; "speech recognition"). Such a conversion involving audio to video conversion would read on claims 1, 15, and 35. See, e.g., Section V.A.1 (Pfeifer96 102) at Claim 1.

However, Yun fails to overcome the deficiencies of Pfeifer96, ...for the ***first packet*** of the message, ***identifying*** a sequence of components for processing the packets of the message... as in claim 1, ...***identifying*** a sequence of components for processing each message based on the ***first packet*** of the message so that subsequent packets of the message can be processed without re-identifying the components, ... as in claim 15 and ...***identifying*** a message-specific sequence of components for processing the packets of each message upon receiving the ***first packet*** of the message wherein subsequent packets of the message can use the message-specific sequence identified when the ***first packet*** was received... as in claim 35.

Hence, for the reasons cited above, it is found that the requester has not shown a reasonable likelihood of prevail with respect to claims 1, 15 and 35.

ISSUE 13

Request submits in page 110, if certain aspects recited in claims 1, 15, and 35 of the '163 patent are not deemed to be disclosed, inherent, suggested, or obvious over Pfeifer96 alone or in combination with the various grounds of rejection presented above, then the inclusion of those aspects certainly would be obvious over Pfeifer96 in view of Arbanowski96, Pfeifer97, ISDN98, Nelson, Cox, Meer96, RFC 793, Franz98, Alam, and Yun under 35 U.S.C. § 103, under Implicit's apparent claim Constructions. All of these references have already been combined with Pfeifer96 in corresponding sections above, and those sections should be consulted for the detailed manner of applying them to Pfeifer96. This section briefly summarizes that material and shows the collective combination of these references would be obvious as well.

However, Arbanowski96, Pfeifer97, ISDN98, Nelson, Cox, Meer96, RFC 793, Franz98, Alam, and Yun fail to overcome the deficiencies of Pfeifer96, ...for the first packet of the message, identifying a sequence of components for processing the packets of the message... as in claim 1, ...identifying a sequence of components for processing each message based on the first packet of the message so that subsequent packets of the message can be processed without re-identifying the components, ... as in claim 15 and ...identifying a message-specific sequence of components for processing the packets of each message upon receiving the first

Art Unit: 3992

packet of the message wherein subsequent packets of the message can use the message-specific sequence identified when the *first packet* was received... as in claim 35.

Hence, for the reasons cited above, it is found that the requester has not shown a reasonable likelihood of prevail with respect to claims 1, 15 and 35.

ISSUE 14

Request submits in pages 110-111, Pfeifer96, Arbanowski96, Pfeifer97, Meer96, and Franz98 collectively provide a comprehensive picture of the iPCSS platform, including its design and possible uses. Pfeifer96 teaches an "iPCSS" system wherein a converter chain is "dynamically generated" only after the first packet of a message is received. It also discloses and renders obvious that the convener components would maintain "state information" in the manners recited by claims 1, 15, and 35. It does so in several manners, including through use of ISDN connection converter components (which would maintain state information in order to execute the stateful ISDN protocol), and through use of components which perform compression or decompression. ISDN98 confirms ISDN connections are stateful. Nelson confirms that obvious implementations of compression/decompression algorithms for use with Pfeifer96 would be stateful. Cox teaches an "invocation-based metering" approach to software revenue collection which would be obvious to apply to Pfeifer96, and maintaining a cumulative invocation count would entail maintaining "state information." Meer96 explains that in the iPCSS system, a portion of every converter component could be located across a stateful network

Art Unit: 3992

connection (e.g., a TCP connection), which would require maintaining "state information" for each. RFC 793 confirms such a TCP connection would be stateful. Franz98 explains that in the iPCSS system, every convener component would maintain state information across packets because of the Operating system "threading" structure used for the converter component jobs. Alam and Yun provide additional on how specific conversions might be implemented and applied in practice.

However, Arbanowski96, Pfeifer97, ISDN98, Nelson, Cox, Meet96, RFC 793, Franz98, Alam, and Yun fail to overcome the deficiencies of Pfeifer96, ...for the ***first packet*** of the message, ***identifying*** a sequence of components for processing the packets of the message... as in claim 1, ...***identifying*** a sequence of components for processing each message based on the ***first packet*** of the message so that subsequent packets of the message can be processed without re-identifying the components, ... as in claim 15 and ...***identifying*** a message-specific sequence of components for processing the packets of each message upon receiving the ***first packet*** of the message wherein subsequent packets of the message can use the message-specific sequence identified when the ***first packet*** was received... as in claim 35.

Hence, for the reasons cited above, it is found that the requester has not shown a reasonable likelihood of prevail with respect to claims 1, 15 and 35.

ISSUE 15

In regards to claim 1, Kerr discloses a method and system for switching in networks responsive to message flow patterns. A message "flow" is defined to comprise a set of packets to be transmitted between a particular source and a particular destination. When routers in a network identify a new message flow, they determine the proper processing for packets in that message flow and cache that information for that message flow.

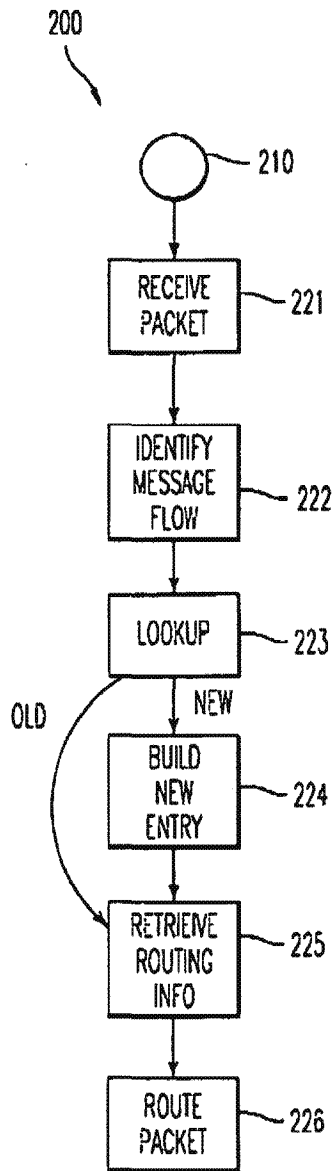


FIG. 2A

Thereafter, when routers in a network identify a packet which is part of that message flow, they process that packet according to the proper processing for packets in that message flow. The proper processing may include a determination of a

Art Unit: 3992

destination port for routing those packets and a determination of whether access control permits routing those packets to their indicated destination.

However, Kerr does not explicitly teach *processing the packets of the message such that the output format of the components match the input format of the next Component*.

Hence, for the reasons cited above, it is found that the requester has not shown a reasonable likelihood of prevail with respect to claim 1 at pages 112-118 of the Request.

In view of the description of Kerr above, in regards to claims 15 and 35, proposed rejection of claims 15 and 35, as set forth in pages 118-121 of the Request, is relied upon in the Request to show a reasonable likelihood that the requester will prevail with respect to at least one of the cited claims of the patent. Hence, for the reasons cited above, it is found that the requester has shown a reasonable likelihood of prevail with respect to claims 15 and 35.

ISSUE 16

In regards to claim 1, Request discloses in pages 124-125, regarding the limitation "such that the output format of the components ... match the input format of the next component," it was well-known to those of ordinary skill in the art that certain operations on a packet must be performed in a certain order: e.g., if a packet is first converted into an encrypted format by a first component, a subsequent component would be unable to, e.g., rewrite its headers (because it was expecting to receive the

Art Unit: 3992

packet in an unencrypted format). See *id.* at 4:31-32 ("encryption treatment for packets..., in the message flow"), 4:57-58 ("rewrite function for..., a header for the packet"). Thus, it was certainly at least obvious for one of ordinary skill in the art to arrange the sequence of components in a compatible manner, such that the output format of one matches the input format of the next-- rather than arranging them in an incompatible manner whereby various component(s) would be unable to perform their function(s).

In view of the description of Kerr above, in regards to claim 1, proposed rejection of claim 1, as set forth at pages 124-125 of the Request, is relied upon in the Request to show a reasonable likelihood that the requester will prevail with respect to claim 1 of the patent. Hence, for the reasons cited above, it is found that the requester has shown a reasonable likelihood of prevail with respect to claim 1.

In regards to claims 15 and 35, the Request submits (in pages 128-131) claims as being obvious over Kerr et al. (US PAT 6243667, hereinafter Kerr). However, in the Request pages 118-121, Requester has shown that all the limitations of claims 15 and 35 are met by Kerr. Requester has not shown in pages 128-131, what claim limitations of claims 15 and 35 are not taught by the Kerr prior art.

Hence, for the reasons cited above, it is found that the requester has not shown a reasonable likelihood of prevail with respect to claims 15 and 35.

ISSUE 17

Art Unit: 3992

In regards to claims 1, 15 and 35, Requester submits that (pages 131-133) if certain aspects recited in claims 1, 15, and 35 of the '163 patent are not deemed to be disclosed, inherent, or obvious over Mosberger (Examiner submits that most likely Requester meant Kerr), then the inclusion of those aspects certainly would be obvious over Kerr in view of NetFlow, under 35 U.S.C. § 103.

It was obvious to supplement the teachings of Kerr with NetFlow because Kerr is a Cisco patent, and NetFlow is an article in a trade publication illustrating how the architecture of Kerr manifested itself in an actual Cisco product feature (named "NetFlow") that was available on the market within the same time period.

Thus, to the extent that Kerr is deemed to lack inadequate disclosure of the relevant limitations for claims 1, 15, and 35, the combination of Kerr with NetFlow clearly makes up for any such perceived deficiency.

However, Requester has not shown which particular limitations of claims 1, 15 and 35 are not taught by Kerr, and as such why the rejection requires the steps of combining the teachings of NetFlow to Kerr.

Hence, for the reasons cited above, it is found that the requester has not shown a reasonable likelihood of prevail with respect to claims 1, 15 and 35.

ISSUE 18

In regards to claims 1, 15 and 35, Requester submits that (pages 133-138) if certain aspects recited in claims 1, 15, and 35 of the '163 patent are not deemed to be disclosed, inherent, or obvious over Kerr alone, then the inclusion of those aspects

Art Unit: 3992

certainly would be obvious over Kerr in view of RFC 1825 and RFC 1829, under 35 U.S.C. § 103.

It was obvious to supplement the teachings of Kerr with RFC 1825 because Kerr applies "encryption" to "IP" (Internet Protocol) packets, and RFC 1825 ("Security Architecture for the Internet Protocol") "describes the security mechanisms for IP version 4 (IPv4) and IP version 6 (IPv6) including "encryption." Ex. 15 (Kerr) at 3:5 ("IP (internet protocol)", 4:30-31; Ex. 26 (RFC 1825) at 1. It was obvious to supplement the teachings of Kerr and RFC 1825 with RFC 1829, because RFC 1829 teaches an encryption algorithm which "MUST" be supported as part of the RFC 1825 "Security Architecture." Ex. 26 (RFC 1825) at 10 (the encryption operation "MUST support the use of the Data Encryption Standard (DES) in Cipher-Block Chaining (CBC) Mode"), 21 (citing "RFC 1829": "The ESP DES-CBC Transform").

However, Requester has not shown which particular limitations of claims 1, 15 and 35 are not taught by Kerr, and as such why the rejection requires the steps of combining the teachings of RFC 1825 and RFC 1829 to Kerr.

Hence, for the reasons cited above, it is found that the requester has not shown a reasonable likelihood of prevail with respect to claims 1, 15 and 35.

ISSUE 19

In regards to claims 1, 15 and 35, Requester submits that (pages 138-143) if certain aspects recited in claims 1, 15, and 35 of the '163 patent are not deemed to be disclosed, inherent, or obvious over Kerr, then the inclusion of those aspects certainly

Art Unit: 3992

would be obvious over Kerr in view of Bellare97 and Bellare95, under 35 U.S.C. § 103. It was Obvious to supplement the teachings of Kerr with Bellare97, because Kerr discloses "encryption" of the packets of a flow, and Bellare97 discloses a specific encryption algorithm that could be used. Ex. 15 at 4:30-31. It was obvious to supplement the teachings of Kerr and Bellare97 with Bellare95, because Bellare95 teaches a similar authentication algorithm which could also be applied to the packets Of a flow. Bellare95 teaches another operation that would advantageous to apply to the packets of a flow- "Authentication"--and it was obvious that this operation by provided by a distinct software routine as well. Ex. 18 at.1 ("A message authentication scheme enables two parties sharing a key..., to authenticate their transmissions. This is one of the most widely used cryptographic primitives," and "as security concerns grow," "it may become even more so"). It was therefore obvious to employ such a stateful algorithm in an encryption component of Kerr, particularly since Kerr does not specify a particular encryption algorithm. It was therefore obvious to employ such a stateful algorithm in an authentication component of Kerr.

However, Examiner submits that "a particular encryption algorithm" is not a necessary component to meet the limitations of claims 1, 15 and 35. Requester has not shown which limitation of claims 1, 15 and 35, in particular requires such teaching.

Therefore, Requester has not shown which particular limitations of claims 1, 15 and 35 are not taught by Kerr, and as such why the rejection requires the steps of combining the teachings of Bellare97 and Bellare95 to Kerr.

Art Unit: 3992

Hence, for the reasons cited above, it is found that the requester has not shown a reasonable likelihood of prevail with respect to claims 1, 15 and 35.

ISSUE 20

In regards to claims 1, 15 and 35, Requester submits that (pages 143-148) if certain aspects recited in claims 1, 15, and 35 of the '163 patent are not deemed to be disclosed, inherent, or obvious over Kerr alone, then the inclusion of those aspects certainly would be obvious over Kerr in view of IBM96, under 35 U.S.C. § 103. It was obvious to supplement the teachings of Kerr with IBM96 because Kerr teaches a flow-based architecture for routing devices, and IBM96 teaches features which would have been typical of routing devices of the time period.

In view of these various benefits of data compression, it was obvious that in addition to supporting operations such as encryption and packet rewrite, Kerr should also support compression. Because Kerr teaches encryption is selectively applied to specific flows, it was obvious to treat compression in the same manner. E.g., Ex. 15 at 4:30-31.

IBM96 discusses and compares the Performance of four specific compression algorithms, the top three of which are all "LZ"-based compression algorithms. See Ex. 19 at 95-96 ("LZ77" has compression ratio of"2.08:1"; "Stacker-LZS" a ratio of"1.82:1 "; "BSD Compress-LZW" a ratio of"2.235:1 "; and "Predictor" a ratio of"1.67:1"). Because the top three algorithms discussed by IBM96 are LZ-based and because the "IBM 2210" router specifically uses the "LZ77" algorithm, an LZ-based algorithm such as LZ77

Art Unit: 3992

would have been an obvious choice for a compression component to be added to Kerr. Id. at 95-96, 84.

However, Requester has not shown which particular limitations of claims 1, 15 and 35 are not taught by Kerr, and as such why the rejection requires the steps of combining the teachings of IBM96 to Kerr.

Hence, for the reasons cited above, it is found that the requester has not shown a reasonable likelihood of prevail with respect to claims 1, 15 and 35.

ISSUE 21

In regards to claims 1, 15 and 35, Requester submits that (pages 148-152) if certain aspects recited in claims 1, 15, and 35 of the '163 patent are not deemed to be disclosed, inherent, or obvious over Kerr in view of IBM96, then the inclusion of those aspects certainly would be obvious over Kerr in view of IBM96 and Nelson, under 35 U.S.C. § 103.

It was obvious to supplement the teachings of Kerr and IBM96 with Nelson, because IBM96 disclose compression operations performed by routers, and Nelson teaches specific compression algorithms which might be used. Nelson explains: "Adaptive coding..., lead[s] to vastly improved compression ratios," and that "compression research in the last 10 years has concentrated on adaptive models." Ex. 5 at 8, 18. Adaptive algorithms include such well-known algorithms as "Adaptive Huffman Coding" (chapter 4; id. at 75), "Adaptive [Statistical] Modeling" (chapter 6; id. at 155), "[Adaptive] Dictionary-Based Compression" (chapter 7: id. at 203), and "Sliding Window

Art Unit: 3992

Compression" (chapter 8; id. at 215); and the prominent "LZ" family of Compression algorithms (chapter 8 and 9, id. at 221,255). All of these adaptive techniques are lossless, which would be important for accurately transmitting information contained in network packets. See id. at 9 ("All of the compression techniques discussed through chapter 9 are 'lossless'"). In view of the prominence, lossless nature, and improved Compression ratios of adaptive algorithms, use of such an algorithm would have been an obvious choice for a compression component. More narrowly, IBM96 teaches that its "2210" router employs the "LZ77" compression algorithm, so use of that algorithm in particular would have been an obvious design decision over IBM96. See Ex. 19 (IBM96) at 95-96, 84. Nelson confirms this algorithm was stateful and "adaptive" in the manner described above. See, e.g., Ex. 5 at 21 ("LZ77" maintains a "dictionary" comprised of, e.g., a sliding "4K-byte window" of the most recently seen data).

However, Requester has not shown which particular limitations of claims 1, 15 and 35 are not taught by Kerr, and as such why the rejection requires the steps of combining of the teachings of IBM96 and Nelson to Kerr.

Hence, for the reasons cited above, it is found that the requester has not shown a reasonable likelihood of prevail with respect to claims 1, 15 and 35.

ISSUE 22

In regards to claims 1, 15 and 35, Requester submits that (pages 153-155) all of these references have already been combined with Kerr in corresponding sections above, and those sections should be consulted for the detailed manner of applying them

Art Unit: 3992

to Kerr. This section briefly summarizes that material and shows the collective combination of these references would be obvious as well.

Kerr teaches a general flow-based architecture for router devices which applies, e.g., encryption, packet re-write, and any other "special treatment" to the packets of specific flows. E.g., Ex. 15 at 4:29-60.

RFC 1825 and Bellare95 confirm the obviousness of employing an additional component for authentication.

IBM96 confirms the obviousness of employing an additional component for compression. Since Kerr teaches that its various possible operations are applied in a tailored manner to each particular flow (see *id.* at 4:12-20), it was obvious that any two or more of these three types of plugins (encryption, authentication, compression) might be applied to the same flow. This is especially obvious since all three of those operations would be useful for implementing, e.g., a virtual private network across an expensive link, as would be appreciated by one of ordinary skill in the art.

RFC 1829 and Bellare97 confirm the obviousness of employing a stateful encryption algorithm which would read on these elements.

Bellare95 confirms the obviousness of employing a stateful authentication algorithm which would read on these elements.

Nelson confirms the obviousness of employing a stateful compression algorithm which would read on these elements.

Claim 1 recites each component "being a software routine for converting data with an input format into data with an output format." Performing encryption on a packet

Art Unit: 3992

would convert it from an unencrypted to an encrypted format, and likewise performing compression on a packet would convert it from an uncompressed to a compressed format. Both of these operations would read on this "converting data" element.

Bellare95 confirms that performing authentication on a packet would entail inserting an extra field into the packet, which would also read on this "converting data" element.

Finally, in addition to the specific plugin components discussed immediately above (encryption, authentication, compression), Kerr discloses a number of other components which would read on the "state information" and/or "format" claim elements of claims 1, 15, and 35, including plugin components for packet rewrite, accounting, and traffic profiling functions. See Sections V.B. 1 (Kerr 102) and V.B.2 (Kerr 103) above.

Since Kerr teaches that its various possible operations are applied in a tailored manner to each particular flow, it was obvious for any of these various components to be applied to the same flow as well, in addition to (or instead of) any of the encryption, authentication, or compression components discussed immediately above.

However, Examiner submits that Requester has not shown which particular limitations of claims 1, 15 and 35 are not taught by Kerr, and as such why the rejection requires the steps of combining the teachings of RFC 1825, RFC 1829, Bellare97, Bellare95, IBM96, and Nelson to Kerr.

Hence, for the reasons cited above, it is found that the requester has not shown a reasonable likelihood of prevail with respect to claims 1, 15 and 35.

ISSUE 23

In regards to claims 1, 15 and 35, Requester submits that (pages 155-158) if certain aspects recited in claims 1, 15, and 35 of the '163 patent are not deemed to be disclosed, inherent, or obvious over Kerr alone, then the inclusion of those aspects certainly would be obvious over Kerr in view of Bellissard, under 35 U.S.C. § 103. It was obvious to supplement the teachings of Kerr with Bellissard because Kerr teaches a general flow-based architecture for routers and firewalls (e.g., Ex. 15 at 4:12-48), and Bellissard teaches a technique for enhancing the dynamic extensibility of such an architecture. Kerr alone renders obvious this element. See Section V.B.2 (Kerr 103) at Claim 1. As applied to Kerr, Bellissard further underscores the "dynamic[]" nature of the identification, under Implicit's apparent claim constructions, as explained below.

It was particularly obvious to apply the technique of Bellissard to the router/firewall architecture of Kerr, because a "firewall" is precisely the example chosen by Bellissard of "a typical full-size application" which would "emphasize the benefits of" the Bellissard technique. Id. at 1 ; Ex. 15 (Kerr) at 4:45-46 (also "useful for implementing security 'firewalls'").

To summarize, the combination of Kerr and Bellissard renders obvious a system in which components of Kerr could be dynamically modified or dynamically added at any moment during runtime--while the system was still operating----and could thereby take advantage of the newly added or modified components. Under Implicit's apparent claim

Art Unit: 3992

constructions, such a system would clearly read on "dynamically identifying a non-predefined sequence of components for processing the packets of the message."

However, Requester has not shown which particular limitations of claims 1, 15 and 35 are not taught by Kerr, and as such why the rejection requires the steps of combining of the teachings of Bellissard to Kerr.

Hence, for the reasons cited above, it is found that the requester has not shown a reasonable likelihood of prevail with respect to claims 1, 15 and 35.

ISSUE 24

In regards to claims 1, 15 and 35, Requester submits that (pages 158-162) if certain aspects recited in claims 1, 15, and 35 of the '163 patent are not deemed to be disclosed, inherent, or obvious over Kerr alone, then the inclusion of those aspects certainly would be obvious over Kerr in view of Fraser, under 35 U.S.C. § 103. It was obvious to supplement the teachings of Kerr with Fraser because Kerr teaches a general flow-based architecture for routers and firewalls (e.g., Ex. 15 at 4:12-48), and Fraser teaches a technique for enhancing the dynamic configurability of such an architecture. Kerr alone renders obvious this element. See Section V.B.2 (Kerr 103) at Claim 1. As applied to Kerr, Fraser further underscores the "dynamic[]" nature of the identification, under Implicit's apparent claim constructions, as explained below.

Fraser teaches "Dynamic Policy Modules" which an administrator uses to control the behavior of a firewall: e.g., these modules define which traffic flowing through the firewall should be encrypted, and which network destinations should be accessible to

Art Unit: 3992

which users. Ex. 24 at 10, 6-7. It was obvious to apply the Dynamic Policy Modules framework of Fraser to Kerr, in order to provide a more comprehensive framework¹⁸ for avoiding any "undesirable and impractical" need to reboot the Kerr device under any circumstances. See *id.* at 9. Kerr was an especially obvious candidate for this technique, because Fraser uses the technique to control the policies of "firewall[s]," and Kerr teaches an architecture that is "useful for implementing security 'firewalls'." *Id.* at 6; Ex. 15 at 4:45-46.

As applied to Kerr, Dynamic Policy Modules would allow an administrator to modify the policies which determine which components are assigned to which flows. See, e.g., Ex. 15 (Kerr) at 4:13-19, 7:47-54. The parallels between the two systems are particularly clear on this point. For example, Fraser's Dynamic Policy Modules control, e.g., which traffic is encrypted, and Kerr's policies control, e.g., which flows are encrypted. Ex. 24 at 7, Ex. 15 at 4:12-34.

To summarize, the combination of Kerr and Fraser renders further obvious a system in which the policies determining the identified sequence of plugin components could be dynamically modified or dynamically added at any moment during runtime-- while the system was still operating. Under Implicit's apparent claim constructions, such a system would clearly read on "dynamically identifying a non-predefined sequence of components for processing the packets of the message."

However, Requester has not shown which particular limitations of claims 1, 15 and 35 are not taught by Kerr, and as such why the rejection requires the steps of combining of the teachings of Fraser to Kerr.

Hence, for the reasons cited above, it is found that the requester has not shown a reasonable likelihood of prevail with respect to claims 1, 15 and 35.

ISSUE 25

In regards to claims 1, 15 and 35, Requester submits that (pages 162-165) if certain aspects recited in claims 1, 15, and 35 of the '163 patent-are not deemed to be disclosed, inherent, suggested, or obvious over Kerr alone or in combination with the various grounds of rejection presented above, then the inclusion of those aspects certainly would be obvious over Kerr in view of RFC 1825, RFC 1829, Bellare97, Bellare95, IBM96, Nelson, Bellissard, and Fraser under 35 U.S.C. § 103, under Implicit's apparent claim constructions. All of these references have already been combined with Kerr in corresponding sections above, and those sections should be consulted for the detailed manner of applying them to Kerr.

This section briefly summarizes that material and shows the collective combination of these references would be obvious as well.

Kerr teaches a general flow-based architecture for router devices which applies, e.g., encryption, packet re-write, and any other "special treatment" to the packets of specific flows. E.g., Ex. 15 at 4:29-60.

RFC 1825 and Bellare95 confirm the obviousness of employing an additional component for authentication.

IBM96 confirms the obviousness of employing an additional component for compression. Since Kerr teaches that its various possible operations are

Art Unit: 3992

applied in a tailored manner to each particular flow (see *id.* at 4; 12-20), it was obvious that any two or more of these three types of plugins (encryption, authentication, compression) might be applied to the same flow. This is especially obvious since all three of those operations would be useful for implementing, e.g., a virtual private network across an expensive link, as would be appreciated by one of ordinary skill in the art.

RFC 1829 and Bellare97 confirm the obviousness of employing a stateful encryption algorithm which would read on these elements.

Bellare95 confirms the obviousness of employing a stateful authentication algorithm which would read on these elements.

Nelson confirms the obviousness of employing a stateful compression algorithm which would read on these elements. Claim 1 recites each component "being a software routine for converting data with an input format into data with an output format." Performing encryption on a packet would convert it from an unencrypted to an encrypted format, and likewise performing compression on a packet would convert it from an uncompressed to a compressed format. Both of these operations would read on this "converting data" element, under Implicit's apparent claim constructions. Bellare95 confirms that performing authentication on a packet would entail inserting an extra field into the packet, which would also read on this "converting data" element, under Implicit's apparent claim constructions.

Finally, in addition to the specific plugin components discussed immediately above (encryption, authentication, compression), Kerr discloses a number of other

Art Unit: 3992

components which would read on the "state information" and/or "format" claim elements of claims 1, 15, and 35, including plugin components for packet rewrite, accounting, and traffic profiling functions. See Sections V.B.1 (Kerr 102) and V.B.2 (Kerr 103) above. Since Kerr teaches that its various possible operations are applied in a tailored manner to each particular flow, it was obvious for any of these various components to be applied to the same flow as well, in addition to (or instead of) any of the encryption, authentication, or compression components discussed immediately above.

Claims 1, 15, and 35 recite "dynamically identifying a... non-predefined sequence of components." Kerr alone makes clear that administrators can make rule-based or policy changes during runtime, which falls within the scope of "dynamically identifying a non-predefined sequence of components" under Implicit's apparent claim construction. E.g., Ex. 15 at 6:14-16 ("changes in access control lists" can occur during an existing "flow," causing it to "expire"), 8:42-44 (after being "initially configured," routing device parameters "may be altered by an operator").

Bellissard teaches dynamically adding new components and modifying existing components while the system is operating. Under Implicit's apparent claim constructions, both of these techniques would read on these "dynamic[]" claim elements.

Like Kerr, Fraser teaches dynamically configuring firewall policies while the system is operating. It teaches a more comprehensive framework for this capability, and details another manner in which it could be implemented. Under Implicit's apparent

Art Unit: 3992

claim constructions; such dynamic configuration of policies would read on these "dynamic[]" claim elements.

However, Examiner submits that Requester has not shown which particular limitations of claims 1, 15 and 35 are not taught by Kerr, and as such why the rejection requires the steps of combining the teachings of RFC 1825, RFC 1829, Bellare97, Bellare95, IBM96, Nelson, Bellissard, and Fraser to Kerr.

Hence, for the reasons cited above, it is found that the requester has not shown a reasonable likelihood of prevail with respect to claims 1, 15 and 35.

ISSUE 26

In regards to claims 1, 15 and 35, Requester submits that (pages 165-167) if certain aspects recited in claims 1, 15, and 35 of the '163 patent are not deemed to be disclosed, inherent, or obvious over Kerr, then the inclusion of those aspects certainly would be obvious over Kerr in view of Checkpoint, and further in view of Shwed, under 35 U.S.C. § 103. Under Implicit's apparent claim construction, the "dynamically" limitation requires some degree of system configurability, and Kerr duly discloses a fully configurable network security product. However, if Kerr is deemed to lack sufficient disclosure regarding system Configurability, combination with Checkpoint and Shwed cures any such deficiency. Checkpoint and Shwed illustrate the fact that network security products such as firewalls have had the ability to arbitrarily add and change rules and policies for years prior to the filing date of the ' 163 patent. And it would, have been obvious to apply the teachings of Checkpoint and Shwed to the networking

Art Unit: 3992

technologies in Kerr, to provide yet additional configurability options to address changing security demands in a network environment.

Thus, to the extent that Kerr is deemed to lack inadequate disclosure of the relevant limitations for claims 1, 15, and 35, the combination of Kerr with Checkpoint and Shwed clearly makes up for any such perceived deficiency.

However, Requester has not shown which particular limitations of claims 1, 15 and 35 are not taught by Kerr, and as such why the rejection requires the steps of combining of the teachings of Checkpoint and Shwed to Kerr.

Hence, for the reasons cited above, it is found that the requester has not shown a reasonable likelihood of prevail with respect to claims 1, 15 and 35.

ISSUE 27

In regards to claims 1, 15 and 35, Requester submits that (pages 167-169) if certain aspects recited in claims 1, 15, and 35 of the '163 patent are not deemed to be disclosed, inherent, or obvious over Kerr, then the inclusion of those aspects certainly would be obvious over Kerr in view of Dietz, under 35 U.S.C. § 103. For example, Dietz, like Kerr, is expressly described as a "flow"-based system, as illustrated in Figure 3, and thus it would have been obvious to jointly consider their combined teachings. Thus, to the extent that Kerr is deemed to lack inadequate disclosure of the relevant limitations for claims 1, 15, and 35, the combination of Kerr with Dietz clearly makes up for any such perceived deficiency.

Art Unit: 3992

However, Requester has not shown which particular limitations of claims 1, 15 and 35 are not taught by Kerr, and as such why the rejection requires the steps of combining of the teachings of Dietz to Kerr.

Hence, for the reasons cited above, it is found that the requester has not shown a reasonable likelihood of prevail with respect to claims 1, 15 and 35.

ISSUE 28

In regards to claims 1, 15 and 35, Requester submits that (pages 169-177) if certain aspects recited in claims 1, 15, and 35 of the '163 patent are not deemed to be disclosed, inherent, or obvious over Kerr alone, then the inclusion of those aspects certainly would be obvious over Kerr in view of Pfeifer96, under 35 U.S.C. § 103. It was obvious to apply this system of Pfeifer96 to Kerr, so Kerr could assure delivery of incoming communications to users in their actual locations. Thus, considering Kerr, in view of Pfeifer96 essentially poses this question to Kerr: knowing and tracking all this information about each flow (including its intended destination device, its source medium), and being responsible for routing the flow onward to its intended destination-- what should be done if the user is not in the vicinity of the destination device? Clearly, an obvious answer is to apply the system of Pfeifer96, whereby a flow can be re-routed and converted for connection to a device at the user's current location, rather than terminating uselessly at a device in a vacant office. This obviousness is further heightened by the straightforward compatibility of the two architectures: the one would fit into the other seamlessly. The combination of Kerr and Pfeifer would also render

Art Unit: 3992

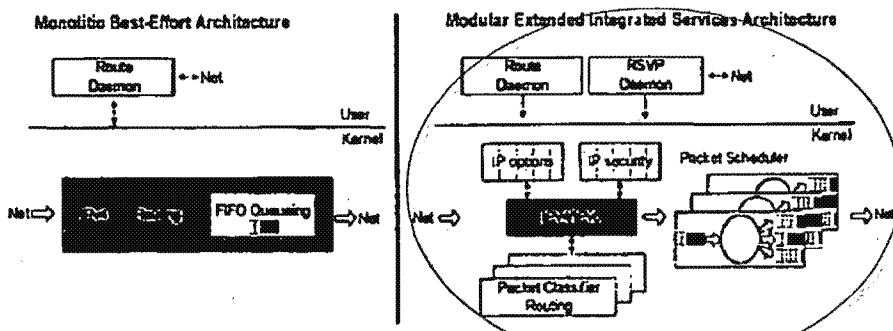
obvious claims 15 and 35, for the reasons set forth immediately above as to claim 1, and in light of the fact that both Kerr and Pfeifer separately disclose every limitation of claims 15 and 35 for the reasons set forth in Section V.A.1 and V.B.1.

However, Requester has not shown which particular limitations of claims 1, 15 and 35 are not taught by Kerr, and as such why the rejection requires the steps of combining of the teachings of Pfeifer96 to Kerr.

Hence, for the reasons cited above, it is found that the requester has not shown a reasonable likelihood of prevail with respect to claims 1, 15 and 35.

ISSUE 29

In regards to claim 1, Decasper98 teaches present day routers typically employ monolithic operating systems which are not easily upgradable and extensible. With the rapid rate of protocol development it is becoming increasingly important to dynamically upgrade router software in an incremental fashion.



**Figure 1. : Best Effort-vs
Extended Integrated Services Router (EISR)**

Decasper98 designed and implemented a high performance, modular, extended integrated services router software architecture in the NetBSD operating system kernel. This architecture allows code modules, called plugins, to be dynamically added and configured at run time. One of the novel features of our design is the ability to bind different plugins to individual flows; this allows for distinct plugin implementations to seamlessly coexist in the same runtime environment.

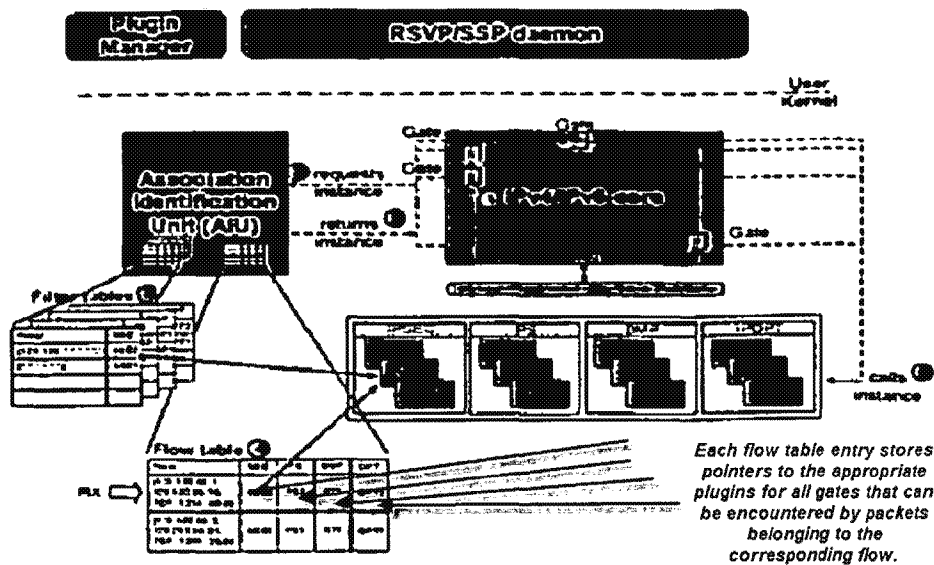


Figure 3. : System Architecture and Data Path

High performance is achieved through a carefully designed modular architecture; an innovative packet classification algorithm that is both powerful and highly efficient; and by caching that exploits the flow-like characteristics of internet traffic.

However, Decasper98 does not explicitly teach *converting data with an input format into data with an output format; processing the packet of the message such that the output format matches the input format of the next component.*

Art Unit: 3992

Hence, for the reasons cited above, it is found that the requester has not shown a reasonable likelihood of prevail with respect to claim 1.

In view of the description of Decasper98 above, in regards to claims 15 and 35, proposed rejection of claims 15 and 35, as set forth in pages 189-192 of the Request, is relied upon in the Request to show a reasonable likelihood that the requester will prevail with respect to at least one of the said cited claims of the patent. Hence, for the reasons cited above, it is found that the requester has shown a reasonable likelihood of prevail with respect to claims 15 and 35.

ISSUE 30

In regards to claim 1, Request discloses in pages 192-202, regarding the limitation "such that the output format of the components ... match the input format of the next component," it was well-known to those of ordinary skill in the art that certain operations on a packet must be performed in a certain order: e.g., if a packet is first converted into an encrypted format by a first component, a subsequent component would be unable to, e.g., process any IPv6 option headers in the packet, or to insert any new ones (because it was expecting to receive the packet in an unencrypted format). Thus, it was certainly obvious for one of ordinary skill in the art to arrange the sequence of components in a compatible manner, such that the output format of one matches the input format of the next.

In view of the description of Decasper98 above, in regards to claim 1, proposed rejection of claim 1, as set forth at pages 192-202 of the Request, is

Art Unit: 3992

relied upon in the Request to show a reasonable likelihood that the requester will prevail with respect to claim 1 of the patent. Hence, for the reasons cited above, it is found that the requester has shown a reasonable likelihood of prevail with respect to claim 1.

In regards to claims 15 and 35, the Request submits (in pages 202-205) claims as being obvious over Decasper98. However, in the Request pages 189-192, Requester has shown that all the limitations of claims 15 and 35 are met by Decasper98. Requester has not shown in pages 202-205, what claim limitations of claims 15 and 35 are not taught by the Decasper98 prior art.

Hence, for the reasons cited above, it is found that the requester has not shown a reasonable likelihood of prevail with respect to claims 15 and 35.

ISSUE 31

In regards to claims 1, 15 and 35, Requester submits that (pages 205-210) if certain aspects recited in claims 1, 15, and 35 of the '163 patent are not deemed to be disclosed, inherent, or obvious over Decasper98 alone, then the inclusion of those aspects certainly would be obvious over Decasper98 in view of RFC 1825 and RFC 1829, under 35 U.S.C. § 103. It was obvious to supplement the teachings of Decasper98 with RFC 1825 and RFC 1829 because Decasper98 expressly cites RFC 1825 to explain its "plugins for IP Security," and RFC 1825 expressly cites RFC 1829 to explain an algorithm which "MUST" be supported for encrypting packets. Ex. 25 (Decasper98) at 2 ("plugins for IP Security" citing footnote "[2]"), 12 (footnote "[2]" citing

Art Unit: 3992

"RFC 1825"); Ex. 26 (RFC 1825) at 10 ("the IP Encapsulating Security Payload MUST support the use of the Data Encryption Standard (DES) in Cipher-Block Chaining (CBC Mode"), 21 (citing RFC 1829: "The ESP DES-CBC Transform").

However, Requester has not shown which particular limitations of claims 1, 15 and 35 are not taught by Decasper98, and as such why the rejection requires the steps of combining the teachings of RFC 1825 and RFC 1829 to Decasper98.

Hence, for the reasons cited above, it is found that the requester has not shown a reasonable likelihood of prevail with respect to claims 1, 15 and 35.

ISSUE 32

In regards to claims 1, 15 and 35, Requester submits that (pages 211-215) if certain aspects recited in claims 1, 15, and 35 of the '163 patent are not deemed to be disclosed, inherent, or obvious over Decasper98, then the inclusion of those aspects certainly would be obvious over Decasper98 in view of RFC 1883 and Huitema, under 35 U.S.C. § 103. It was Obvious to supplement the teachings of Decasper98 with RFC 1883 and Huitema because Decasper98 discloses "plugins implementing IPv6 options," which are explained by RFC 1883 and Huitema. Ex. 25 at 4 ("plugins implementing IPv6 options"). Moreover, Decasper98 and Huitema expressly cite to RFC 1883. Id. at 12 (citation to "RFC 1883"); Ex 29 at 43.

However, Requester has not shown which particular limitations of claims 1, 15 and 35 are not taught by Decasper98, and as such why the rejection requires the steps of combining the teachings of RFC 1883 and Huitema to Decasper98.

Hence, for the reasons cited above, it is found that the requester has not shown a reasonable likelihood of prevail with respect to claims 1, 15 and 35.

ISSUE 33

In regards to claims 1, 15 and 35, Requester submits that (pages 215-220) if certain aspects recited in claims 1, 15, and 35 of the '163 patent are not deemed to be disclosed, inherent, or obvious over Decasper98 alone, then the inclusion of those aspects certainly would be obvious over Decasper98 in view of Decasper97, under 35 U.S.C. § 103. It was obvious to supplement the teachings of Decasper98 with Decasper97, because both describe a very similar architecture for dynamically loading router components on the basis of independent filters. Compare Ex. 25 (Decasper98) at 5 ("entries in the flow table"), 2 ("New plugins can be dynamically loaded at run time"), 5-7 (filter operation), 4 ("plugins implementing IPv6 options, plugins for packet scheduling..., and plugins for IP security"); Ex. 30 (Decasper97) at 4 ("Flow entries"), 3 ("dynamically loadable modules"), 3-4 (filter operation), 3-4 (modules include "authentication modules..., encryption modules...IPv6 option modules... and packet scheduling modules."). Though it was obvious over Decasper98 alone to employ distinct components for encryption and authentication (since they are distinct operations not always performed together on the same packet), Decasper97 renders this even more obvious by teaching precisely that. See Ex. 30 at 3 ("Five different module types are supported in the initial version," including "authentication modules" and "encryption modules").

However, Requester has not shown which particular limitations of claims 1, 15 and 35 are not taught by Decasper98, and as such why the rejection requires the steps of combining the teachings of Decasper97 to Decasper98.

Hence, for the reasons cited above, it is found that the requester has not shown a reasonable likelihood of prevail with respect to claims 1, 15 and 35.

ISSUE 34

In regards to claims 1, 15 and 35, Requester submits that (pages 220-224) if certain aspects recited in claims 1, 15, and 35 of the '163 patent are not deemed to be disclosed, inherent, or obvious over Decasper98 in view of Decasper97, then the inclusion of those aspects certainly would be obvious over Decasper98 in view of Decasper97, Bellare97, and Bellare95, under 35 U.S.C. § 103. It was obvious to supplement, the teachings of Decasper98 and Decasper97 with Bellare97 and Bellare95, because Decasper98 and Decasper97 disclose encryption and authentication operations, and Bellare97 and Bellare95 disclose specific encryption (Bellare97) and authentication (Bellare95) algorithms which might be used. Decasper98 repeatedly emphasizes the "extensibility" of its platform and expressly declares: "Doubtless, additional plugin types will be introduced by third parties once we have released our code into the public domain." Ex. 25 at 6, 2, 3, 11. Thus, additional plugins implementing the algorithms of Bellare97 and Bellare95 would be exactly the sort of extensions supported and expected by Decasper98.

Art Unit: 3992

However, Requester has not shown which particular limitations of claims 1, 15 and 35 are not taught by Decasper98, and as such why the rejection requires the steps of combining the teachings of Decasper97, Bellare97, and Bellare95 to Decasper98.

Hence, for the reasons cited above, it is found that the requester has not shown a reasonable likelihood of prevail with respect to claims 1, 15 and 35.

ISSUE 35

In regards to claims 1, 15 and 35, Requester submits that (pages 224-229) if certain aspects recited in claims 1, 15, and 35 of the '163 patent are not deemed to be disclosed, inherent, or obvious over Decasper98 alone, then the inclusion of those aspects certainly would be obvious over Decasper98 in view of IBM96, under 35 U.S.C. § 103. It was obvious to supplement the teachings of Decasper98 with IBM96 because Decasper98 teaches a general, extensible platform for implementing routers, and IBM96 teaches features which would have been typical of routers of the time period.

However, Requester has not shown which particular limitations of claims 1, 15 and 35 are not taught by Decasper98, and as such why the rejection requires the steps of combining the teachings of IBM96 to Decasper98.

Hence, for the reasons cited above, it is found that the requester has not shown a reasonable likelihood of prevail with respect to claims 1, 15 and 35.

ISSUE 36

In regards to claims 1, 15 and 35, Requester submits that (pages 229-234) if certain aspects recited in claims 1, 15, and 35 of the '163 patent are not deemed to be disclosed, inherent, or obvious over Decasper98 in view of IBM96, then the inclusion of those aspects certainly would be obvious over Decasper98 in view of IBM96 and Nelson, under 35 U.S.C. § 103. It was obvious to supplement the teachings of Decasper98 and IBM96 with Nelson, because IBM96 disclose compression operations performed by routers, and Nelson teaches specific compression algorithms which might be used. Decasper98 repeatedly emphasizes the "extensibility" of its platform and expressly declares: "Doubtless, additional plugin types will be introduced by third parties once we have released our code into the public domain." Ex. 25 at 6, 2, 3, 11. Thus, an additional plugin implementing a compression algorithm would be exactly the sort of extension supported and expected by Decasper98. Thus, an obvious implementation of an adaptive algorithm would entail, for each packet, retrieving state information, using it to perform the compression processing, updating it to reflect the data in the most recent packet, and storing it so it can be applied to the next packet. More narrowly, IBM96 teaches that its "2210" router employs the "LZ77" compression algorithm, so use of that algorithm in particular would have been an obvious design decision over IBM96. See Ex. 19 (IBM96) at 95-96, 84. Nelson confirms this algorithm was stateful and "adaptive" in the manner described above. See, e.g., Ex. 5 at 21 ("LZ77" maintains a "dictionary" comprised of, e.g., a sliding "4K-byte window" of the most recently seen data).

Art Unit: 3992

However, Requester has not shown which particular limitations of claims 1, 15 and 35 are not taught by Decasper98, and as such why the rejection requires the steps of combining the teachings of IBM96 and Nelson to Decasper98.

Hence, for the reasons cited above, it is found that the requester has not shown a reasonable likelihood of prevail with respect to claims 1, 15 and 35.

ISSUE 37

In regards to claims 1, 15 and 35, Requester submits that (pages 234-236) if certain aspects recited in claims i, 15, and 35 of the '163 patent are not deemed to be disclosed, inherent, suggested, or obvious over Decasper98 alone or in combination With the various grounds of rejection presented above, then the inclusion of those aspects certainly would be obvious over Decasper98 in view of RFC 1825, RFC 1829, Decasper97, Bellare97, Bellare95, IBM96, and Nelson, under 35 U.S.C. § 103, under Implicit's apparent claim constructions. All of these references have already been combined with Decasper98 in corresponding sections above, and those sections should be consulted for the detailed manner of applying them to Decasper98. This section briefly summarizes that material and shows the collective combination of these references would be obvious as well.

Decasper98 teaches a general architecture for router/firewall plugins and repeatedly emphasizes its "extensibility." Ex. 25 at 1, 2, 3, 11, 6 ("Doubtless, additional plugin types will be introduced by third parties once we have released our code into the public domain."). Decasper98 teaches "plugins for IP security," and Deeasper97

Art Unit: 3992

confirms the obviousness of providing separate plugin components for encryption and authentication. IBM96 confirms the obviousness of an additional plugin component for compression. Since Decasper98 teaches that its plugin components are selected on the basis of separate, independent filter tables, it was obvious that any two or more of these three types of plugins (encryption, authentication, compression) might be applied to the same flow. This is especially obvious since all three operations would be useful for implementing, e.g., a virtual private network across an expensive link. See Ex. 25 (Decasper98) at 5 ("system is configured as entry point into a virtual private network").

RFC 1829 and Bellare97 confirm the obviousness of employing a stateful encryption algorithm which would read on these elements.

Bellare95 confirms the obviousness of employing a stateful authentication algorithm which would read on these elements.

Nelson confirms the obviousness of employing a stateful compression algorithm which would read on these elements.

RFC 1825 confirms the obviousness of inserting separate headers into a packet for both encryption and authentication, and this would read on this "converting data" element, under Implicit's apparent claim constructions. Performing compression on a packet would read on this "converting data" element as well, under Implicit's apparent claim constructions.

Finally, in addition to the specific plugin components discussed immediately above (encryption, authentication, compression), Decasper98 discloses a number of other plugin components which would read on the "state information" and/or "format"

Art Unit: 3992

claim elements of claims 1, 15, and 35, including plugin components for IPv6 options, statistics gathering, packet scheduling, and firewall functions. See Sections V.C.1 (Decasper98 102) and V.C.2 (Decasper98 103) above. Since Decasper98 teaches that its plugin components are selected on the basis of separate, independent filter tables, it was obvious for any of these various plugin components to be applied to the same flow as well, in addition to (or instead of) any of the encryption, authentication, or compression components discussed immediately above.

However, Requester has not shown which particular limitations of claims 1, 15 and 35 are not taught by Decasper98, and as such why the rejection requires the steps of combining the teachings of RFC 1825, RFC 1829, Decasper97, Bellare97, Bellare95, IBM96, and Nelson to Decasper98.

Hence, for the reasons cited above, it is found that the requester has not shown a reasonable likelihood of prevail with respect to claims 1, 15 and 35.

ISSUE 38

In regards to claims 1, 15 and 35, Requester submits that (pages 236-240) if certain aspects recited in claims 1, 15, and 35 of the '163 patent are not deemed to be disclosed, inherent, or obvious over Decasper98 alone, then the inclusion of those aspects certainly would be obvious over Decasper98 in view of Bellissard, under 35 U.S.C. § 103.

It was obvious to supplement the teachings of Decasper98 with Bellissard because Decasper98 teaches an extensible architecture for implementing firewalls and

Art Unit: 3992

routers, and Bellissard teaches a technique for enhancing the dynamic extensibility of such an architecture. While Decasper98 already teaches a platform wherein an administrator can dynamically add and configure components "even when network traffic is transiting through the system" (Ex. 25 at 9), Bellissard provides additional detail on how such a system could operate and on another way in which it could be implemented. Decasper98 alone renders obvious these elements. See Section V.C.2 (Decasper98 103) at Claim 1. As applied to Decasper98, Bellissard further underscores the "dynamic[]" nature of the identification, under Implicit's apparent claim constructions, as explained below. It was particularly obvious to apply the technique of Bellissard to the extensible router/firewall architecture of Decasper98, because a "firewall" is precisely the example chosen by Bellissard of "a typical full-size application" which would "emphasize the benefits of" the Bellissard technique. *Id.* at 1 ; Ex. 25 (Decasper98) at 2 ("Our framework is also very well suited to... security devices like Firewalls"). It was further obvious to apply the Bellissard technique of "dynamic reconfiguration" to Decasper98, because Decasper98 repeatedly emphasizes that the "extensibility" of its architecture which permits new components to be "dynamically loaded at run time." E.g., Ex. 25 at 2 ("Extensibility: New plugins can be dynamically loaded at run time"), 3 ("The primary goal of our proposed architecture was to build a modular and extensible networking subsystem that supported the concept of flows," including "Dynamic loading and unloading of plugins at run time into the networking subsystem."). To summarize, the combination of Decasper98 and Bellissard renders obvious a system in which the plugin components of Decasper98 could be dynamically

Art Unit: 3992

modified or dynamically added at any moment during runtime--while the system was still operating--and could thereby take advantage of the newly added or modified components. Under Implicit's apparent claim constructions, such a system would clearly read on "dynamically identifying a non-predefined sequence of components for processing the packets of the message."

However, Requester has not shown which particular limitations of claims 1, 15 and 35 are not taught by Decasper98, and as such why the rejection requires the steps of combining the teachings of Bellissard to Decasper98.

Hence, for the reasons cited above, it is found that the requester has not shown a reasonable likelihood of prevail with respect to claims 1, 15 and 35.

ISSUE 39

In regards to claims 1, 15 and 35, Requester submits that (pages 241-245) if certain aspects recited in claims 1, 15, and 35 of the '163 patent are not deemed to be disclosed, inherent, or obvious over Decasper98 alone, then the inclusion of those aspects certainly would be obvious over Decasper98 in view of Fraser, under 35 U.S.C. § 103. It was obvious to supplement the teachings of Decasper98 with Fraser because Decasper98 teaches an extensible architecture for implementing firewalls and routers, and Fraser teaches a technique for enhancing the dynamic configurability of such an architecture. While Decasper98 already teaches a platform wherein an administrator can dynamically configure policies (expressed in filters) "even when network traffic is transiting through the system" (Ex. 25 at9), Fraser teaches a more comprehensive framework for such a capability, and provides additional detail on how such a framework

Art Unit: 3992

would be implemented. It was obvious to apply the Dynamic Policy Modules framework of Fraser to Decasper98, in order to provide a more comprehensive framework³² for avoiding any "undesirable and impractical" need to reboot the Decasper98 device under any circumstances. See *id.* at 9. Decasper98 was an especially obvious candidate for this technique, because Fraser uses the technique to control the policies of "application gateway firewall[s]," and Decasper98 teaches an architecture that is "very well suited to Application Layer Gateways... and to security devices like Firewalls." *Id.* at 6; Ex. 25 at 2. To summarize, the combination of Decasper98 and Fraser renders further obvious a system in which the policies determining the identified sequence of plugin components could be dynamically modified or dynamically added at any moment during runtime-- while the system was still operating. Under Implicit's apparent claim constructions, such a system would clearly read on "dynamically identifying a non-predefined sequence of components for processing the packets of the message."

However, Requester has not shown which particular limitations of claims 1, 15 and 35 are not taught by Decasper98, and as such why the rejection requires the steps of combining the teachings of Fraser to Decasper98.

Hence, for the reasons cited above, it is found that the requester has not shown a reasonable likelihood of prevail with respect to claims 1, 15 and 35.

ISSUE 40

In regards to claims 1, 15 and 35, Requester submits that (pages 245-248) if certain aspects recited in claims 1, 15, and 35 of the '163 patent are not deemed to be

Art Unit: 3992

disclosed, inherent, suggested, or obvious over Decasper98 alone or in combination with the various grounds of rejection presented above, then the inclusion of those aspects certainly would be obvious over Decasper98 in view of RFC 1825, RFC 1829, RFC 1883, Huitema, Decasper97, Bellare97, Bellare95, IBM96, Nelson, Bellissard, and Fraser under 35 U.S.C. § 103, under Implicit's apparent claim constructions. All of these references have already been combined with Decasper98 in corresponding sections above, and those sections should be consulted for the detailed manner of applying them to Decasper98. This section briefly summarizes that material and shows the collective combination of these references would be obvious as well.

Decasper98 teaches a general architecture for router/firewall plugins and repeatedly emphasizes its "extensibility." Ex. 25 at 1, 2, 3, 11, 6 ("Doubtless, additional plugin types will be introduced by third parties once we have released our code into the public domain."). Decasper98 teaches "plugins for IP security," and Decasper97 confirms the obviousness of providing separate plugin components for encryption and authentication. IBM96 confirms the obviousness of an additional plugin component for compression. Decasper98 also teaches "plugins implementing IPv6 options." Id. at 4. Since Decasper98 teaches that its plugin components are selected on the basis of separate, independent filter tables, it was obvious that any two or more of these four types of plugins (encryption, authentication, compression, IPv6 options) might be applied to the same flow. This is especially obvious since the first three operations would be useful for implementing, e.g., a virtual private network across an expensive link, and IPv6 options are of general usefulness. See Ex. 25 (Decasper98) at 5 ("system

Art Unit: 3992

is configured as entry point into a virtual private network").

Claims 1, 15, and 35 recite elements regarding "state information."

RFC 1829 and Bellare97 confirm the obviousness of employing a stateful encryption algorithm which would read on these elements. Bellare95 confirms the obviousness of employing a stateful authentication algorithm which would read on these elements. Nelson confirms the obviousness of employing a stateful compression algorithm which would read on these elements. RFC 1883 confirms the obviousness of employing a stateful algorithm for implementing IPv6 options which would read on these elements.

RFC 1825 confirms the obviousness of inserting separate headers into a packet for both encryption and authentication, and this would read on this "converting data" element, under Implicit's apparent claim constructions. Performing compression on a packet would read on this "converting data" element as well, under Implicit's apparent claim constructions.

Huitema confirms the obviousness of adding or removing headers while processing IPv6 options, which would read on the "converting data" element as well, under Implicit's apparent claim constructions. In addition to the specific plugin Components discussed immediately above (encryption, authentication, compression, IPv6 options),

Decasper98 discloses a number of other plugin components which would read on the "state information" and/or "format" claim elements of claims 1, 15, and 35, including plugin components for statistics gathering, packet scheduling, and firewall

Art Unit: 3992

functions. See Sections V.C. 1 (Decasper98 102) and V.C.2 (Decasper98 103) above. Since Decasper98 teaches that its plugin components are selected on the basis of separate, independent filter tables, it was obvious for any of these various plugin components to be applied to the same flow as well, in addition to (or instead of) any of the encryption, authentication, compression, or IPv6 options components discussed immediately above.

Decasper98 selects the sequence of plugin components for a flow on the basis of multiple independent filters; which "even with very few installed filters" leads to "exponentially" many valid component sequences--so many, in fact, that it is "infeasible" to even list them in memory ahead of time. Ex. 25 at 7. Decasper98 therefore adopts an algorithmic approach, of dynamically generating the sequence when the first packet of a flow arrives, by applying its multiple independent filters to the packet data which did not exist in the system until the packet arrived. Under Implicit's apparent claim constructions, this technique alone reads on these "dynamic[]" claim elements. Moreover, Decasper98 also teaches that new plugin components may be added and configured by an administrator at runtime, "even when network traffic is transiting through the system"--including at least up to the very moment before a new flow would begin. Id. at 9. This also reads on these "dynamic[]" claim elements, under Implicit's apparent claim constructions.

Like Decasper98, Bellissard teaches dynamically adding new components while the system is operating. It provides additional detail on how such a system could operate, and on another way in which it could be implemented. Bellissard further teaches

Art Unit: 3992

the dynamic ' modification of existing components--again, while the system is operating. Under Implicit's apparent claim constructions, both of these techniques would read on these "dynamic[]" claim elements.

Like Decasper98, Fraser teaches dynamically configuring firewall policies while the system is operating. It teaches a more comprehensive framework for this capability, and details another manner in which it could be implemented. Under Implicit's apparent claim constructions, such dynamic configuration of policies would read on these "dynamic[]" claim elements.

However, Requester has not shown which particular limitations of claims 1, 15 and 35 are not taught by Decasper98, and as such why the rejection requires the steps of combining the teachings of RFC 1825, RFC 1829, RFC 1883, Huitema, Decasper97, Bellare97, Bellare95, IBM96, Nelson, Bellissard, and Fraser to Decasper98.

Hence, for the reasons cited above, it is found that the requester has not shown a reasonable likelihood of prevail with respect to claims 1, 15 and 35.

ISSUE 41

Claim 1 states:

...for the *first packet* of the message, *identifying* a sequence of components for processing the packets of the message such that the output format of the components of the sequence match the input format of the next component in the sequence; and *storing* an indication of each of the identified components so that the sequence does not need to be re-identified for subsequent packets of the message...

Art Unit: 3992

Similarly, claim 15 states:

...**identifying** a sequence of components for processing each message based on the **first packet** of the message so that subsequent packets of the message can be processed without re-identifying the components, ...

Similarly, claim 35 states:

...**identifying** a message-specific sequence of components for processing the packets of each message upon receiving the **first packet** of the message wherein subsequent packets of the message can use the message-specific sequence identified when the **first packet** was received...

In regards to these limitations relating to “**first packet**”, Requester submits in pages 250-253:

Mosberger discloses "dynamically identifying" under Implicit's apparent claim construction. Mosberger identifies two possible approaches to path creation: (a) "paths are pre-specified" or (b) "paths are created (discovered) incrementally." Ex. 31 at 39. Mosberger rejects the "pre-specifying paths" approach- i.e., a system that "provide[s] a table that translates the properties of the desired path into a sequence of modules that the path needs to traverse to satisfy these properties"- in favor of a system that "create[s] paths incrementally." Id. at 40. This is because "[i]n many cases it is beneficial to exploit information that is available at runtime only.

For this reason, paths need to be created and destroyed dynamically at runtime." Id. at 39. As Mosberger explains, "runtime covers all the steps that occur after the system has been booted on the target machine. During that time, paths may be created,

Art Unit: 3992

used, and destroyed." *Id.* at 61 ; see also *id.* Figure 3.1. Mosberger also makes clear that path changes can happen during runtime. For example, Mosberger explains that "a command-line interpreter is likely to create a path to the input device (e.g., the keyboard) during initialization." Ex. 31 at 47. "New paths" can then be created through the "handling of key-strokes." *Id.* Moreover, given Implicit's apparent claim constructions, Mosberger expressly illustrates that the Scout system can make "dynamic routing decision[s]": *id.* at 42. The components disclosed in Mosberger are also used in a manner "such that the output format of the components..., match the input format of the next component," under Implicit's apparent claim construction. See Section IV(C). In Mosberger's system, "a data-item arrives at the input queue, the path is scheduled for execution, and the transformed data is deposited in an output queue." Ex. 31 at 48. As explained previously, data may be processed by multiple components (or modules) in the course of moving through a path. See *id.* at 36; see also Figure 2.4. Because packets compatibly move from component to component, this element is satisfied under Implicit's apparent claim construction.

The "dynamically identifying" as disclosed in Mosberger (under Implicit's apparent claim construction) also "includes selecting individual components to create the non-predefined sequence of components after the first packet is received." In Mosberger's system, individual modules can "make a dynamic routing decision" to ensure that data is "processed appropriately." Ex. 31 at 41-42; see also Figure 2.5. This "dynamic routing decision" is "based on the contents of the data being communicated." *Id.* at 88. It is designed to be able "to exploit information that is available at runtime

Art Unit: 3992

only." Id. at 36. As Mosberger explains: path creation is initiated at the module that is to form one end of the path. This module uses the invariants to make a routing decision, that is, a decision as to which module a path with the specified invariants must traverse next. Path creation is then forwarded to that next module. This process repeats itself until either there is no next module (i.e., the edge of the module graph has been reached) or until a module is reached that, based on the specified invariants, cannot make a definite routing decision. As part of making a routing decision, a module is free to update the invariants since new invariants may become available in that module or old invariants may be invalid beyond that module. Id. at 40.iv. Mosberger discloses this element.

As explained above, paths form when modules make "dynamic routing decision[s]" that are "based on the contents of the data being communicated." Ex. 31 at 41-41, 88. "Stages" along the path "provide a place to store information that is path-specific, but private to the modules." Id. at 73; see also id. Figure 3.5: Once a path is formed, the "sequence of modules being traversed is known and fixed for the lifetime of a path." Id. at 54. To be known and fixed, the sequence of modules for any given path must be stored as claimed in the patent.

Examiner submits that Mosberger does not appear to **disclose** "first packet" initiating the "identifying" step of sequence of components. Nowhere in the Request, pertaining to Mosberger, it is stated that the first packet initiates "identifying" step of sequence of components and all other subsequent "retrieving" step of state information

Art Unit: 3992

relating to performing processing of previous packet; and the "storing" step of the state information, , as required by the claim limitations.

Therefore, Examiner submits that Mosberger does not appear to teach ...for the *first packet* of the message, *identifying* a sequence of components for processing the packets of the message... as in claim 1, ...*identifying* a sequence of components for processing each message based on the *first packet* of the message so that subsequent packets of the message can be processed without re-identifying the components, ... as in claim 15 and ...*identifying* a message-specific sequence of components for processing the packets of each message upon receiving the *first packet* of the message wherein subsequent packets of the message can use the message-specific sequence identified when the *first packet* was received... as in claim 35.

Since, Mosberger does not dynamically identify sequences of components based only on the first packet with using pre-defined fields, therefore, proposed rejection of claims 1, 15 and 35, does not show a reasonable likelihood that the requester will prevail with respect to at least one of the said claims of the patent. Hence, for the reasons cited above, it is found that the requester has not shown a reasonable likelihood of prevail with respect to claims 1, 15 and 35.

ISSUE 42

Claim 1 states:

...for the **first packet** of the message, **identifying** a sequence of components for processing the packets of the message such that the output format of the components of the sequence match the input format of the next component in the sequence; and **storing** an indication of each of the identified components so that the sequence does not need to be re-identified for subsequent packets of the message...

Similarly, claim 15 states:

...**identifying** a sequence of components for processing each message based on the **first packet** of the message so that subsequent packets of the message can be processed without re-identifying the components, ...

Similarly, claim 35 states:

...**identifying** a message-specific sequence of components for processing the packets of each message upon receiving the **first packet** of the message wherein subsequent packets of the message can use the message-specific sequence identified when the **first packet** was received...

In regards to these limitations relating to "**first packet**", Requester submits in pages 259-260:

Mosberger discloses all of the limitations of Claims 1, 15, and 35--including the "dynamically identifying" limitation--for the reasons set forth above. However, even if Mosberger is deemed not to have an express disclosure of the "dynamically identifying" limitation, one of ordinary skill in the art would have immediately appreciated that the system disclosed in Mosberger could have been modified without difficulty to include such functionality.

Art Unit: 3992

Specifically, during prior *ex parte* reexamination of the '163 patent, focus was placed on the passage of Mosberger at page 71 to the effect that "the Scout module graph is presently configured at build time and, hence, it is not possible to extend the graph at runtime." Ex. 31 at 71. However, Mosberger goes on to expressly state "However, it is straight-forward to add a dynamic module-loading facility to Scout." *Id.* Mosberger expresses further confidence in the ease with which such a "dynamic loading" functionality could be added to the disclosed Scout system, stating that the "actual dynamic loading" is not the "biggest issue" in modifying Scout, but rather "the security issue.;" *Id.* And, of course, the claims of the '163 patent contain no limitation directed to any such "security issue"; in other words, even an insecure implementation of "dynamic loading" would satisfy the "dynamically identifying" limitation of the '163 patent.

Furthermore, Mosberger proposes yet another modification of Scout that would permit "dynamically identifying," which is "to configure a virtual machine module into the graph that would allow interpreted code to be downloaded and executed inside Scout." *Id.* For example, Mosberger here drops a reference to footnote 39, which directs the reader to a reference entitled "The Java Application Programming Interface." Ex. 31 at 71,167. One of ordinary skill in the art would have appreciated that the Java programming environment can readily provide a "virtual machine" to be used to permit code to be dynamically "downloaded and executed inside Scout." *Id.*

Examiner submits that Mosberger does not appear to disclose "first packet" initiating the "identifying" step of sequence of components. Nowhere in the Request,

Art Unit: 3992

pertaining to Mosberger, it is stated that the first packet initiates "identifying" step of sequence of components and all other subsequent "retrieving" step of state information relating to performing processing of previous packet; and the "storing" step of the state information, , as required by the claim limitations.

Therefore, Examiner submits that Mosberger does not appear to disclose ...for the *first packet* of the message, *identifying* a sequence of components for processing the packets of the message... as in claim 1, ...*identifying* a sequence of components for processing each message based on the *first packet* of the message so that subsequent packets of the message can be processed without re-identifying the components, ... as in claim 15 and ...*identifying* a message-specific sequence of components for processing the packets of each message upon receiving the *first packet* of the message wherein subsequent packets of the message can use the message-specific sequence identified when the *first packet* was received... as in claim 35.

Since, Mosberger does not dynamically identify sequences of components based only on the first packet with using pre-defined fields, therefore, proposed rejection of claims 1, 15 and 35, as set forth in pages 258-260 of the Request, does not show a reasonable likelihood that the requester will prevail with respect to at least one of the said claims of the patent. Hence, for the reasons cited above, it is found that the requester has not shown a reasonable likelihood of prevail with respect to claims 1, 15 and 35.

ISSUE 43

Claim 1 states:

...for the **first packet** of the message, **identifying** a sequence of components for processing the packets of the message such that the output format of the components of the sequence match the input format of the next component in the sequence; and **storing** an indication of each of the identified components so that the sequence does not need to be re-identified for subsequent packets of the message...

Similarly, claim 15 states:

...**identifying** a sequence of components for processing each message based on the **first packet** of the message so that subsequent packets of the message can be processed without re-identifying the components, ...

Similarly, claim 35 states:

...**identifying** a message-specific sequence of components for processing the packets of each message upon receiving the **first packet** of the message wherein subsequent packets of the message can use the message-specific sequence identified when the **first packet** was received...

In regards to these limitations relating to "**first packet**", Requester submits in pages 262-264:

HotLava expressly describes itself as a "dynamic" system: "In our Java-based protocol architecture, special service classes dynamically construct protocol graphs at runtime as applications need communications services." Id. at 96; see also id. at Fig. 1.

HotLava explains that it is "natural to consider" the Java environment as a way of addressing the need for "flexible communication protocols and services to support them," as a way of solving the problem of "the number and variety of Web- and network-based applications [that] continue[] to increase." *Id.* at 93. Using the system disclosed in HotLava, "protocols and additional code required to support them can be downloaded and executed on the fly as needed." *Id.* see also *id.* at 96 ("This extensible architecture... allows on-the-fly introduction of new or replacement protocol code."). Thus, "new classes, such as those making up our protocol subsystem and protocol implementations, can be added dynamically" *Id.* at 95. Among other things, the HotLava approach overcomes shortcomings of certain "traditional" approaches and systems, which had to be "completely recompiled and redeployed" in order to accommodate change: *Id.* Not only is the protocol graph of software modules ("sequence of protocols") determined "dynamically... at runtime," each also receives a separate instantiation in memory; thus, "multiple instances of the same protocol can be executing simultaneously." *Id.* at 98. "For example, an application needing AppleTalk services need only create an instance of its corresponding service class." *Id.* at 96.

As explained earlier, Mosberger proposed configuring "a virtual machine module into the graph that would allow interpreted code to be downloaded and executed inside Scout." *Ex.* 31 at 71. As shown above, HotLava expressly provides "the ability to incorporate new protocol classes... into the virtual machine." *Ex.* 32 at 96. Thus, under the HotLava approach, "protocols and additional code required to support them can be downloaded and executed on the fly as needed." *Id.* at 93. Incorporating into Scout the

Art Unit: 3992

HotLava approach--a Java-based solution as expressly proposed in Mosberger--thus clearly satisfies any perceived shortcoming of Mosberger with respect to the "dynamically identifying" limitation.

Examiner submits that Mosberger in view of HotLava does not appear to disclose "first packet" initiating the "identifying" step of sequence of components. Nowhere within the above cited pages of the Request it is stated that the first packet initiates "identifying" step of sequence of components and all other subsequent "retrieving" step of state information relating to performing processing of previous packet; and the "storing" step of the state information, , as required by the claim limitations.

Therefore, Examiner submits that Mosberger in view of HotLava does not appear to disclose ... for the first packet of the message, identifying a sequence of components for processing the packets of the message... as in claim 1, ...identifying a sequence of components for processing each message based on the first packet of the message so that subsequent packets of the message can be processed without re-identifying the components, ... as in claim 15 and ...identifying a message-specific sequence of components for processing the packets of each message upon receiving the first packet of the message wherein subsequent packets of the message can use the message-specific sequence identified when the first packet was received... as in claim 35.

Since, Mosberger in view of HotLava does not dynamically identify sequences of components based only on the first packet with using pre-defined

Art Unit: 3992

fields, therefore, proposed rejection of claims 1, 15 and 35, as set forth in pages 263-268 of the Request, does not show a reasonable likelihood that the requester will prevail with respect to at least one of the said claims of the patent. Hence, for the reasons cited above, it is found that the requester has not shown a reasonable likelihood of prevail with respect to claims 1, 15 and 35.

ISSUE 44

Claim 1 states:

...for the *first packet* of the message, *identifying* a sequence of components for processing the packets of the message such that the output format of the components of the sequence match the input format of the next component in the sequence; and *storing* an indication of each of the identified components so that the sequence does not need to be re-identified for subsequent packets of the message...

Similarly, claim 15 states:

...*identifying* a sequence of components for processing each message based on the *first packet* of the message so that subsequent packets of the message can be processed without re-identifying the components, ...

Similarly, claim 35 states:

...*identifying* a message-specific sequence of components for processing the packets of each message upon receiving the *first packet* of the message wherein subsequent packets of the message can use the message-specific sequence identified when the *first packet* was received...

Art Unit: 3992

In regards to these limitations relating to “**first packet**”, Requester submits in page 269:

The HotLava reference not only renders the claims of the '163 patent when considered in combination with Mosberger, but HotLava also independently and standing alone discloses each and every element of claims 1, 15, and 35. Accordingly, HotLava also fully anticipates these claims for the reasons Set forth in detail above, which are incorporated by reference in this proposed ground of rejection.

Examiner submits that HotLava does not appear to **disclose** “first packet” initiating the “identifying” step of sequence of components. Nowhere within the above cited pages of the Request it is stated that the first packet initiates “identifying” step of sequence of components and all other subsequent “retrieving” step of state information relating to performing processing of previous packet; and the “storing” step of the state information, , as required by the claim limitations.

Therefore, Examiner submits that HotLava does not appear to disclose ...for the **first packet** of the message, **identifying** a sequence of components for processing the packets of the message... as in claim 1, ...**identifying** a sequence of components for processing each message based on the **first packet** of the message so that subsequent packets of the message can be processed without re-identifying the components, ... as in claim 15 and ...**identifying** a message-specific sequence of components for processing the packets of each message upon receiving the **first packet** of the message wherein subsequent packets of the message can use the

Art Unit: 3992

message-specific sequence identified when the *first packet* was received... as in claim 35.

Since, HotLava does not dynamically identify sequences of components based only on the first packet with using pre-defined fields, therefore, proposed rejection of claims 1, 15 and 35, as set forth in pages 263-268 of the Request, does not show a reasonable likelihood that the requester will prevail with respect to at least one of the said claims of the patent. Hence, for the reasons cited above, it is found that the requester has not shown a reasonable likelihood of prevail with respect to claims 1, 15 and 35.

ISSUE 45

Claim 1 states:

...for the *first packet* of the message, *identifying* a sequence of components for processing the packets of the message such that the output format of the components of the sequence match the input format of the next component in the sequence; and *storing* an indication of each of the identified components so that the sequence does not need to be re-identified for subsequent packets of the message...

Similarly, claim 15 states:

...*identifying* a sequence of components for processing each message based on the *first packet* of the message so that subsequent packets of the message can be processed without re-identifying the components, ...

Similarly, claim 35 states:

...*identifying* a message-specific sequence of components for processing the packets of each message upon receiving the *first packet* of the message wherein subsequent packets of the message can use the message-specific sequence identified when the *first packet* was received...

In regards to these limitations relating to "**first packet**", Requester submits in pages 269-270:

If certain aspects recited in claims 1, 15, and 35 of the ' 163 patent are not deemed to be disclosed, inherent, or obvious over Mosberger, then the inclusion of those aspects certainly would be obvious over Mosberger in view of Plexus, under 35 U.S.C. § 103.

It was obvious to supplement the teachings of Mosberger with Plexus because Mosberger expressly states it is "straight-forward to add a dynamic module-loading facility to Scout" (Ex. 31 at 71), and a "key aspect of Plexus is... [a] protocol graph that can be dynamically changed as applications come and go." Ex. 33 at 55. Plexus does so in a way that "does not compromise the safety of other applications or the operating system." *Id.* at 55.

"Plexus allows applications to define new protocols or to change the implementation of existing protocols." *Id.* Indeed, Plexus even "supports multiple implementations of the same protocol for different endpoints." *Id.* at 58. The Plexus system is also "dynamic" under Implicit's apparent claim construction because it permits "[r]untime adaptation." *Id.* at 56. Specifically, "[a]pplications may add extensions to the kernel at any point during the system's execution without requiring superuser privileges

Art Unit: 3992

or a system reboot." Id.; see also id. ("Plexus allows extensions to be safely loaded and unloaded into a running system").

Thus, to the extent that Mosberger is deemed to lack inadequate disclosure of the "dynamically identifying" limitation for claims 1, 15, and 35, the combination of Mosberger with Plexus clearly makes up for any such perceived deficiency.

Examiner submits that Mosberger in view of Plexus does not appear to **disclose** "first packet" initiating the "identifying" step of sequence of components. Nowhere within the above cited pages of the Request it is stated that the first packet initiates "identifying" step of sequence of components and all other subsequent "retrieving" step of state information relating to performing processing of previous packet; and the "storing" step of the state information, , as required by the claim limitations.

Therefore, Examiner submits that Mosberger in view of Plexus does not appear to disclose ...for the **first packet** of the message, **identifying** a sequence of components for processing the packets of the message... as in claim 1, ...**identifying** a sequence of components for processing each message based on the **first packet** of the message so that subsequent packets of the message can be processed without re-identifying the components, ... as in claim 15 and ...**identifying** a message-specific sequence of components for processing the packets of each message upon receiving the **first packet** of the message wherein subsequent packets of the message can use the message-specific sequence identified when the **first packet** was received... as in claim 35.

Art Unit: 3992

Furthermore, Plexus Fails to overcome the deficiencies of Mosberger, ...for the first packet of the message, identifying a sequence of components for processing the packets of the message... as in claim 1, ...identifying a sequence of components for processing each message based on the first packet of the message so that subsequent packets of the message can be processed without re-identifying the components, ... as in claim 15 and ...identifying a message-specific sequence of components for processing the packets of each message upon receiving the first packet of the message wherein subsequent packets of the message can use the message-specific sequence identified when the first packet was received... as in claim 35.

Since, Mosberger in view of Plexus does not dynamically identify sequences of components based only on the first packet with using pre-defined fields, therefore, proposed rejection of claims 1, 15 and 35, as set forth in pages 269-270 of the Request, does not show a reasonable likelihood that the requester will prevail with respect to at least one of the said claims of the patent. Hence, for the reasons cited above, it is found that the requester has not shown a reasonable likelihood of prevail with respect to claims 1, 15 and 35.

ISSUE 46

Claim 1 states:

...for the first packet of the message, identifying a sequence of components for processing the packets of the message such that the output format of the components

Art Unit: 3992

of the sequence match the input format of the next component in the sequence; and **storing** an indication of each of the identified components so that the sequence does not need to be re-identified for subsequent packets of the message...

Similarly, claim 15 states:

...**identifying** a sequence of components for processing each message based on the **first packet** of the message so that subsequent packets of the message can be processed without re-identifying the components, ...

Similarly, claim 35 states:

...**identifying** a message-specific sequence of components for processing the packets of each message upon receiving the **first packet** of the message wherein subsequent packets of the message can use the message-specific sequence identified when the **first packet** was received...

In regards to these limitations relating to "**first packet**", Requester submits in pages 271-272:

If certain aspects recited in claims 1, 15, and 35 of the '163 patent are not deemed to be disclosed, inherent, or obvious over Mosberger, then the inclusion of those aspects certainly would be obvious over Mosberger in view of ComScript, under 35 U.S.C. § 103.

It was obvious to Supplement the teachings of Mosberger with ComScript because Mosberger expressly states it is "straight-forward to add a dynamic module-loading facility to Scout" (Ex. 31 at 71), and Plexus expressly proposes an approach that "brings more flexibility by allowing an application to dynamically (re)configure an entire

Art Unit: 3992

protocol stack " Ex. 34 at 1.

ComScript provides, as an illustration, the following example of how the disclosed system can be used to create a new protocol stack or sequence of "modules" on the fly for purposes of a given session between hosts A and B:

An application running on host A establishes a communication with a COMSCRIPT server (CS) on the remote machine B by opening two connections, one for control information and the other for data exchange. The control connection is used by the application to send requests to the remote server. The application then downloads its own code to host B using the control channel. The execution of this code in the remote host results in the creation of a protocol stack which can then be used by the application to exchange data with host B. *Id.* at 6-7; Fig. 10; see also Figs. 7-9 (illustrating how to add or remove a "module" from a stack; "the number of configurable entities is unlimited").

The ComScript system is also "dynamic" under Implicit's apparent claim construction because it expressly states that one its "primary goal[s]" is to "make protocol stacks truly configurable at run time." *Id.* at 8.

Thus, to the extent that Mosberger is deemed to lack inadequate disclosure of the "dynamically identifying" limitation for claims 1, 15, and 35, the combination of Mosberger with ComScript clearly makes up for any such perceived deficiency.

Examiner submits that Mosberger in view of ComScript does not appear to **disclose** "first packet" initiating the "identifying" step of sequence of components. Nowhere within the above cited pages of the Request it is stated that the first packet

Art Unit: 3992

initiates "identifying" step of sequence of components and all other subsequent "retrieving" step of state information relating to performing processing of previous packet; and the "storing" step of the state information, , as required by the claim limitations.

Therefore, Examiner submits that Mosberger in view of ComScript does not appear to disclose ...for the **first packet** of the message, **identifying** a sequence of components for processing the packets of the message... as in claim 1, ...**identifying** a sequence of components for processing each message based on the **first packet** of the message so that subsequent packets of the message can be processed without re-identifying the components, ... as in claim 15 and ...**identifying** a message-specific sequence of components for processing the packets of each message upon receiving the **first packet** of the message wherein subsequent packets of the message can use the message-specific sequence identified when the **first packet** was received... as in claim 35.

Furthermore, ComScript Fails to overcome the deficiencies of Mosberger, ...for the **first packet** of the message, **identifying** a sequence of components for processing the packets of the message... as in claim 1, ...**identifying** a sequence of components for processing each message based on the **first packet** of the message so that subsequent packets of the message can be processed without re-identifying the components, ... as in claim 15 and ...**identifying** a message-specific sequence of components for processing the packets of each message upon receiving the **first packet** of the message wherein subsequent packets of the message can use the

Art Unit: 3992

message-specific sequence identified when the *first packet* was received... as in claim 35.

Since, Mosberger in view of ComScript does not dynamically identify sequences of components based only on the first packet with using pre-defined fields, therefore, proposed rejection of claims 1, 15 and 35, as set forth in pages 270-271 of the Request, does not show a reasonable likelihood that the requester will prevail with respect to at least one of the said claims of the patent. Hence, for the reasons cited above, it is found that the requester has not shown a reasonable likelihood of prevail with respect to claims 1, 15 and 35.

Scope of Reexamination

4. Claims 1, 15 and 35 will be reexamined as requested in the request.

Conclusion

5. Extensions of time under 37 CFR 1.136(a) will not be permitted in *inter partes* reexamination proceedings because the provisions of 37 CFR 1.136 apply only to "an applicant" and not to the patent owner in a reexamination proceeding. Additionally, 35 U.S.C. 314(c) requires that inter partes reexamination proceedings "will be conducted with special dispatch" (37 CFR 1.937). Patent owner extensions of time in inter partes reexamination proceedings are provided for in 37 CFR 1.956. Extensions of time are not available for third party requester comments, because a comment period of 30 days from service of patent owner's response is set by statute. 35 U.S.C. 314(b)(3).

Art Unit: 3992

6. The Patent Owner is reminded of the continuing responsibility under 37 CFR 1.985(a) to apprise the Office of any litigation activity, or other prior or concurrent proceeding, involving the US Patent 6,629,163 throughout the course of this reexamination proceeding. The Third Party Requester is also reminded of the ability to similarly apprise the Office of any such activity or proceeding through the course of this reexamination proceeding. See MPEP § 2686 and 2686.04.

All correspondence relating to this *inter partes* reexamination proceeding should be directed:

By EFS: Registered users may submit via the electronic filing system EFS-Web, at <https://efs.uspto.gov/efile/myportal/efs-registered>

By Mail to: Mail Stop *Inter Partes* Reexam
Attn: Central Reexamination Unit
Commissioner for Patents
United States Patent & Trademark Office
P.O. Box 1450
Alexandria, Virginia 22313-1450

By FAX to: (571) 273-9900
Central Reexamination Unit

By hand: Customer Service Window
Attn: Central Reexamination Unit
Randolph Building, Lobby Level
401 Dulany Street
Alexandria, VA 22314

For EFS-Web transmissions, 37 CFR 1.8(a)(1)(i) (C) and (ii) states that correspondence (except for a request for reexamination and a corrected or replacement request for reexamination) will be considered timely filed if (a) it is transmitted via the Office's electronic filing system in accordance with 37 CFR 1.6(a)(4), and (b) includes a certificate of transmission for each piece of correspondence stating the data of transmission, which is prior to the expiration of the set period of time in the Office action.

Art Unit: 3992

Any inquiry concerning this communication or earlier communications from the examiner, or as to the status of this proceeding, should be directed to the Central Reexamination Unit at telephone number (571) 272-7705.

/Salman Ahmed/
Salman Ahmed
Primary Examiner
Central Reexamination Unit - Art Unit 3992
(571) 272-8307

Conferee:

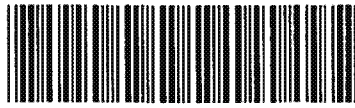
Conferee:

/Ovidio Escalante/

/Daniel J Ryman/

Supervisory Patent Examiner,

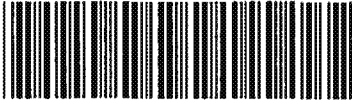
Art Unit 3992

Index of Claims 	Application/Control No. 95000659	Applicant(s)/Patent Under Reexamination 6629163
	Examiner SALMAN AHMED	Art Unit 3992

✓	Rejected	-	Cancelled	N	Non-Elected	A	Appeal
=	Allowed	÷	Restricted	I	Interference	O	Objected

Claims renumbered in the same order as presented by applicant
 CPA
 T.D.
 R.1.47

CLAIM		DATE									
Final	Original	03/28/2012									
	1	✓									
	2										
	3										
	4										
	5										
	6										
	7										
	8										
	9										
	10										
	11										
	12										
	13										
	14										
	15	✓									
	16										
	17										
	18										
	19										
	20										
	21										
	22										
	23										
	24										
	25										
	26										
	27										
	28										
	29										
	30										
	31										
	32										
	33										
	34										
	35	✓									

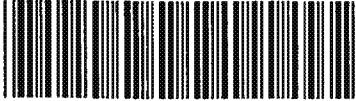
Reexamination 	Application/Control No. 95000659	Applicant(s)/Patent Under Reexamination 6629163
	Certificate Date	Certificate Number

Requester Correspondence Address: Patent Owner Third Party

Irell and Manella, LLP
830 Newport Center Dr. , Ste 400
Newport Beach CA 92660

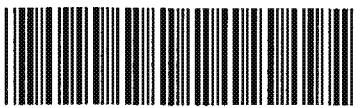
LITIGATION REVIEW <input checked="" type="checkbox"/>	ISA/ (examiner initials)	03/28/2012 (date)
Case Name	Director Initials	
3:10cv4234 (OPEN) Implicit Networks, Inc v. Juniper Networks,	DJR G SY	
3:10cv3766 (CLOSED) Implicit Networks, Inc v. Citrix Systems,	DJR G SY	
3:10cv3746 (OPEN) Implicit Networks, Inc v. Hewlett-Packard	DJR G SY	
5:10cv3606 (OPEN) Implicit Networks, Inc v. Cisco Systems,	DJR G SY	
3:10cv3606 (CLOSED) Implicit Networks, Inc v. Cisco Systems	DJR G SY	
3:10cv3365 (OPEN) Implicit Networks, Inc v. F5 Networks	DJR G SY	
3:09cv5628 (CLOSED) Implicit Networks, Inc v. Microsoft	DJR G SY	
2:08cv184(CLOSED) Implicit Networks, Inc v. A M Devices	DJR G SY	

COPENDING OFFICE PROCEEDINGS	
TYPE OF PROCEEDING	NUMBER

Reexamination 	Application/Control No. 95000659	Applicant(s)/Patent Under Reexamination 6629163
	Certificate Date	Certificate Number

PENDING OFFICE PROCEEDINGS	
TYPE OF PROCEEDING	NUMBER

--	--

Search Notes 	Application/Control No. 95000659	Applicant(s)/Patent Under Reexamination 6629163
	Examiner SALMAN AHMED	Art Unit 3992

SEARCHED			
Class	Subclass	Date	Examiner
None	None		

SEARCH NOTES		
Search Notes	Date	Examiner
File history	3/28/2012	SA
Patent prosecution history	3/28/2012	SA

INTERFERENCE SEARCH			
Class	Subclass	Date	Examiner

--	--



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
95/000,659	02/13/2012	6629163	159291-0025(163)	6219
55959	7590	04/03/2012	EXAMINER	
Newman Du Wors LLP 1201 Third Avenue, Suite 1600 SEATTLE, WA 98101			AHMED, SALMAN	
			ART UNIT	PAPER NUMBER
			3992	
			MAIL DATE	DELIVERY MODE
			04/03/2012	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.



DO NOT USE IN PALM PRINTER

THIRD PARTY REQUESTER'S CORRESPONDENCE ADDRESS
IRELL & MANELLA, LLP
DAVID MCPHIE
840 NEWPORT CENTER DR., STE 400
NEWPORT BEACH, CA 92660

Date: 4-3-12

**Transmittal of Communication to Third Party Requester
Inter Partes Reexamination**

REEXAMINATION CONTROL NO. : 95000659
PATENT NO. : 6629163
TECHNOLOGY CENTER : 3999
ART UNIT : 3992

Enclosed is a copy of the latest communication from the United States Patent and Trademark Office in the above identified Reexamination proceeding. 37 CFR 1.903.

Prior to the filing of a Notice of Appeal, each time the patent owner responds to this communication, the third party requester of the inter partes reexamination may once file written comments within a period of 30 days from the date of service of the patent owner's response. This 30-day time period is statutory (35 U.S.C. 314(b)(2)), and, as such, it cannot be extended. See also 37 CFR 1.947.

If an ex parte reexamination has been merged with the inter partes reexamination, no responsive submission by any ex parte third party requester is permitted.

All correspondence relating to this inter partes reexamination proceeding should be directed to the Central Reexamination Unit at the mail, FAX, or hand-carry addresses given at the end of the communication enclosed with this transmittal.

PTOL-2070(Rev.07-04)

OFFICE ACTION IN INTER PARTES REEXAMINATION	Control No.	Patent Under Reexamination
	95/000,659	6629163
	Examiner	Art Unit
	SALMAN AHMED	3992

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address. --

Responsive to the communication(s) filed by:

Patent Owner on _____

Third Party(ies) on 13 February, 2012

RESPONSE TIMES ARE SET TO EXPIRE AS FOLLOWS:

For Patent Owner's Response:

2 MONTH(S) from the mailing date of this action. 37 CFR 1.945. EXTENSIONS OF TIME ARE GOVERNED BY 37 CFR 1.956.

For Third Party Requester's Comments on the Patent Owner Response:

30 DAYS from the date of service of any patent owner's response. 37 CFR 1.947. NO EXTENSIONS OF TIME ARE PERMITTED. 35 U.S.C. 314(b)(2).

All correspondence relating to this inter partes reexamination proceeding should be directed to the **Central Reexamination Unit** at the mail, FAX, or hand-carry addresses given at the end of this Office action.

This action is not an Action Closing Prosecution under 37 CFR 1.949, nor is it a Right of Appeal Notice under 37 CFR 1.953.

PART I. THE FOLLOWING ATTACHMENT(S) ARE PART OF THIS ACTION:

1. Notice of References Cited by Examiner, PTO-892
2. Information Disclosure Citation, PTO/SB/08
3. _____

PART II. SUMMARY OF ACTION:

- 1a. Claims 1,15 and 35 are subject to reexamination.
- 1b. Claims _____ are not subject to reexamination.
2. Claims _____ have been canceled.
3. Claims _____ are confirmed. [Unamended patent claims]
4. Claims _____ are patentable. [Amended or new claims]
5. Claims 1,15 and 35 are rejected.
6. Claims _____ are objected to.
7. The drawings filed on _____ are acceptable are not acceptable.
8. The drawing correction request filed on _____ is: approved. disapproved.
9. Acknowledgment is made of the claim for priority under 35 U.S.C. 119 (a)-(d). The certified copy has:
 - been received. not been received. been filed in Application/Control No 95000659.
10. Other _____

DETAILED ACTION

1. A reasonable likelihood that the requestor will prevail with respect to at least one of the patent claims affecting the patentability of claims 1, 15 and 35 of United States Patent Number 6,629,163 (Balassanlan, Edward) is raised by the present Request for *inter partes* reexamination filed on 02/13/2012 (hereinafter the "Request").

Status of the Claims

2. Original claims 1, 15 and 35 are rejected.

Claim Rejections - 35 USC § 102

3. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

A person shall be entitled to a patent unless –

(e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

Claim Rejections - 35 USC § 103

4. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

Art Unit: 3992

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

5. The factual inquiries set forth in *Graham v. John Deere Co.*, 383 U.S. 1, 148

USPQ 459 (1966), that are applied for establishing a background for determining

obviousness under 35 U.S.C. 103(a) are summarized as follows:

1. Determining the scope and contents of the prior art.
2. Ascertaining the differences between the prior art and the claims at issue.
3. Resolving the level of ordinary skill in the pertinent art.
4. Considering objective evidence present in the application indicating obviousness or nonobviousness.

6. Claim 1 is rejected under 35 U.S.C. 103(a) as being unpatentable over Kerr et al. (US PAT 6243667, hereinafter Kerr).

In regards to claim 1, Kerr discloses ***a method in a computer system*** (column 2, lines 30-32, However, those skilled in the art would recognize, after perusal of this application, that embodiments of the invention may be implemented using a set of general purpose computers operating under program control, and that modification of a set of general purpose computers to implement the process steps and data structures described herein would not require undue invention) ***for processing a message having a sequence of packets*** (column 1 lines 59-60, The invention provides a method and system for switching in networks responsive to message flow patterns. A message "flow" is defined to comprise a set of packets to be transmitted between a particular source and a particular destination. When routers in a network identify a new

Art Unit: 3992

message flow, they determine the proper processing for packets in that message flow and cache that information for that message flow. Thereafter, when routers in a network identify a packet which is part of that message flow, they process that packet according to the proper processing for packets in that message flow. The proper processing may include a determination of a destination port for routing those packets and a determination of whether access control permits routing those packets to their indicated destination) ***the method comprising:***

providing a plurality of components, each component being a software routine for converting data with an input format into data with an output format
(Kerr discloses a "plurality of components" for processing messages. For example, claim 1 of Kerr describes using a "plurality of devices" to apply "policy treatments" to a "plurality of messages," where policy treatments are used to perform "access control,security," "queuing," "accounting," "traffic profiling," etc. Id. at 10:27-40. Processing components can include "treatment with regard to switching," "access control," and "encryption." Id. at 4:20-34. "[S]pecial processing" can include "authentication" techniques "useful for implementing security 'firewalls.'" Id. at 35-46. Kerr further discloses that a "rewrite function" may be invoked "to alter the header for the packet." Id. at 4:55-62. These components can be used for "converting data with an input format into data with an output format," under Implicit's apparent claim constructions, for example (as described above), the "encryption" and "rewrite" components to "alter" data to be processed. Id. at 4:30- 31, 4:55-62. The processing components of Kerr comprise "software routine" embodiments, as Kerr states that the processing instrumentality "may

Art Unit: 3992

include specific hardware constructed or programmed performing the process steps described herein" or "a general purpose processor operating under program control." Id. at 2:51-55; see also id. at Figs. 3-4 (illustrating software data structures));

for the first packet of the message, dynamically identifying a non-predefined sequence of components for processing the packets of the message wherein dynamically identifying includes selecting individual components to create the non-predefined sequence of components after the first packet is received (claim 1, lines 31-40, column 4 lines 20-34, column 3 line 38-column 6 line 27 identifying a first one message of a first plurality of messages associated with an application layer, said first plurality of messages having at least one policy treatment in common, said first plurality of messages being identified in response to an address of a selected source device and an address of a selected destination device, wherein said policy treatment comprises at least one of the access control information, security information, queuing information, accounting information, traffic profiling information, and policy information; In a preferred embodiment, the proper treatment of packets 150 in the message flow 160 includes treatment with regard to switching (thus, the routing device 140 determines an output port for switching packets 150 in the message flow 160), with regard to access control (thus, the routing device 140 determines whether packets 150 in the message flow 160 meet the requirements of access control, as defined by access control lists in force at the routing device 140), with regard to accounting (thus, the routing device 140 creates an accounting record for the message flow 160), with regard to encryption (thus, the routing device 140 determines encryption

Art Unit: 3992

treatment for packets 150 in the message flow 160), and any special treatment for packets 150 in the message flow 160. FIG. 2 shows a method for routing in networks responsive to message flow patterns. In broad overview, the method for routing in networks responsive to message flow patterns comprises two parts. In a first part, the routing device 140 builds and uses a flow cache described in further detail with regard to FIG. 3), in which routing information to be used for packets 150 in each particular message flow 160 is recorded and from which such routing information is retrieved for use...A method 200 for routing in networks responsive to message flow patterns is performed by the routing device 140. At a flow point 210, the routing device 140 is disposed for building and using the flow cache. At a step 221, the routing device 140 receives a packet 150. At a step 222, the routing device 140 identifies a message flow 160 for the packet 150. In a preferred embodiment, the routing device 140 examines a header for the packet 150 and identifies the IP address for the source device 120, the IP address for the destination device 130, and the protocol type for the packet 150. The routing device 140 determines the port number for the source device 120 and the port number for the destination device 130 responsive to the protocol type. Responsive to this set of information, the routing device 140 determines a flow key 310 (described with reference to FIG. 3) for the message flow 160. At a step 223, the routing device 140 performs a lookup in a flow cache for the identified message flow 160. If the lookup is unsuccessful, the identified message flow 160 is a "new" message flow 160, and the routing device 140 continues with the step 224. If the lookup is successful, the identified message flow 160 is an "old" message flow 160, and the routing device 140

Art Unit: 3992

continues with the step 225. In a preferred embodiment, the routing device 140 determines a hash table key responsive to the flow key 310. This aspect of the step 223 is described in further detail with regard to FIG. 3. At a step 224, the routing device 140 builds a new entry in the flow cache. The routing device 140 determines proper treatment of packets 150 in the message flow 160 and enters information regarding such proper treatment in a data structure pointed to by the new entry in the flow cache. In a preferred embodiment, the routing device 140 determines the proper treatment by performing a lookup in an IP address cache as shown in FIG. 4. In a preferred embodiment, the proper treatment of packets 150 in the message flow 160 includes treatment with regard to switching (thus, the routing device 140 determines an output port for switching packets 150 in the message flow 160), with regard to access control (thus, the routing device 140 determines whether packets 150 in the message flow 160 meet the requirements of access control, as defined by access control lists in force at the routing device 140), with regard to accounting (thus, the routing device 140 creates an accounting record for the message flow 160), with regard to encryption (thus, the routing device 140 determines encryption treatment for packets 150 in the message flow 160), and any special treatment for packets 150 in the message flow 160. In a preferred embodiment, the routing device 140 performs any special processing for new message flows 160 at this time... Thereafter, the routing device 140 proceeds with the step 225, using the information from the new entry in the flow cache, just as if the identified message flow 160 were an "old" message flow 160 and the lookup in a flow cache had been successful. At a step 225, the routing device 140 retrieves routing

Art Unit: 3992

information from the entry in the flow cache for the identified message flow 160. In a preferred embodiment, the entry in the flow cache includes a pointer to a rewrite function for at least part of a header for the packet 150. If this pointer is non-null, the routing device 140 invokes the rewrite function to alter the header for the packet 150. At a step 226, the routing device 140 routes the packet 150 responsive to the routing information retrieved at the step 225. Thus, in a preferred embodiment, the routing device 140 does not separately determine, for each packet 150 in the message flow 160, the information stored in the entry in the flow cache. Rather, when routing a packet 150 in the message flow 160, the routing device 140 reads the information from the entry in the flow cache and treats the packet 150 according to the information in the entry in the flow cache. Thus, in a preferred embodiment, the routing device 140 routes the packet 150 to an output port, determines whether access is allowed for the packet 150, determines encryption treatment for the packet 150, and performs any special treatment for the packet 150, all responsive to information in the entry in the flow cache. In a preferred embodiment, the routing device 140 also enters accounting information in the entry in the flow cache for the packet 150. When routing each packet 150 in the message flow 160, the routing device 140 records the cumulative number of packets 150 and the cumulative number of bytes for the message flow 160. Because the routing device 140 processes each packet 150 in the message flow 160 responsive to the entry for the message flow 160 in the flow cache, the routing device 140 is able to implement administrative policies which are designated for each message flow 160 rather than for each packet 150);

and storing an indication of each of the identified components so that the non-predefined sequence does not need to be re-identified for subsequent packets of the message; and for each of a plurality of packets of the message in sequence, for each of a plurality of components in the identified non-predefined sequence, retrieving state information relating to performing the processing of the component with the previous packet of the message; performing the processing of the identified component with the packet and the retrieved state information; and storing state information relating to the processing of the component with the packet for use when processing the next packet of the message (After receiving the first packet of a new flow, Kerr builds a new flow entry that is cached in memory, which constitutes "storing". Kerr also explains that building and caching a flow entry upon receiving the first new packet in a flow is specifically performed so that information "does not need to be re-identified for subsequent packets of the message," as that term is apparently construed by Implicit. Kerr explains that, for the sake of efficiency: information about message flow patterns is used to identify packets for which processing has already been determined, and therefore to process those packets without having to re-determine the same processing Thus, in a preferred embodiment, the routing device 140 does not separately determine, for each packet 150 in the message flow 160, the information stored in the entry in the flow cache. Rather, when routing a packet 150 in the message flow 160, the routing device 140 reads the information from the entry in the flow cache and treats the packet 150 according to the information in the entry in the flow cache. Ex. 15 at 1:33-36, 4:64-5:4.

Art Unit: 3992

In other words, when the first packet of a flow arrives, Kerr goes through the somewhat expensive and elaborate process of determining how all the packets of that flow should be treated: e.g., whether they should be encrypted, whether they should be modified or partially re-written, and where they should be routed next. *Id.* at 1:33-35, 4:13-60. It then records all this information about the proper processing for a flow by "build[ing] a new entry in the flow cache" for the flow, so the proper processing does not have to be wastefully and redundantly determined again for subsequent packets of the flow. *Id.* at 4:12-13. Kerr discloses this "state information" element. Implicit has taken a broad view of the "state information" limitations, arguing that they cover the retrieval, use, and storage of the identified sequence of components (e.g., a flow record) after the first packet is received. As demonstrated above (for the "storing an indication" element), Kerr retrieves, uses, and stores flow records in this manner to facilitate processing of packets in the same message after the first packet is received and a flow entry built. Kerr also discloses the retrieval, use, and storage of state information on a component-by-component basis. For example, in one embodiment of Kerr, there are components for access control, encryption, "special treatment," accounting, rewrite, among others. Ex. 15 at 5:5-25. The processing by these components is "all responsive to information in the entry in the flow cache." *Id.* at 5:9-10. As a specific example, an accounting component can maintain state information, such as "time stamp" data, "a cumulative count for the number of packets," and "a cumulative count for the number of bytes." *Id.* at 6:58-63. Kerr later uses timing information to identify expired or otherwise invalid flows (among other reasons). *Id.* at 5:52 - 6:19. As another example, Kerr can retrieve

Art Unit: 3992

the latest "usage information regarding relative use of network resources" in order to appropriately prioritize traffic using the relevant component. *Id.* at 5:41-49).

Kerr does not explicitly teach ***processing the packets of the message such that the output format of the components match the input format of the next component.***

Regarding the limitation "such that the output format of the components ... match the input format of the next component," it was well-known to those of ordinary skill in the art that certain operations on a packet must be performed in a certain order: e.g., if a packet is first converted into an encrypted format by a first component, a subsequent component would be unable to, e.g., rewrite its headers (because it was expecting to receive the packet in an unencrypted format). See *id.* at 4:31-32 ("encryption treatment for packets..., in the message flow"), 4:57-58 ("rewrite function for..., a header for the packet"). Thus, it was certainly at least obvious for one of ordinary skill in the art to arrange the sequence of components in a compatible manner, such that the output format of one matches the input format of the next-- rather than arranging them in an incompatible manner whereby various component(s) would be unable to perform their function(s).

7. Claim 1 is rejected under 35 U.S.C. 103(a) as being unpatentable over Decasper98.

In regards to claim 1, Decasper98 teaches ***a method in a computer system***

Art Unit: 3992

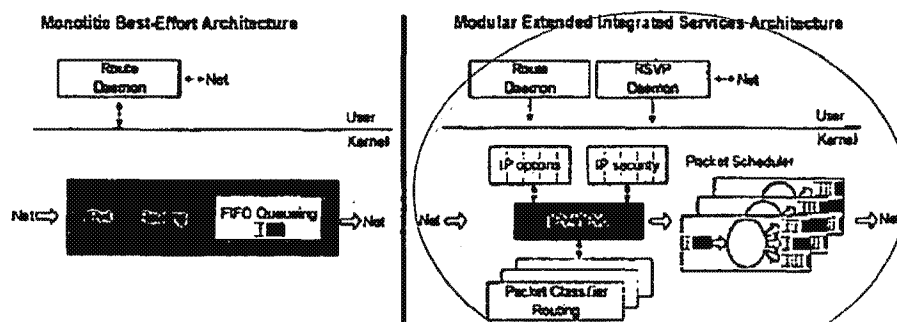


Figure 1. : Best Effort vs
Extended Integrated Services Router (EISR)

for processing a message having a sequence of packets (Decasper98 explains: "it is very important to be able to quickly and efficiently classify packets into flows, and to apply different policies to different flows; these are both things that our architecture excels at doing." Ex. 25 at 2. Flows may represent "longer lived packet streams": Because the deployment of multimedia data sources and applications (e.g. real-time audio/video) will produce longer lived packet streams with more packets per session than is common in today's environment, an integrated services router architecture should support the notion of flows and build upon it. Id. at 3. A flow is defined as a group of packets which satisfy a specific filter. See id. at 3 ("Sets of flows are specified using filters Filters can also match individual end-to-end application flows"). Id. at 3. A flow would comprise a "message" under Implicit's apparent claim constructions. See section IV.C), *the method comprising:*

providing a plurality of components (Decasper98 teaches that "[o]ne of the novel features of our design is the ability to bind different plugins to individual flows." Id. at 1), *each component being a software routine for converting data* (Id. at 2

Art Unit: 3992

("plugins are kernel software modules that are..., responsible for performing certain functions on specified network flows.") for the first packet of the message, dynamically identifying a non-predefined sequence of components for processing the packets of the message wherein dynamically identifying includes selecting individual components to create the non-predefined sequence of components after the first packet is received (When the first packet of a new flow arrives, Decasper98 performs an expensive series of filter operations to determine the correct sequence of plugin components to be applied to the flow. See Ex. 25 at 5-6 ("The processing of the first packet of a new flow..., involves n filter table lookups to create a single entry in the flow table for the new flow."). This expensive series of filter operations does not need to be repeated for subsequent packets of the flow, because the new "entry... in the flow" table serves as a fast cache for future lookup of packets belonging to that flow," and the entry "stores pointers to the appropriate plugins." Id. at 5. Performance is thus enhanced for subsequent packets of the flow, since "[u]sually, filter table lookups are much slower than flow table lookups." Id. See also id. at 3 ("Subsequent packets get this information from a fast flow cache which temporarily stores the information gathered by processing the first packet."). Decasper98 assigns the sequence of plugins to the flow on the basis of lookups in multiple independent "filter tables." E.g., id. at 5-7 ("The processing of the first packet of a new flow..., involves n filter table lookup to create a single entry in the flow table for the new flow"); 7 ("multiple lookups (in different filter tables)"). E.g., a first filter table determines whether a first plugin is added to the sequence, a second independent filter

Art Unit: 3992

table determines whether a second plugin is added, a third independent filter table determines whether a third plugin is added, and so on. See Id. at 5-7.

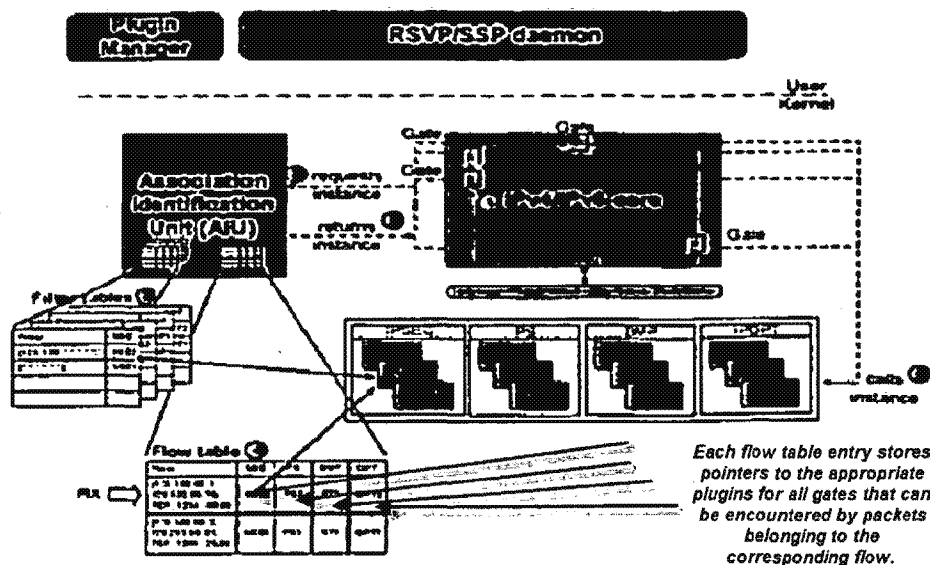


Figure 3. : System Architecture and Data Path

In the AIU, all flows start out being uncached (i.e., they do not have an entry in the flow table). If an incoming packet belongs to an uncached flow, its lookup in the flow table data structure will fail (i.e., there is a cache miss). In this case, the packet needs to be looked up in a different data structure that we call a filter table. Filter tables store the bindings between filters and plugins for each gate. The filter table lookup algorithm finds the most specific matching filter (described later) that has been installed in the table, and returns the corresponding plugin instance. Usually, filter table lookups are much slower than flow table lookups. An entry for a flow in the flow table serves as a fast cache for future lookups of packets belonging to that flow. Each flow table entry stores pointers to the appropriate plugins for all gates that can be encountered by packets belonging to the corresponding flow. The processing of the first packet of a new flow with n gates involves n filter table lookups to create a single entry in the flow table for the new flow.

for the first packet of the message, dynamically identifying a non-predefined sequence of components for processing the packets of the message wherein dynamically identifying includes selecting individual components to create the non-predefined sequence of components after the first packet is received (when the first packet of a new flow arrives, Decasper98 performs an expensive series of filter operations to determine the correct sequence of plugin components to be applied to the flow. See Ex. 25 at 5-6 ("The processing of the first packet of a new flow... involves n filter table lookups to create a single entry in the flow table for the new flow."). This expensive series of filter operations does not need to be repeated for subsequent packets of the flow, because the new "entry... in the flow" table serves as a fast cache for future lookup of packets belonging to that flow," and the entry "stores pointers to the appropriate plugins." Id. at 5. Performance is thus enhanced for subsequent packets of the flow, since "[u]sually, filter table lookups are much slower than flow table lookups." Id. See also id. at 3 ("Subsequent packets get this information from a fast flow cache which temporarily stores the information gathered by processing the first packet.").

Decasper98 assigns the sequence of plugins to the flow on the basis of lookups in multiple independent "filter tables." E.g., id. at 5-7 ("The processing of the first packet of a new flow... involves n filter table lookup to create a single entry in the flow table for the new flow"); 7 ("multiple lookups (in different filter tables)"). E.g., a first filter table determines whether a first plugin is added to the sequence, a second independent filter

Art Unit: 3992

table determines whether a second plugin is added, a third independent filter table determines whether a third plugin is added, and so on. See *id.* at 5-7.

This leads "exponentially" to an enormous number of possible sequences that might be applied to the first packet of a flow when it arrives, "even with very few installed filters." See *id.* at 7.2. These various possible sequences are not stored or enumerated anywhere in the system ahead of time. Instead, the sequence of plugins for a flow is generated algorithmically when the first packet of a flow arrives, by applying a series of filter operation to packet data which was not available to the system until that moment. See *id.* at 5-7.

Decasper98 explicitly considers and rejects a "theoretically possible" alternative approach, which is to replace this system of multiple independent filters with "a single global filter table." *Id.* at 7.. Under this alternative approach, only a single filter would apply to a particular flow, and that single filter would specify the entire sequence of components to be applied to it. See *id.* When the first packet arrived, the system would find the single matching filter and then essentially just read off the sequence of components to be applied to that flow. See *id.* Thus, the sequence would be pre-defined and readily identifiable as such in a specific filter entry, even before the first packet arrived.

However, Decasper98 rejects this approach as "practically infeasible because the space requirements for the global table can, even with very few installed filters, increase very quickly (exponentially) to unacceptable levels." *Id.* In other words, Decasper98's multiple filter table approach implies so many potential valid sequences

Art Unit: 3992

that it is impossible to even enumerate them all ahead of time in memory--since they would not fit.

Instead, Decasper98 adopts an algorithmic approach where the correct sequence is generated dynamically on demand, by applying the series of multiple filters to the first packet when it arrives. Thus, under Implicit's apparent claim constructions, Decasper98 discloses "for the first packet of the message, dynamically identifying a non-predefined sequence of components for processing the packets of the message."

Decasper98 discloses this "dynamically identifying" claim element under Implicit's apparent claim constructions. Decasper98 also teaches "selecting individual components to create the non-predefined sequence of components after the first packet is received." As explained above, after the first packet of a flow arrives, Descasper98 applies a series of independent filters to it, each of which may select a different individual plugin. *Id.* at 5-7. See also, e.g., *id.* at 4 (Figure 2, showing various individual plugins that might be selected within each category, e.g., "BMP1 BMP2 BMP3"). The very purpose of this architecture is to apply the right specific individual plugins in a tailored manner to each particular flow. E.g., *id.* at 2 ("it is very important to be able to quickly and efficiently classify packets into flows, and to apply different policies to different flows"), 3, 7)

and storing an indication of each of the identified components so that the non-predefined sequence does not need to be re-identified for subsequent packets of the message; and for each of a plurality of packets of the message in sequence, for each of a plurality of components in the identified non-predefined

Art Unit: 3992

sequence, retrieving state information (pointers) relating to performing the processing of the component with the previous packet of the message; performing the processing of the identified component with the packet and the retrieved state information; and storing state information relating to the processing of the component with the packet for use when processing the next packet of the message (

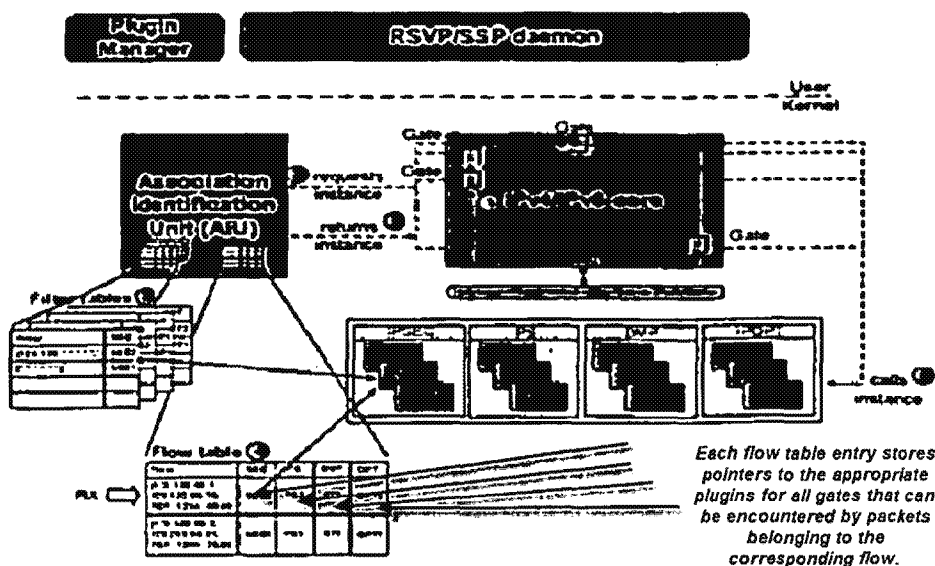


Figure 3. : System Architecture and Data Path

In the AIU, all flows start out being uncached (i.e., they do not have an entry in the flow table). If an incoming packet belongs to an uncached flow, its lookup in the flow table data structure will fail (i.e., there is a cache miss). In this case, the packet needs to be looked up in a different data structure that we call a filter table. Filter tables store the bindings between filters and plugins for each gate. The filter table lookup algorithm finds the most specific matching filter (described later) that has been installed in the table, and returns the corresponding plugin instance. Usually, filter table lookups are much slower than flow table lookups. An entry for a flow in the flow table serves as a fast cache for future lookups of packets belonging to that flow. Each flow table entry stores pointers to the appropriate plugins for all gates that can be encountered by packets belonging to the corresponding flow. The processing of the first packet of a new flow with n gates involves n filter table lookups to create a single entry in the flow table for the new flow.

This cycle is executed only for the first packet arriving on an uncached flow. Subsequent packets follow a faster path because of the cached entry in the flow table. Note that in our system, we have created optimized implementations of both the flow and filter tables, allowing for high performance on both the cached and uncached paths. These implementations are described in Section 5.

Cached flow processing involves the following sequence:

- **Processing at the first gate:** When a packet from a cached flow encounters the first gate, the AIU is called to request the plugin instance. This time, the pointer to the instance requested is already in the flow table. The flow table is looked up efficiently, and the plugin instance pointer corresponding to the calling gate is returned. No filter table lookups are required.
- **Associating the packet with a flow index:** Together with the instance requested, the AIU returns a pointer to the row in the flow table where the information associated with the flow is stored. This pointer is called the flow index (FIX), and is stored in the packet's mbuf¹. The instance is then called to process the packet, following which the IP stack passes the packet on to the next gate.
- **Processing at subsequent gates:** Once the packet has made its way past the first gate, the AIU does not have to be called upon to classify the packets at the remaining gates. Macros implementing a gate can retrieve the instance pointers cached in the flow table by accessing the FIX stored in the packet. This allows us to pass packets to the appropriate instances in a very efficient manner using an indirect function call instead of a "hardwired" function call. We show in section 7 that this does not imply significant performance penalties.

Decasper98 does not explicitly teach ***converting data with an input format into data with an output format; processing the packet of the message such that the output format matches the input format of the next component.***

Regarding the limitation "such that the output format of the components ... match the input format of the next component," it was well-known to those of ordinary skill in the art that certain operations on a packet must be performed in a certain order: e.g., if a packet is first converted into an encrypted format by a first component, a subsequent component would be unable to, e.g., process any IPv6 option headers in the packet, or to insert any new ones (because it was expecting to receive the packet in an unencrypted format). Thus, it was certainly obvious for one of ordinary skill in the art to arrange the sequence of components in a compatible manner, such that the output format of one matches the input format of the next.

8. Claims 15 and 35 are rejected under 35 U.S.C. 102(e) as being anticipated by Kerr et al. (US PAT 6243667, hereinafter Kerr).

In regards claim 15, Kerr anticipates ***a method in a computer system*** (column 2, lines 30-32, However, those skilled in the art would recognize, after perusal of this application, that embodiments of the invention may be implemented using a set of general purpose computers operating under program control, and that modification of a set of general purpose computers to implement the process steps and data structures described herein would not require undue invention) ***for demultiplexing packets of messages*** (column 1 lines 59-60, The invention provides a method and system for

Art Unit: 3992

switching in networks responsive to message flow patterns. A message "flow" is defined to comprise a set of packets to be transmitted between a particular source and a particular destination. When routers in a network identify a new message flow, they determine the proper processing for packets in that message flow and cache that information for that message flow. Thereafter, when routers in a network identify a packet which is part of that message flow, they process that packet according to the proper processing for packets in that message flow. The proper processing may include a determination of a destination port for routing those packets and a determination of whether access control permits routing those packets to their indicated destination), ***the method comprising:***

dynamically identifying a non-predefined sequence of components for processing each message based on the first packet of the message so that subsequent packets of the message can be processed without re-identifying the components (claim 1, lines 31-40, column 4 lines 20-34, column 3 line 38-column 6 line 27 identifying a first one message of a first plurality of messages associated with an application layer, said first plurality of messages having at least one policy treatment in common, said first plurality of messages being identified in response to an address of a selected source device and an address of a selected destination device, wherein said policy treatment comprises at least one of the access control information, security information, queuing information, accounting information, traffic profiling information, and policy information; In a preferred embodiment, the proper treatment of packets 150 in the message flow 160 includes treatment with regard to switching (thus, the routing

Art Unit: 3992

device 140 determines an output port for switching packets 150 in the message flow 160), with regard to access control (thus, the routing device 140 determines whether packets 150 in the message flow 160 meet the requirements of access control, as defined by access control lists in force at the routing device 140), with regard to accounting (thus, the routing device 140 creates an accounting record for the message flow 160), with regard to encryption (thus, the routing device 140 determines encryption treatment for packets 150 in the message flow 160), and any special treatment for packets 150 in the message flow 160. FIG. 2 shows a method for routing in networks responsive to message flow patterns. In broad overview, the method for routing in networks responsive to message flow patterns comprises two parts. In a first part, the routing device 140 builds and uses a flow cache described in further detail with regard to FIG. 3), in which routing information to be used for packets 150 in each particular message flow 160 is recorded and from which such routing information is retrieved for use...A method 200 for routing in networks responsive to message flow patterns is performed by the routing device 140. At a flow point 210, the routing device 140 is disposed for building and using the flow cache. At a step 221, the routing device 140 receives a packet 150. At a step 222, the routing device 140 identifies a message flow 160 for the packet 150. In a preferred embodiment, the routing device 140 examines a header for the packet 150 and identifies the IP address for the source device 120, the IP address for the destination device 130, and the protocol type for the packet 150. The routing device 140 determines the port number for the source device 120 and the port number for the destination device 130 responsive to the protocol type. Responsive to

this set of information, the routing device 140 determines a flow key 310 (described with reference to FIG. 3) for the message flow 160. At a step 223, the routing device 140 performs a lookup in a flow cache for the identified message flow 160. If the lookup is unsuccessful, the identified message flow 160 is a "new" message flow 160, and the routing device 140 continues with the step 224. If the lookup is successful, the identified message flow 160 is an "old" message flow 160, and the routing device 140 continues with the step 225. In a preferred embodiment, the routing device 140 determines a hash table key responsive to the flow key 310. This aspect of the step 223 is described in further detail with regard to FIG. 3. At a step 224, the routing device 140 builds a new entry in the flow cache. The routing device 140 determines proper treatment of packets 150 in the message flow 160 and enters information regarding such proper treatment in a data structure pointed to by the new entry in the flow cache. In a preferred embodiment, the routing device 140 determines the proper treatment by performing a lookup in an IP address cache as shown in FIG. 4. In a preferred embodiment, the proper treatment of packets 150 in the message flow 160 includes treatment with regard to switching (thus, the routing device 140 determines an output port for switching packets 150 in the message flow 160), with regard to access control (thus, the routing device 140 determines whether packets 150 in the message flow 160 meet the requirements of access control, as defined by access control lists in force at the routing device 140), with regard to accounting (thus, the routing device 140 creates an accounting record for the message flow 160), with regard to encryption (thus, the routing device 140 determines encryption treatment for packets 150 in the message

Art Unit: 3992

flow 160), and any special treatment for packets 150 in the message flow 160. In a preferred embodiment, the routing device 140 performs any special processing for new message flows 160 at this time... Thereafter, the routing device 140 proceeds with the step 225, using the information from the new entry in the flow cache, just as if the identified message flow 160 were an "old" message flow 160 and the lookup in a flow cache had been successful. At a step 225, the routing device 140 retrieves routing information from the entry in the flow cache for the identified message flow 160. In a preferred embodiment, the entry in the flow cache includes a pointer to a rewrite function for at least part of a header for the packet 150. If this pointer is non-null, the routing device 140 invokes the rewrite function to alter the header for the packet 150. At a step 226, the routing device 140 routes the packet 150 responsive to the routing information retrieved at the step 225. Thus, in a preferred embodiment, the routing device 140 does not separately determine, for each packet 150 in the message flow 160, the information stored in the entry in the flow cache. Rather, when routing a packet 150 in the message flow 160, the routing device 140 reads the information from the entry in the flow cache and treats the packet 150 according to the information in the entry in the flow cache. Thus, in a preferred embodiment, the routing device 140 routes the packet 150 to an output port, determines whether access is allowed for the packet 150, determines encryption treatment for the packet 150, and performs any special treatment for the packet 150, all responsive to information in the entry in the flow cache. In a preferred embodiment, the routing device 140 also enters accounting information in the entry in the flow cache for the packet 150. When routing each packet 150 in the

Art Unit: 3992

message flow 160, the routing device 140 records the cumulative number of packets 150 and the cumulative number of bytes for the message flow 160. Because the routing device 140 processes each packet 150 in the message flow 160 responsive to the entry for the message flow 160 in the flow cache, the routing device 140 is able to implement administrative policies which are designated for each message flow 160 rather than for each packet 150),

wherein different non-predefined sequences of components can be identified for different messages, each component being a software routine, and wherein dynamically identifying includes selecting individual components to create the non-predefined sequence of components (Kerr discloses a "plurality of components" for processing messages. For example, claim 1 of Kerr describes using a "plurality of devices" to apply "policy treatments" to a "plurality of messages," where policy treatments are used to perform "access control,security," "queuing," "accounting," "traffic profiling," etc. Id. at 10:27-40. Processing components can include "treatment with regard to switching," "access control," and "encryption." Id. at 4:20-34. "[S]pecial processing" can include "authentication" techniques "useful for implementing security 'firewalls.'" Id. at 35-46. Kerr further discloses that a "rewrite function" may be invoked "to alter the header for the packet." Id. at 4:55-62. These components can be used for "converting data with an input format into data with an output format," under Implicit' s apparent claim constructions, for example (as described above), the "encryption" and "rewrite" components to "alter" data to be processed. Id. at 4:30- 31, 4:55-62. The processing components of Kerr comprise "software routine" embodiments,

Art Unit: 3992

as Kerr states that the processing instrumentality "may include specific hardware constructed or programmed performing the process steps described herein" or "a general purpose processor operating under program control." Id. at 2:51-55; see also id. at Figs. 3-4 (illustrating software data structures). Kerr discloses rather than applying a single predefined sequence to all flows, Kerr "determines proper treatment of packets 150 in the message flow" only when it "build a new entry in the flow cache" for that specific flow. Ex. 15 at 4:12-18. This includes, e.g., "determin[ing] encryption treatment for packets 150 in the message flow ... and any special treatment for packets 150 in the message flow." Id. at 4:31-34..); and

for each packet of each message, performing the processing of the identified non-predefined sequence of components of the message wherein state information generated by performing the processing of a component for a packet is available, to the component when the component processes the next packet of the message (After receiving the first packet of a new flow, Kerr builds a new flow entry that is cached in memory, which constitutes "storing". Kerr also explains that building and caching a flow entry upon receiving the first new packet in a flow is specifically performed so that information "does not need to be re-identified for subsequent packets of the message," as that term is apparently construed by Implicit. Kerr explains that, for the sake of efficiency: information about message flow patterns is used to identify packets for which processing has already been determined, and therefore to process those packets without having to re-determine the same processing Thus, in a preferred embodiment, the routing device 140 does not separately determine, for each

Art Unit: 3992

packet 150 in the message flow 160, the information stored in the entry in the flow cache. Rather, when routing a packet 150 in the message flow 160, the routing device 140 reads the information from the entry in the flow cache and treats the packet 150 according to the information in the entry in the flow cache. Ex. 15 at 1:33-36, 4:64-5:4. In other words, when the first packet of a flow arrives, Kerr goes through the somewhat expensive and elaborate process of determining how all the packets of that flow should be treated: e.g., whether they should be encrypted, whether they should be modified or partially re-written, and where they should be routed next. Id. at 1:33-35, 4:13-60. It then records all this information about the proper processing for a flow by "build[ing] a new entry in the flow cache" for the flow, so the proper processing does not have to be wastefully and redundantly determined again for subsequent packets of the flow. Id. at 4:12-13. Kerr discloses this "state information" element. Implicit has taken a broad view of the "state information" limitations, arguing that they cover the retrieval, use, and storage of the identified sequence of components (e.g., a flow record) after the first packet is received. As demonstrated above (for the "storing an indication" element), Kerr retrieves, uses, and stores flow records in this manner to facilitate processing of packets in the same message after the first packet is received and a flow entry built. Kerr also discloses the retrieval, use, and storage of state information on a component-by-component basis. For example, in one embodiment of Kerr, there are components for access control, encryption, "special treatment," accounting, rewrite, among others. Ex. 15 at 5:5-25. The processing by these components is "all responsive to information in the entry in the flow cache." Id. at 5:9-10. As a specific example, an accounting

Art Unit: 3992

component can maintain state information, such as "time stamp" data, "a cumulative count for the number of packets," and "a cumulative count for the number of bytes." *Id.* at 6:58-63. Kerr later uses timing information to identify expired or otherwise invalid flows (among other reasons). *Id.* at 5:52 - 6:19. As another example, Kerr can retrieve the latest "usage information regarding relative use of network resources" in order to appropriately prioritize traffic using the relevant component. *Id.* at 5:41-49).

In regard to claim 35, Kerr anticipates ***a computer-readable medium containing instructions*** (column 2, lines 30-32, However, those skilled in the art would recognize, after perusal of this application, that embodiments of the invention may be implemented using a set of general purpose computers operating under program control, and that modification of a set of general purpose computers to implement the process steps and data structures described herein would not require undue invention) ***for demultiplexing packets of messages*** (column 1 lines 59-60, The invention provides a method and system for switching in networks responsive to message flow patterns. A message "flow" is defined to comprise a set of packets to be transmitted between a particular source and a particular destination. When routers in a network identify a new message flow, they determine the proper processing for packets in that message flow and cache that information for that message flow. Thereafter, when routers in a network identify a packet which is part of that message flow, they process that packet according to the proper processing for packets in that message flow. The proper processing may include a determination of a destination port for routing those

Art Unit: 3992

packets and a determination of whether access control permits routing those packets to their indicated destination), **by method comprising:**

dynamically identifying a message-specific non-predefined sequence of components for processing the packets of each message upon receiving the first packet of the message wherein subsequent packets of the message can use the message-specific non-predefined sequence identified when the first packet was received (claim 1, lines 31-40, column 4 lines 20-34, column 3 line 38-column 6 line 27 identifying a first one message of a first plurality of messages associated with an application layer, said first plurality of messages having at least one policy treatment in common, said first plurality of messages being identified in response to an address of a selected source device and an address of a selected destination device, wherein said policy treatment comprises at least one of the access control information, security information, queuing information, accounting information, traffic profiling information, and policy information; In a preferred embodiment, the proper treatment of packets 150 in the message flow 160 includes treatment with regard to switching (thus, the routing device 140 determines an output port for switching packets 150 in the message flow 160), with regard to access control (thus, the routing device 140 determines whether packets 150 in the message flow 160 meet the requirements of access control, as defined by access control lists in force at the routing device 140), with regard to accounting (thus, the routing device 140 creates an accounting record for the message flow 160), with regard to encryption (thus, the routing device 140 determines encryption treatment for packets 150 in the message flow 160), and any special treatment for

Art Unit: 3992

packets 150 in the message flow 160. FIG. 2 shows a method for routing in networks responsive to message flow patterns. In broad overview, the method for routing in networks responsive to message flow patterns comprises two parts. In a first part, the routing device 140 builds and uses a flow cache described in further detail with regard to FIG. 3), in which routing information to be used for packets 150 in each particular message flow 160 is recorded and from which such routing information is retrieved for use...A method 200 for routing in networks responsive to message flow patterns is performed by the routing device 140. At a flow point 210, the routing device 140 is disposed for building and using the flow cache. At a step 221, the routing device 140 receives a packet 150. At a step 222, the routing device 140 identifies a message flow 160 for the packet 150. In a preferred embodiment, the routing device 140 examines a header for the packet 150 and identifies the IP address for the source device 120, the IP address for the destination device 130, and the protocol type for the packet 150. The routing device 140 determines the port number for the source device 120 and the port number for the destination device 130 responsive to the protocol type. Responsive to this set of information, the routing device 140 determines a flow key 310 (described with reference to FIG. 3) for the message flow 160. At a step 223, the routing device 140 performs a lookup in a flow cache for the identified message flow 160. If the lookup is unsuccessful, the identified message flow 160 is a "new" message flow 160, and the routing device 140 continues with the step 224. If the lookup is successful, the identified message flow 160 is an "old" message flow 160, and the routing device 140 continues with the step 225. In a preferred embodiment, the routing device 140

Art Unit: 3992

determines a hash table key responsive to the flow key 310. This aspect of the step 223 is described in further detail with regard to FIG. 3. At a step 224, the routing device 140 builds a new entry in the flow cache. The routing device 140 determines proper treatment of packets 150 in the message flow 160 and enters information regarding such proper treatment in a data structure pointed to by the new entry in the flow cache. In a preferred embodiment, the routing device 140 determines the proper treatment by performing a lookup in an IP address cache as shown in FIG. 4. In a preferred embodiment, the proper treatment of packets 150 in the message flow 160 includes treatment with regard to switching (thus, the routing device 140 determines an output port for switching packets 150 in the message flow 160), with regard to access control (thus, the routing device 140 determines whether packets 150 in the message flow 160 meet the requirements of access control, as defined by access control lists in force at the routing device 140), with regard to accounting (thus, the routing device 140 creates an accounting record for the message flow 160), with regard to encryption (thus, the routing device 140 determines encryption treatment for packets 150 in the message flow 160), and any special treatment for packets 150 in the message flow 160. In a preferred embodiment, the routing device 140 performs any special processing for new message flows 160 at this time... Thereafter, the routing device 140 proceeds with the step 225, using the information from the new entry in the flow cache, just as if the identified message flow 160 were an "old" message flow 160 and the lookup in a flow cache had been successful. At a step 225, the routing device 140 retrieves routing information from the entry in the flow cache for the identified message flow 160. In a

Art Unit: 3992

preferred embodiment, the entry in the flow cache includes a pointer to a rewrite function for at least part of a header for the packet 150. If this pointer is non-null, the routing device 140 invokes the rewrite function to alter the header for the packet 150. At a step 226, the routing device 140 routes the packet 150 responsive to the routing information retrieved at the step 225. Thus, in a preferred embodiment, the routing device 140 does not separately determine, for each packet 150 in the message flow 160, the information stored in the entry in the flow cache. Rather, when routing a packet 150 in the message flow 160, the routing device 140 reads the information from the entry in the flow cache and treats the packet 150 according to the information in the entry in the flow cache. Thus, in a preferred embodiment, the routing device 140 routes the packet 150 to an output port, determines whether access is allowed for the packet 150, determines encryption treatment for the packet 150, and performs any special treatment for the packet 150, all responsive to information in the entry in the flow cache. In a preferred embodiment, the routing device 140 also enters accounting information in the entry in the flow cache for the packet 150. When routing each packet 150 in the message flow 160, the routing device 140 records the cumulative number of packets 150 and the cumulative number of bytes for the message flow 160. Because the routing device 140 processes each packet 150 in the message flow 160 responsive to the entry for the message flow 160 in the flow cache, the routing device 140 is able to implement administrative policies which are designated for each message flow 160 rather than for each packet 150),

and wherein dynamically identifying includes selecting individual components to create the message-specific non-predefined sequence of components (Kerr discloses a "plurality of components" for processing messages. For example, claim 1 of Kerr describes using a "plurality of devices" to apply "policy treatments" to a "plurality of messages," where policy treatments are used to perform "access control,security," "queuing," "accounting," "traffic profiling," etc. Id. at 10:27-40. Processing components can include "treatment with regard to switching," "access control," and "encryption." Id. at 4:20-34. "[S]pecial processing" can include "authentication" techniques "useful for implementing security 'firewalls.'" Id. at 35-46. Kerr further discloses that a "rewrite function" may be invoked "to alter the header for the packet." Id. at 4:55-62. These components can be used for "converting data with an input format into data with an output format," under Implicit' s apparent claim constructions, for example (as described above), the "encryption" and "rewrite" components to "alter" data to be processed. Id. at 4:30- 31, 4:55-62. The processing components of Kerr comprise "software routine" embodiments, as Kerr states that the processing instrumentality "may include specific hardware constructed or programmed performing the process steps described herein" or "a general purpose processor operating under program control." Id. at 2:51-55; see also id. at Figs. 3-4 (illustrating software data structures)); and

for each packet of the message, invoking the identified non-predefined sequence of components in sequence to perform the processing of each component for the packet wherein each component saves message-specific state

Art Unit: 3992

information so that that component can use the saved message-specific state information when that component performs its processing on the next packet of the message (After receiving the first packet of a new flow, Kerr builds a new flow entry that is cached in memory, which constitutes "storing". Kerr also explains that building and caching a flow entry upon receiving the first new packet in a flow is specifically performed so that information "does not need to be re-identified for subsequent packets of the message," as that term is apparently construed by Implicit. Kerr explains that, for the sake of efficiency: information about message flow patterns is used to identify packets for which processing has already been determined, and therefore to process those packets without having to re-determine the same processing Thus, in a preferred embodiment, the routing device 140 does not separately determine, for each packet 150 in the message flow 160, the information stored in the entry in the flow cache. Rather, when routing a packet 150 in the message flow 160, the routing device 140 reads the information from the entry in the flow cache and treats the packet 150 according to the information in the entry in the flow cache. Ex. 15 at 1:33-36, 4:64-5:4. In other words, when the first packet of a flow arrives, Kerr goes through the somewhat expensive and elaborate process of determining how all the packets of that flow should be treated: e.g., whether they should be encrypted, whether they should be modified or partially re-written, and where they should be routed next. Id. at 1:33-35, 4:13-60. It then records all this information about the proper processing for a flow by "build[ing] a new entry in the flow cache" for the flow, so the proper processing does not have to be wastefully and redundantly determined again for subsequent packets of the flow. Id. at

Art Unit: 3992

4:12-13. Kerr discloses this "state information" element. Implicit has taken a broad view of the "state information" limitations, arguing that they cover the retrieval, use, and storage of the identified sequence of components (e.g., a flow record) after the first packet is received. As demonstrated above (for the "storing an indication" element), Kerr retrieves, uses, and stores flow records in this manner to facilitate processing of packets in the same message after the first packet is received and a flow entry built. Kerr also discloses the retrieval, use, and storage of state information on a component-by-component basis. For example, in one embodiment of Kerr, there are components for access control, encryption, "special treatment," accounting, rewrite, among others. Ex. 15 at 5:5-25. The processing by these components is "all responsive to information in the entry in the flow cache." Id. at 5:9-10. As a specific example, an accounting component can maintain state information, such as "time stamp" data, "a cumulative count for the number of packets," and "a cumulative count for the number of bytes." Id. at 6:58-63. Kerr later uses timing information to identify expired or otherwise invalid flows (among other reasons). Id. at 5:52 - 6:19. As another example, Kerr can retrieve the latest "usage information regarding relative use of network resources" in order to appropriately prioritize traffic using the relevant component. Id. at 5:41-49)

Art Unit: 3992

9. Claims 15 and 35 are rejected under 35 U.S.C. 102(b) as being anticipated by Decasper98.

In regards to claim 15, Decasper98 anticipates a method in a computer system

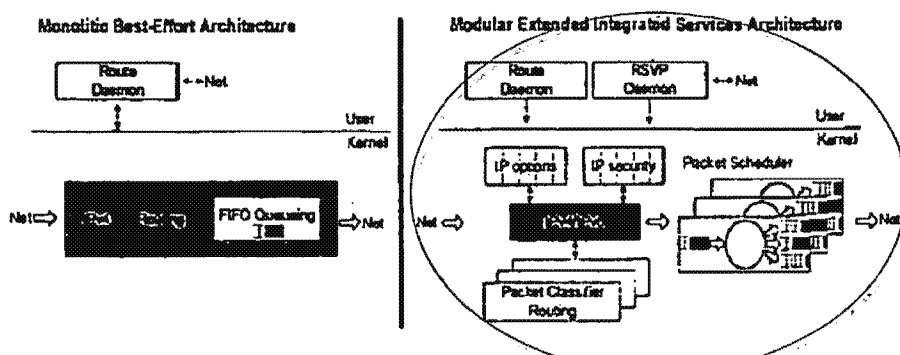


Figure 1. : Best Effort-vs
Extended Integrated Services Router (EISR)

for demultiplexing packets of messages (Decasper98 explains: "it is very important to be able to quickly and efficiently classify packets into flows, and to apply different policies to different flows; these are both things that our architecture excels at doing." Ex. 25 at 2. Flows may represent "longer lived packet streams": Because the deployment of multimedia data sources and applications (e.g. real-time audio/video) will produce longer lived packet streams with more packets per session than is common in today's environment, an integrated services router architecture should support the notion of flows and build upon it. Id. at 3. A flow is defined as a group of packets which satisfy a specific filter. See id. at 3 ("Sets of flows are specified using filters Filters can also match individual end-to-end application flows"). Id. at 3. A flow would comprise a "message" under Implicit's apparent claim constructions. See section IV.C), *the method comprising:*

dynamically identifying a non-predefined sequence of components for processing each message based on the first packet of the message so that subsequent packets of the message can be processed without re-identifying the components, wherein different non-predefined sequences of components can be identified for different messages (when the first packet of a new flow arrives, Decasper98 performs an expensive series of filter operations to determine the correct sequence of plugin components to be applied to the flow. See Ex. 25 at 5-6 ("The processing of the first packet of a new flow... involves n filter table lookups to create a single entry in the flow table for the new flow."). This expensive series of filter operations does not need to be repeated for subsequent packets of the flow, because the new "entry... in the flow" table serves as a fast cache for future lookup of packets belonging to that flow," and the entry "stores pointers to the appropriate plugins." Id. at 5. Performance is thus enhanced for subsequent packets of the flow, since "[u]sually, filter table lookups are much slower than flow table lookups." Id. See also id. at 3 ("Subsequent packets get this information from a fast flow cache which temporarily stores the information gathered by processing the first packet." Decasper98 explains: "it is very important to be able to... apply different policies to different flows." Ex. 25 at 2. This is why Decasper98 applies a series of filters to each flow, wherein each filter may select a specific plugin component implementing a different policy. See Id. at 5-7).

Decasper98 assigns the sequence of plugins to the flow on the basis of lookups in multiple independent "filter tables." E.g., id. at 5-7 ("The processing of the first packet of a new flow... involves n filter table lookup to create a single entry in the flow table for

Art Unit: 3992

the new flow"); 7 ("multiple lookups (in different filter tables)"). E.g., a first filter table determines whether a first plugin is added to the sequence, a second independent filter table determines whether a second plugin is added, a third independent filter table determines whether a third plugin is added, and so on. See *id.* at 5-7.

This leads "exponentially" to an enormous number of possible sequences that might be applied to the first packet of a flow when it arrives, "even with very few installed filters." See *id.* at 7.2. These various possible sequences are not stored or enumerated anywhere in the system ahead of time. Instead, the sequence of plugins for a flow is generated algorithmically when the first packet of a flow arrives, by applying a series of filter operation to packet data which was not available to the system until that moment. See *id.* at 5-7.

Decasper98 explicitly considers and rejects a "theoretically possible" alternative approach, which is to replace this system of multiple independent filters with "a single global filter table." *Id.* at 7. Under this alternative approach, only a single filter would apply to a particular flow, and that single filter would specify the entire sequence of components to be applied to it. See *id.* When the first packet arrived, the system would find the single matching filter and then essentially just read off the sequence of components to be applied to that flow. See *id.* Thus, the sequence would be pre-defined and readily identifiable as such in a specific filter entry, even before the first packet arrived.

However, Decasper98 rejects this approach as "practically infeasible because the space requirements for the global table can, even with very few installed filters,

Art Unit: 3992

increase very quickly (exponentially) to unacceptable levels." *Id.* In other words, Decasper98's multiple filter table approach implies so many potential valid sequences that it is impossible to even enumerate them all ahead of time in memory--since they would not fit.

Instead, Decasper98 adopts an algorithmic approach where the correct sequence is generated dynamically on demand, by applying the series of multiple filters to the first packet when it arrives. Thus, under Implicit's apparent claim constructions, Decasper98 discloses "for the first packet of the message, dynamically identifying a non-predefined sequence of components for processing the packets of the message."

Decasper98 discloses this "dynamically identifying" claim element under Implicit's apparent claim constructions. Decasper98 also teaches "selecting individual components to create the non-predefined sequence of components after the first packet is received." As explained above, after the first packet of a flow arrives, Decasper98 applies a series of independent filters to it, each of which may select a different individual plugin. *Id.* at 5-7. See also, e.g., *id.* at 4 (Figure 2, showing various individual plugins that might be selected within each category, e.g., "BMP1 BMP2 BMP3"). The very purpose of this architecture is to apply the right specific individual plugins in a tailored manner to each particular flow. E.g., *id.* at 2 ("it is very important to be able to quickly and efficiently classify packets into flows, and to apply different policies to different flows"), 3, 7.

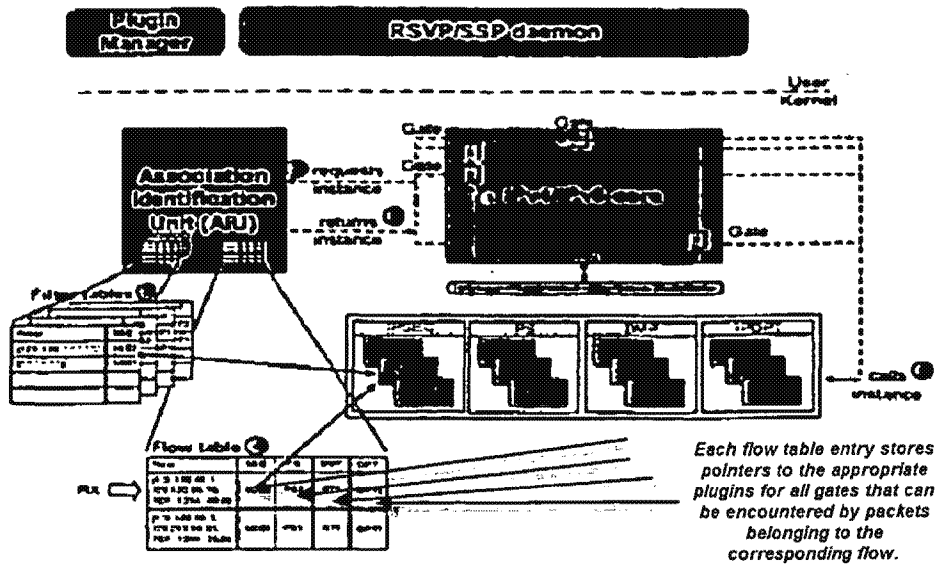


Figure 3. : System Architecture and Data Path

In the AU, all flows start out being uncached (i.e., they do not have an entry in the flow table). If an incoming packet belongs to an uncached flow, its lookup in the flow table data structure will fail (i.e., there is a cache miss). In this case, the packet needs to be looked up in a different data structure that we call a filter table. Filter tables store the bindings between filters and plugins for each gate. The filter table lookup algorithm finds the most specific matching filter (described later) that has been installed in the table, and returns the corresponding plugin instance. Usually, filter table lookups are much slower than flow table lookups. An entry for a flow in the flow table serves as a fast cache for future lookups of packets belonging to that flow. Each flow table entry stores pointers to the appropriate plugins for all gates that can be encountered by packets belonging to the corresponding flow. The processing of the first packet of a new flow with n gates involves n filter table lookups to create a single entry in the flow table for the new flow.

This cycle is executed only for the first packet arriving on an uncached flow. Subsequent packets follow a faster path because of the cached entry in the flow table. Note that in our system, we have created optimized implementations of both the flow and filter tables, allowing for high performance on both the cached and uncached paths. These implementations are described in Section 5.

Cached flow processing involves the following sequence:

- **Processing at the first gate:** When a packet from a cached flow encounters the first gate, the AIU is called to request the plugin instance. This time, the pointer to the instance requested is already in the flow table. The flow table is looked up efficiently, and the plugin instance pointer corresponding to the calling gate is returned. No filter table lookups are required.
- **Associating the packet with a flow index:** Together with the instance requested, the AIU returns a pointer to the row in the flow table where the information associated with the flow is stored. This pointer is called the flow index (FIX), and is stored in the packet's mbuf^d. The instance is then called to process the packet, following which the IP stack passes the packet on to the next gate.
- **Processing at subsequent gates:** Once the packet has made its way past the first gate, the AIU does not have to be called upon to classify the packets at the remaining gates. Macros implementing a gate can retrieve the instance pointers cached in the flow table by accessing the FIX stored in the packet. This allows us to pass packets to the appropriate instances in a very efficient manner using an indirect function call instead of a "hardwired" function call. We show in section 7 that this does not imply significant performance penalties.

each component being a software routine (Id. at 2 ("plugins are kernel software modules that are... responsible for performing certain functions on specified

Art Unit: 3992

network flows.") for the first packet of the message, dynamically identifying a non-predefined sequence of components for processing the packets of the message wherein dynamically identifying includes selecting individual components to create the non-predefined sequence of components after the first packet is received (When the first packet of a new flow arrives, Decasper98 performs an expensive series of filter operations to determine the correct sequence of plugin components to be applied to the flow. See Ex. 25 at 5-6 ("The processing of the first packet of a new flow..., involves n filter table lookups to create a single entry in the flow table for the new flow."). This expensive series of filter operations does not need to be repeated for subsequent packets of the flow, because the new "entry... in the flow" table serves as a fast cache for future lookup of packets belonging to that flow," and the entry "stores pointers to the appropriate plugins." Id. at 5. Performance is thus enhanced for subsequent packets of the flow, since "[u]sually, filter table lookups are much slower than flow table lookups." Id. See also id. at 3 ("Subsequent packets get this information from a fast flow cache which temporarily stores the information gathered by processing the first packet."). Decasper98 assigns the sequence of plugins to the flow on the basis of lookups in multiple independent "filter tables." E.g., id. at 5-7 ("The processing of the first packet of a new flow..., involves n filter table lookup to create a single entry in the flow table for the new flow"); 7 ("multiple lookups (in different filter tables)"). E.g., a first filter table determines whether a first plugin is added to the sequence, a second independent filter table determines whether a second plugin is added, a third independent filter table determines whether a third plugin is added, and so on. See Id. at 5-7, and wherein

dynamically identifying includes selecting individual components to create the non-predefined sequence of components; and

for each packet of each message, performing the processing of the identified non-predefined sequence of components of the message wherein state information generated by performing the processing of a component for a packet is available, to the component when the component processes the next packet of the message (

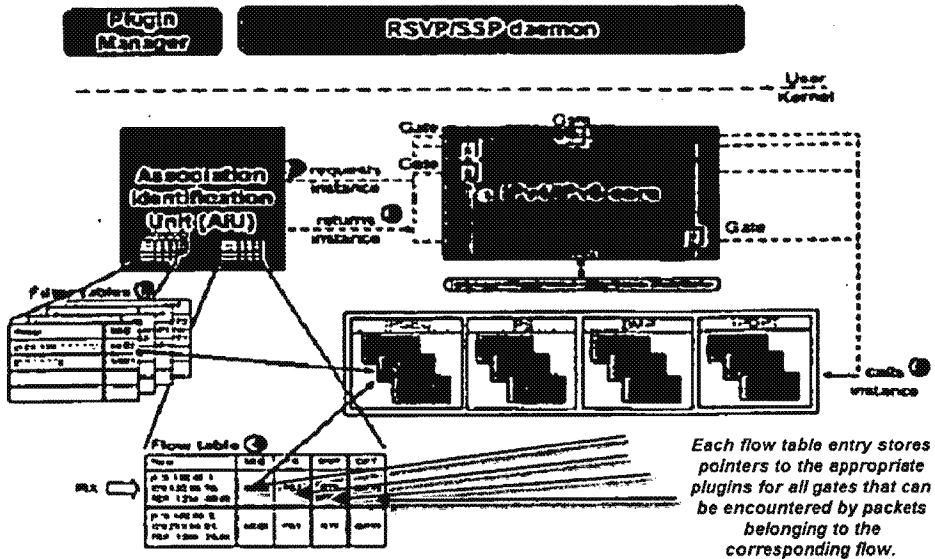


Figure 3. : System Architecture and Data Path

In the AIU, all flows start out being uncached (i.e., they do not have an entry in the flow table). If an incoming packet belongs to an uncached flow, its lookup in the flow table data structure will fail (i.e., there is a cache miss). In this case, the packet needs to be looked up in a different data structure that we call a filter table. Filter tables store the bindings between filters and plugins for each gate. The filter table lookup algorithm finds the most specific matching filter (described later) that has been installed in the table, and returns the corresponding plugin instance. Usually, filter table lookups are much slower than flow table lookups. An entry for a flow in the flow table serves as a fast cache for future lookups of packets belonging to that flow. Each flow table entry stores pointers to the appropriate plugins for all gates that can be encountered by packets belonging to the corresponding flow. The processing of the first packet of a new flow with n gates involves n filter table lookups to create a single entry in the flow table for the new flow.

This cycle is executed only for the first packet arriving on an uncached flow. Subsequent packets follow a faster path because of the cached entry in the flow table. Note that in our system, we have created optimized implementations of both the flow and filter tables, allowing for high performance on both the cached and uncached paths. These implementations are described in Section 5.

Cached flow processing involves the following sequence:

- **Processing at the first gate:** When a packet from a cached flow encounters the first gate, the AIU is called to request the plugin instance. This time, the pointer to the instance requested is already in the flow table. The flow table is looked up efficiently, and the plugin instance pointer corresponding to the calling gate is returned. No filter table lookups are required.
- **Associating the packet with a flow index:** Together with the instance requested, the AIU returns a pointer to the row in the flow table where the information associated with the flow is stored. This pointer is called the flow index (FIX), and is stored in the packet's mbuf^d. The instance is then called to process the packet, following which the IP stack passes the packet on to the next gate.
- **Processing at subsequent gates:** Once the packet has made its way past the first gate, the AIU does not have to be called upon to classify the packets at the remaining gates. Macros implementing a gate can retrieve the instance pointers cached in the flow table by accessing the FIX stored in the packet. This allows us to pass packets to the appropriate instances in a very efficient manner using an indirect function call instead of a "hardwired" function call. We show in section 7 that this does not imply significant performance penalties.

).

In regards to claim 35, Decasper98 anticipates *a computer-readable medium containing instructions*

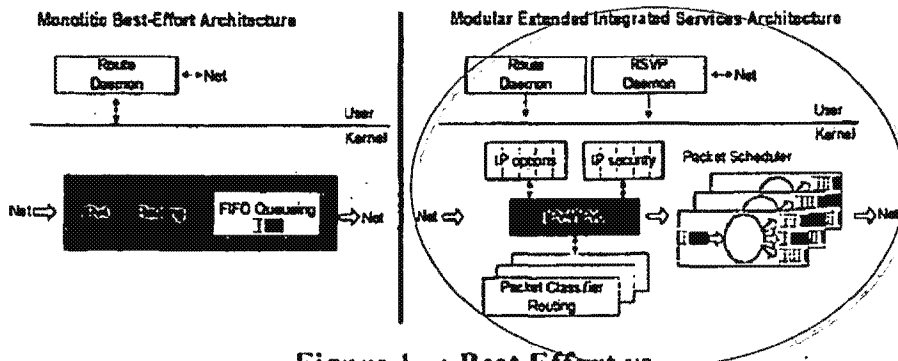


Figure 1. : Best Effort-vs
Extended Integrated Services Router (EISR)

for demultiplexing packets of messages (Decasper98 explains: "it is very important to be able to quickly and efficiently classify packets into flows, and to apply different policies to different flows; these are both things that our architecture excels at doing." Ex. 25 at 2. Flows may represent "longer lived packet streams": Because the deployment of multimedia data sources and applications (e.g. real-time audio/video) will produce longer lived packet streams with more packets per session than is common in today's environment, an integrated services router architecture should support the notion of flows and build upon it. Id. at 3. A flow is defined as a group of packets which satisfy a specific filter. See id. at 3 ("Sets of flows are specified using filters Filters can also match individual end-to-end application flows"). Id. at 3. A flow would comprise a "message" under Implicit's apparent claim constructions. See section IV.C), *by method comprising:*

dynamically identifying a message-specific non-predefined sequence of components for processing the packets of each message upon receiving the first

Art Unit: 3992

packet of the message wherein subsequent packets of the message can use the message-specific non-predefined sequence identified when the first packet was received, and wherein dynamically identifying includes selecting individual components to create the message-specific non-predefined sequence of components (when the first packet of a new flow arrives, Decasper98 performs an expensive series of filter operations to determine the correct sequence of plugin components to be applied to the flow. See Ex. 25 at 5-6 ("The processing of the first packet of a new flow..., involves n filter table lookups to create a single entry in the flow table for the new flow."). This expensive series of filter operations does not need to be repeated for subsequent packets of the flow, because the new "entry... in the flow" table serves as a fast cache for future lookup of packets belonging to that flow," and the entry "stores pointers to the appropriate plugins." *Id.* at 5. Performance is thus enhanced for subsequent packets of the flow, since "[u]sually, filter table lookups are much slower than flow table lookups." *Id.* See also *id.* at 3 ("Subsequent packets get this information from a fast flow cache which temporarily stores the information gathered by processing the first packet.").

Decasper98 assigns the sequence of plugins to the flow on the basis of lookups in multiple independent "filter tables." E.g., *id.* at 5-7 ("The processing of the first packet of a new flow..., involves n filter table lookup to create a single entry in the flow table for the new flow"); 7 ("multiple lookups (in different filter tables)"). E.g., a first filter table determines whether a first plugin is added to the sequence, a second independent filter

Art Unit: 3992

table determines whether a second plugin is added, a third independent filter table determines whether a third plugin is added, and so on. See *id.* at 5-7.

This leads "exponentially" to an enormous number of possible sequences that might be applied to the first packet of a flow when it arrives, "even with very few installed filters." See *id.* at 7.2. These various possible sequences are not stored or enumerated anywhere in the system ahead of time. Instead, the sequence of plugins for a flow is generated algorithmically when the first packet of a flow arrives, by applying a series of filter operation to packet data which was not available to the system until that moment. See *id.* at 5-7.

Decasper98 explicitly considers and rejects a "theoretically possible" alternative approach, which is to replace this system of multiple independent filters with "a single global filter table." *Id.* at 7. Under this alternative approach, only a single filter would apply to a particular flow, and that single filter would specify the entire sequence of components to be applied to it. See *id.* When the first packet arrived, the system would find the single matching filter and then essentially just read off the sequence of components to be applied to that flow. See *id.* Thus, the sequence would be pre-defined and readily identifiable as such in a specific filter entry, even before the first packet arrived.

However, Decasper98 rejects this approach as "practically infeasible because the space requirements for the global table can, even with very few installed filters, increase very quickly (exponentially) to unacceptable levels." *Id.* In other words, Decasper98's multiple filter table approach implies so many potential valid sequences

Art Unit: 3992

that it is impossible to even enumerate them all ahead of time in memory--since they would not fit.

Instead, Decasper98 adopts an algorithmic approach where the correct sequence is generated dynamically on demand, by applying the series of multiple filters to the first packet when it arrives. Thus, under Implicit's apparent claim constructions, Decasper98 discloses "for the first packet of the message, dynamically identifying a non-predefined sequence of components for processing the packets of the message."

Decasper98 discloses this "dynamically identifying" claim element under Implicit's apparent claim constructions. Decasper98 also teaches "selecting individual components to create the non-predefined sequence of components after the first packet is received." As explained above, after the first packet of a flow arrives, Descasper98 applies a series of independent filters to it, each of which may select a different individual plugin. *Id.* at 5-7. See also, e.g., *id.* at 4 (Figure 2, showing various individual plugins that might be selected within each category, e.g., "BMP1 BMP2 BMP3"). The very purpose of this architecture is to apply the fight specific individual plugins in a tailored manner to each particular flow. E.g., *id.* at 2 ("it is very important to be able to quickly and efficiently classify packets into flows, and to apply different policies to different flows"), 3, 7.

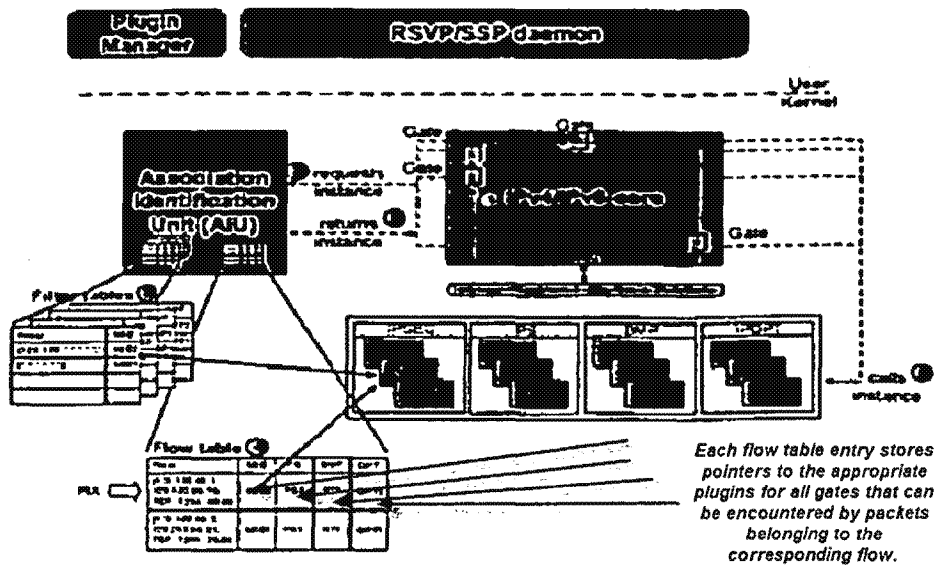


Figure 3. : System Architecture and Data Path

In the AIU, all flows start out being uncached (i.e., they do not have an entry in the flow table). If an incoming packet belongs to an uncached flow, its lookup in the flow table data structure will fail (i.e., there is a cache miss). In this case, the packet needs to be looked up in a different data structure that we call a filter table. Filter tables store the bindings between filters and plugins for each gate. The filter table lookup algorithm finds the most specific matching filter (described later) that has been installed in the table, and returns the corresponding plugin instance. Usually, filter table lookups are much slower than flow table lookups. An entry for a flow in the flow table serves as a fast cache for future lookups of packets belonging to that flow. Each flow table entry stores pointers to the appropriate plugins for all gates that can be encountered by packets belonging to the corresponding flow. The processing of the first packet of a new flow with n gates involves n filter table lookups to create a single entry in the flow table for the new flow.

This cycle is executed only for the first packet arriving on an uncached flow. Subsequent packets follow a faster path because of the cached entry in the flow table. Note that in our system, we have created optimized implementations of both the flow and filter tables, allowing for high performance on both the cached and uncached paths. These implementations are described in Section 5.

Cached flow processing involves the following sequence:

- **Processing at the first gate:** When a packet from a cached flow encounters the first gate, the AIU is called to request the plugin instance. This time, the pointer to the instance requested is already in the flow table. The flow table is looked up efficiently, and the plugin instance pointer corresponding to the calling gate is returned. No filter table lookups are required.
- **Associating the packet with a flow index:** Together with the instance requested, the AIU returns a pointer to the row in the flow table where the information associated with the flow is stored. This pointer is called the flow index (FIX), and is stored in the packet's mbuf^d. The instance is then called to process the packet, following which the IP stack passes the packet on to the next gate.
- **Processing at subsequent gates:** Once the packet has made its way past the first gate, the AIU does not have to be called upon to classify the packets at the remaining gates. Macros implementing a gate can retrieve the instance pointers cached in the flow table by accessing the FIX stored in the packet. This allows us to pass packets to the appropriate instances in a very efficient manner using an indirect function call instead of a "hardwired" function call. We show in section 7 that this does not imply significant performance penalties.

and for each packet of the message, invoking the identified non-predefined sequence of components in sequence to perform the processing of each

component for the packet wherein each component saves message-specific state information so that that component can use the saved message-specific state information when that component performs its processing on the next packet of the message

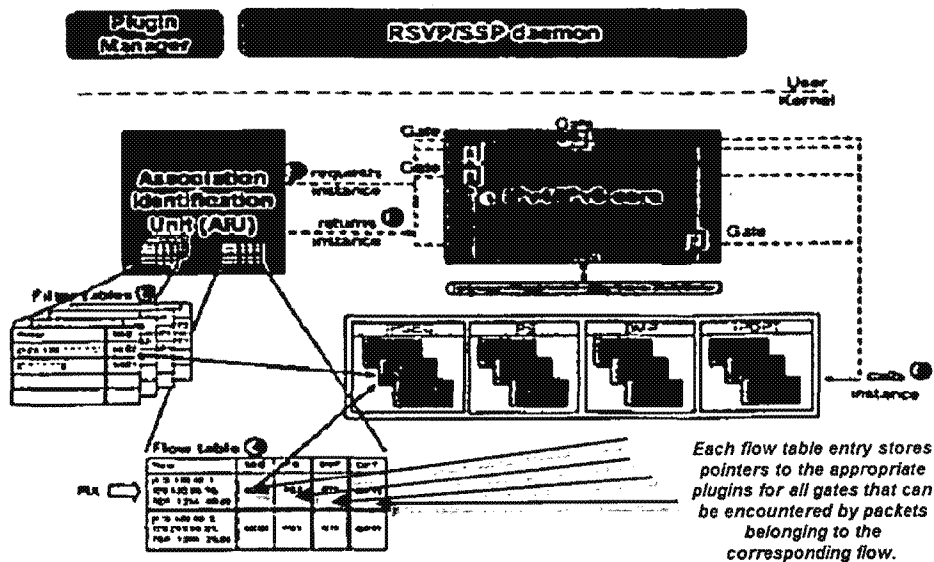


Figure 3. : System Architecture and Data Path

Art Unit: 3992

In the AIU, all flows start out being uncached (i.e., they do not have an entry in the flow table). If an incoming packet belongs to an uncached flow, its lookup in the flow table data structure will fail (i.e., there is a cache miss). In this case, the packet needs to be looked up in a different data structure that we call a filter table. Filter tables store the bindings between filters and plugins for each gate. The filter table lookup algorithm finds the most specific matching filter (described later) that has been installed in the table, and returns the corresponding plugin instance. Usually, filter table lookups are much slower than flow table lookups. An entry for a flow in the flow table serves as a fast cache for future lookups of packets belonging to that flow. Each flow table entry stores pointers to the appropriate plugins for all gates that can be encountered by packets belonging to the corresponding flow. The processing of the first packet of a new flow with n gates involves n filter table lookups to create a single entry in the flow table for the new flow.

This cycle is executed only for the first packet arriving on an uncached flow. Subsequent packets follow a faster path because of the cached entry in the flow table. Note that in our system, we have created optimized implementations of both the flow and filter tables, allowing for high performance on both the cached and uncached paths. These implementations are described in Section 5.

Cached flow processing involves the following sequence:

- **Processing at the first gate:** When a packet from a cached flow encounters the first gate, the AIU is called to request the plugin instance. This time, the pointer to the instance requested is already in the flow table. The flow table is looked up efficiently, and the plugin instance pointer corresponding to the calling gate is returned. No filter table lookups are required.
- **Associating the packet with a flow index:** Together with the instance requested, the AIU returns a pointer to the row in the flow table where the information associated with the flow is stored. This pointer is called the flow index (FIX), and is stored in the packet's mbuf¹. The instance is then called to process the packet, following which the IP stack passes the packet on to the next gate.
- **Processing at subsequent gates:** Once the packet has made its way past the first gate, the AIU does not have to be called upon to classify the packets at the remaining gates. Macros implementing a gate can retrieve the instance pointers cached in the flow table by accessing the FIX stored in the packet. This allows us to pass packets to the appropriate instances in a very efficient manner using an indirect function call instead of a "hardwired" function call. We show in section 7 that this does not imply significant performance penalties.

).

Service of Papers

10. After the filing of a request for reexamination by a third party requester, any document filed by either the patent owner or the third party requester must be served on the other party (or parties where two or more third party requester proceedings are merged) in the reexamination proceeding in the manner provided in 37 CFR 1.248. See 37 CFR 1.550(t).

Extensions of Time

11. Extensions of time under 37 CFR 1.136(a) will not be permitted in inter partes reexamination proceedings because the provisions of 37 CFR 1.136 apply only to "an applicant" and not to parties in a reexamination proceeding. Additionally, 35 U.S.C. 314(c) requires that inter partes reexamination proceedings "will be conducted with special dispatch" (37 CFR 1.937). Patent owner extensions of time in inter partes reexamination proceedings are provided for in 37 CFR 1.956. Extensions of time are not available for third party requester comments, because a comment period of 30 days from service of patent owner's response is set by statute 35 U.S.C. 314(b)(3). Time periods may be extended only upon a strong showing of sufficient cause.

Notification of Concurrent Proceedings

12. The patent owner is reminded of the continuing responsibility under 37 CFR 1.985(a), to apprise the Office of any litigation activity, or other prior or concurrent proceeding, involving the 6,629,163 patent throughout the course of this reexamination

proceeding. The third party requester is also reminded of the ability to similarly apprise the Office of any such activity or proceeding throughout the course of this reexamination proceeding. See MPEP 2686 and 2686.04.

Complete Response Reminder

13. In order to ensure full consideration of any amendments, affidavits or declarations, or other documents as evidence of patentability, such documents must be submitted in response to this Office action. Submissions after the next Office action, which is intended to be an Action Closing Prosecution (ACP), will be governed by 37 CFR 1.116(b) and (d), which will be strictly enforced.

Conclusion

14. Extensions of time under 37 CFR 1.136(a) will not be permitted in *inter partes* reexamination proceedings because the provisions of 37 CFR 1.136 apply only to "an applicant" and not to the patent owner in a reexamination proceeding. Additionally, 35 U.S.C. 314(c) requires that *inter partes* reexamination proceedings "will be conducted with special dispatch" (37 CFR 1.937). Patent owner extensions of time in *inter partes* reexamination proceedings are provided for in 37 CFR 1.956. Extensions of time are not available for third party requester comments, because a comment period of 30 days from service of patent owner's response is set by statute. 35 U.S.C. 314(b)(3).

15. All correspondence relating to this *inter partes* reexamination proceeding should be directed:

Art Unit: 3992

By EFS: Registered users may submit via the electronic filing system EFS-Web, at <https://efs.uspto.gov/efile/myportal/efs-registered>

By Mail to: *Mail Stop Inter Partes* Reexam
Attn: Central Reexamination Unit
Commissioner for Patents
United States Patent & Trademark Office
P.O. Box 1450
Alexandria, Virginia 22313-1450

By FAX to: (571) 273-9900
Central Reexamination Unit

By hand: Customer Service Window
Attn: Central Reexamination Unit
Randolph Building, Lobby Level
401 Dulany Street
Alexandria, VA 22314

For EFS-Web transmissions, 37 CFR 1.8(a)(1)(i) (C) and (ii) states that correspondence (except for a request for reexamination and a corrected or replacement request for reexamination) will be considered timely filed if (a) it is transmitted via the Office's electronic filing system in accordance with 37 CFR 1.6(a)(4), and (b) includes a certificate of transmission for each piece of correspondence stating the data of transmission, which is prior to the expiration of the set period of time in the Office action.

Any inquiry concerning this communication or earlier communications from the examiner, or as to the status of this proceeding, should be directed to the Central Reexamination Unit at telephone number (571) 272-7705.

/Salman Ahmed/
Salman Ahmed
Primary Examiner
Central Reexamination Unit - Art Unit 3992
(571) 272-8307

Conferee:

/Ovidio Escalante/

Conferee:

/Daniel J Ryman/
Supervisory Patent Examiner,
Art Unit 3992

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO INFORMATION DISCLOSURE STATEMENT BY APPLICANT (Use as many sheets as necessary)		<i>Complete if Known</i>			
		Application Number	New		
		Filing Date	February 13, 2012		
		First Named Inventor	6,629,163		
		Art Unit	3992		
		Examiner Name	Unknown		
Sheet	2	of	6	Attorney Docket Number	159291-0025(163)

NON PATENT LITERATURE DOCUMENTS			
Examiner Initials*	Cite No. ¹	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T ²
	3	PFEIFER et al., Generic Conversion of Communication Media for Supporting Personal Mobility, Multimedia Telecommunication and Applications, COST 237 Workshop, Nov. 25-27, 1996	
	4	NORTHERN TELECOM, Digital Switching Systems, ISDN Primary Rate User-Network Interface Specification, NA011, Std 08.01, Aug. 1998	
	5	NELSON et al., The Data Compression Book, 2nd Edition, Nov. 6, 1995, M&T Books, New York, NY	
	6	COX, Superdistribution: objects as property on the electronic frontier; June 4, 1996, Addison-Wesley Publishing, Reading, MA	
	7	FRANZ, Job and Stream Control in Heterogeneous Hardware and Software Architectures, April 22, 1998, Berlin, DE	
	8	van der MEER, Dynamic Configuration Management of the Equipment in Distributed Communication Environments, Oct. 6, 1996, Technische Universitat Berlin, DE	
	9	Information Sciences Institute, RFC:793, Transmission Control Protocol, DARPA Internet Program Protocol Specification, Sept. 1981, Marina Del Rey, California	
	10	MILLS et al., Principles of Information Systems Analysis and Design, copyright 1986, Academic Press, Inc., San Diego, CA	
	11	ARBANOWSKI, Generic Description of Telecommunication Services and Dynamic Resource Selection in Intelligent Communication Environments, Oct. 9, 1996, Berlin, DE	

Examiner Signature	/Salman Ahmed/	Date Considered	03/28/2012
--------------------	----------------	-----------------	------------

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

1 Applicant's unique citation designation number (optional). 2 Applicant is to place a check mark here if English language Translation is attached.
This collection of information is required by 37 CFR 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

If you need assistance in completing the form, call 1-800-PTO-9199 (1-800-786-9199) and select option 2.

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /S.A./

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO INFORMATION DISCLOSURE STATEMENT BY APPLICANT (Use as many sheets as necessary)		<i>Complete if Known</i>			
		Application Number	New		
		Filing Date	February 13, 2012		
		First Named Inventor	6,629,163		
		Art Unit	3992		
		Examiner Name	Unknown		
Sheet	3	of	6	Attorney Docket Number	159291-0025(163)

NON PATENT LITERATURE DOCUMENTS			
Examiner Initials*	Cite No. ¹	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T ²
	16	LAWSON, Cisco NetFlow Switching speeds traffic routing, InfoWorld, July 7, 1997, ProQuest Center, pg. 19	
	17	BELLARE et al., A Concrete Security Treatment of Symmetric Encryption: Analysis of the DES Modes of Operation, IEEE, Aug. 15, 1997	
	18	BELLARE, XOR MACS: New Methods for Message Authentication Using Finite Pseudorandom Functions. CRYPTO '95, LNCS 963, pp. 15-28, 1995, Berlin Heidelberg DE	
	19	IBM Raleigh Center, Local Area Network Concepts and Products: Routers and Gateways, 1st Ed., May 1996, Research Triangle Park, NC	
	20	NATIONAL INST. OF STDS AND TECH., CheckPoint FireWall-1 White Paper, Version 2.0, Sept. 1995, Germany	
	23	BELLISSARD et al., Dynamic Reconfiguration of Agent-Based Applications, Proceedings of ACM European SIGOPS Workshop, Sinatra, Sept. 1998	
	24	FRASER et al., DTE Firewalls Phase Two Measurement and Evaluation Report, TIS Report #0682, July 22, 1997, Glenwood, MD	
	25	DECASPER et al., Router Plugins A Software Architecture of Next Generation Routers, Proceedings of ACM SIGCOMM '98, Sept. 1998, Vancouver B.C.	
	26	ATKINSON, Security Architecture for the Internet Protocol, RFC: 1825, Standard Track, Naval Research Lab., Aug. 1995	
	27	KARN et al, RFC: 1883: The ESP DES-CBC Transform, Aug. 1995	

Examiner Signature	/Salman Ahmed/	Date Considered	03/28/2012
--------------------	----------------	-----------------	------------

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.
 1 Applicant's unique citation designation number (optional). 2 Applicant is to place a check mark here if English language Translation is attached.
 This collection of information is required by 37 CFR 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

If you need assistance in completing the form, call 1-800-PTO-9199 (1-800-786-9199) and select option 2.

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /S.A./

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO INFORMATION DISCLOSURE STATEMENT BY APPLICANT (Use as many sheets as necessary)		Complete if Known	
		Application Number	New
		Filing Date	February 13, 2012
		First Named Inventor	6,629,163
		Art Unit	3992
		Examiner Name	Unknown
Sheet	4	of	6
		Attorney Docket Number	159291-0025(163)

NON PATENT LITERATURE DOCUMENTS			
Examiner Initials*	Cite No. ¹	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T ²
	28	DEERING & HINDEN, Internet Protocol, Version 6 (IPv6) Specification, RFC: 1883, Standards Track, Dec. 1995	
	29	HUITEMA, IPv6: The New Internet Protocol, Oct. 28, 1997, Prentice-Hall, Upper Saddle River, NJ	
	30	DECASPER, Crossbow A Toolkit for Integrated Services over Cell Switched IPv6, IEEE, 1997, Zurich, CH/St. Louis, MO	
	31	MOSBERGER, Scout: A Path-Based Operating System, Dissertation submitted to Dept of Computer Science, 1997, University of Arizona	
	32	KRUPCZAK et al., Implementing Communication Protocols in Java, IEEE Communication Magazine, October 1998	
	33	FIUCZYNSKI et al., An Extensible Protocol Architecture for Application-Specific Networking, Department of Computer Science and Engineering, Seattle, WA	
	34	MUHUGUSA et al., COMSCRIPT*: An Environment for the Implementation of Protocol Stacks and their Dynamic Reconfiguration, 1994	

Examiner Signature	/Salman Ahmed/	Date Considered	03/28/2012
--------------------	----------------	-----------------	------------

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.
¹ Applicant's unique citation designation number (optional). ² Applicant is to place a check mark here if English language Translation is attached.
 This collection of information is required by 37 CFR 1.88. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

If you need assistance in completing the form, call 1-800-PTO-9199 (1-800-786-9199) and select option 2.

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /S.A./

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO INFORMATION DISCLOSURE STATEMENT BY APPLICANT (Use as many sheets as necessary)		<i>Complete if Known</i>			
		Application Number	New		
		Filing Date	February 13, 2012		
		First Named Inventor	6,629,163		
		Art Unit	3992		
		Examiner Name	Unknown		
Sheet	5	of	6	Attorney Docket Number	159291-0025(163)

NON PATENT LITERATURE DOCUMENTS			
Examiner Initials*	Cite No. ¹	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T ²
	36A	"First Amended Complaint and Demand for Jury Trial", filed Dec. 1, 2010, in Implicit Networks, Inc. v Juniper Networks, Inc., Case No. 3:10-cv-4234 SI, N.D. of California	
	36B	"Plaintiff's 1/10/2012 Amended Disclosure of Asserted Claims and Infringement Contentions", filed Jan. 10, 2012, in Implicit Networks v Juniper Networks, Case #3:10-cv-4234 SI	
	36C	"Implicit Networks, Inc., U.S. Patent No. 6,629,163 C1 Claims Chart, Implicit Networks, Inc. v Juniper Networks, Inc., Security Flow Based Processing"	
	36D	"Implicit Networks, Inc., U.S. Patent No. 6,629,163 C1 Claims Chart, Implicit Networks, Inc. v Juniper Networks, Inc., Application Acceleration and Optimization"	
	37A	"Plaintiff's Opening Claim Construction Brief" filed Nov. 28, 2011, in Implicit Networks, Inc. v Juniper Networks, Inc., Case No. 3:10-cv-4234 SI, in U.S. District Court	
	37B	"Plaintiff's Claim Construction Brief Pursuant to Patent L.R. 4-5(b), filed Dec. 12, 2011, in Implicit Networks, Inc. v Juniper Networks, Inc., Case No. 3:10-cv-4234 SI	
	37C	"Plaintiff's Reply to Defendant's Responsive Claim Construction Brief", filed Dec. 19, 2011, in Implicit Networks, Inc. v Juniper Networks, Inc., Case No. 3:10-cv-4234 SI	
	37D	Implicit Networks Technical Tutorial, Claim Construction Hearing, January 17, 2012	
	37E	Defendants' Technology Tutorial, Claim Construction Hearing, January 17, 2012	
	37F	2012-01-17 Technical Tutorial - Transcript of Proceedings, Case No. C 10-4234 SI, in U.S. District Court, Northern District of California	

Examiner Signature	/Salman Ahmed/	Date Considered	03/28/2012
--------------------	----------------	-----------------	------------

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.
 1 Applicant's unique citation designation number (optional). 2 Applicant is to place a check mark here if English language Translation is attached.
 This collection of information is required by 37 CFR 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

If you need assistance in completing the form, call 1-800-PTO-9199 (1-800-786-9199) and select option 2.

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /S.A./

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO INFORMATION DISCLOSURE STATEMENT BY APPLICANT (Use as many sheets as necessary)		<i>Complete if Known</i>	
		Application Number	New
		Filing Date	February 13, 2012
		First Named Inventor	6,629,163
		Art Unit	3992
		Examiner Name	Unknown
Sheet	6	of	6
		Attorney Docket Number	159291-0025(163)

NON PATENT LITERATURE DOCUMENTS			
Examiner Initials*	Cite No. ¹	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T ²
	37I	Transcript of Proceedings, Claim Construction Hearing - Jan. 18, 2012 Case No. C 10-4234 SI, in U.S. District Court, Norther District of California	
	37J	Transcript of Proceedings, Claim Construction Hearing - Jan. 19, 2012 Case No. C 10-4234 SI, in U.S. District Court, Norther District of California	
	37G	Implicit Networks, Inc., Claim Construction Hearing Slides - Jan. 18-19, 2012	
	37H	(Juniper Networks, Inc.) Claim Construction Presentation - Jan. 18-19, 2012	

Examiner Signature	/Salman Ahmed/	Date Considered	03/28/2012
--------------------	----------------	-----------------	------------

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.
¹ Applicant's unique citation designation number (optional). ² Applicant is to place a check mark here if English language Translation is attached.
 This collection of information is required by 37 CFR 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

If you need assistance in completing the form, call 1-800-PTO-9199 (1-800-786-9199) and select option 2.

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /S.A./