

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

JUNIPER NETWORKS, INC. and PALO ALTO NETWORKS, INC.,
Petitioners,

v.

PACKET INTELLIGENCE LLC,
Patent Owner.

In re *Inter Partes* Review of:
U.S. Patent Nos. 6,651,099, 6,665,725, 6,771,646, 6,839,751, and 6,954,789

**DECLARATION OF DR. JON B. WEISSMAN UNDER 37 C.F.R.
§ 1.68 IN SUPPORT OF PETITION FOR *INTER PARTES* REVIEW**

TABLE OF CONTENTS

I.	INTRODUCTION	1
	A. Professional Background	1
	B. Documents and Information Considered	4
	C. Summary of Opinions	5
II.	LEGAL STANDARDS TO BE APPLIED	8
III.	SUMMARY OF THE CHALLENGED PATENTS	10
	A. Technology Overview.....	10
	1. Network Protocols and Protocol Layering.....	10
	2. Network Packets	20
	3. Monitoring Network Traffic	21
	4. Control and Data Transmission in Network Protocols.....	23
	B. The '099 Patent Overview	26
	C. The '725 Patent Overview	32
	D. The '646 Patent Overview	35
	E. The '751 Patent Overview	37
	F. The '789 Patent Overview	38
	G. Prosecution History Overview	40
	1. The '099 Patent's Prosecution History	40
	2. The '725 Patent's Prosecution History	40
	3. The '646 Patent's Prosecution History	42
	4. The '751 Patent's Prosecution History	45
	5. The '789 Patent's Prosecution History	49

H.	Sandvine’s IPR Petitions.....	49
I.	Nokia’s IPR Petitions.....	51
J.	German Nullity Proceeding.....	53
IV.	SUMMARY OF THE PRIOR ART.....	57
A.	Riddle Overview.....	57
1.	Overview of Riddle.....	58
2.	Riddle’s Hardware Components	62
3.	Riddle’s Parsing of Packets.....	64
4.	Riddle’s Classifying Flows Based on Conversations.....	66
5.	Riddle’s Conversational Flow Analyzer.....	74
6.	Riddle’s Traffic Identification Based on RTP and RTSP	77
B.	Ferdinand Overview	83
C.	Yu Overview	87
D.	RFC1945 Overview	90
E.	Baker Overview	96
F.	Wakeman Overview	99
G.	Hasani Overview	104
V.	A PERSON OF ORDINARY SKILL IN THE ART	105
VI.	CLAIM CONSTRUCTION	108
A.	“Conversational Flow” / “Conversational Flow-Sequence”	108
B.	“State Of The Flow” / “State Of The Conversational Flow”	115
C.	“The Flow” / “New Flow” / “Existing Flow”	117
D.	“State Operations” / “State Processing Operations”	121

E.	“Flow-Entry Database ...” Terms	123
F.	“Parser Record”	126
G.	“Child Protocol”	127
H.	“Slicer”	127
I.	Means- and Step-Plus-Function Terms	128
VII.	THE CLAIMS OF THE ’099 PATENT ARE UNPATENTABLE	134
A.	For the ’099 Patent, Riddle in View of Ferdinand Renders Obvious Claims 1 and 2.....	135
1.	Reasons to Modify Riddle in View of Ferdinand	135
2.	Independent ’099 Claim 1	138
3.	Dependent ’099 Claim 2	226
B.	For the ’099 Patent, Riddle in View of Ferdinand and Further in View of Baker Renders Obvious Dependent Claims 4 and 5.....	229
1.	Reasons to Modify the Combination of Riddle and Ferdinand and Further in View of Baker	230
2.	Dependent ’099 Claim 4	233
3.	Dependent ’099 Claim 5	241
C.	For the ’099 Patent, Riddle in View of Ferdinand and Further in View of Yu Renders Obvious Claims 1 and 2.....	246
1.	Reasons to Modify the Combination of Riddle and Ferdinand Further in View of Yu.....	248
2.	Riddle in View of Ferdinand and Further in View of Yu Renders Obvious Identifying a “Conversational Flow-Sequence” and the Claimed State Tracking.....	249
D.	For the ’099 Patent, Riddle in View of Ferdinand and Baker and Further in View of Yu Renders Obvious Dependent Claims 4 and 5. 251	

E.	For the '099 Patent, Riddle in View of Ferdinand and Further in View of RFC1945 Renders Obvious Claims 1 and 2.....	253
1.	Reasons to Modify the Combination of Riddle and Ferdinand Further in View of RFC1945	258
2.	Riddle in View of Ferdinand and Further in View of RFC1945 Renders Obvious Identifying A “Conversational Flow-Sequence.”	262
F.	For the '099 Patent, Riddle in View of Ferdinand and Baker and Further in View of RFC1945 Renders Obvious Dependent Claims 4 and 5.	264
VIII.	THE CLAIMS OF THE '725 PATENT ARE UNPATENTABLE	265
A.	For the '725 Patent, Riddle in View of Baker Renders Obvious Claims 10, 12, 13, 16, and 17.....	266
1.	Reasons to Modify Riddle in View of Baker.....	266
2.	Independent '725 Claims 10 and 17	269
3.	Dependent '725 Claim 12	331
4.	Dependent '725 Claim 13	333
5.	Dependent '725 Claim 16	338
B.	For the '725 Patent, Riddle in View of Baker and Further in View of Yu Renders Obvious Claims 10, 12, 13, 16, and 17.....	339
C.	For the '725 Patent, Riddle in View of Baker and Further in View of RFC1945 Renders Obvious Claims 10, 12, 13, 16, and 17.....	341
IX.	THE CLAIMS OF THE '646 PATENT ARE UNPATENTABLE	342
A.	For the '646 Patent, Riddle in View of Ferdinand and Wakeman Renders Obvious Claims 1-3, 7, 16, and 18.	343
1.	Reasons to Modify Riddle in View of Ferdinand and Wakeman	343
2.	Independent '646 Claim 1	345

3. Dependent '646 Claim 2	380
4. Dependent '646 Claim 3	383
5. Independent '646 Claim 7	386
6. Independent '646 Claim 16.....	411
7. Dependent '646 Claim 18	416
B. For the '646 Patent, Riddle in View of Ferdinand and Wakeman and Further in View of Yu Renders Obvious Claims 1-3, 7, 16, and 18.....	419
C. For the '646 Patent, Riddle in View of Ferdinand and Wakeman and Further in View of RFC1945 Renders Obvious Claims 1-3, 7, 16, and 18.	421
X. THE CLAIMS OF THE '751 PATENT ARE UNPATENTABLE	423
A. For the '751 Patent, Riddle in View of Ferdinand Renders Obvious Claims 1, 2, 5, 10, 14, 15, and 17.....	424
1. Reasons to Modify Riddle in View of Ferdinand	424
2. Independent '751 Claim 1	424
3. Dependent '751 Claim 2	446
4. Dependent '751 Claim 5	451
5. Dependent '751 Claim 10	452
6. Dependent '751 Claim 14	455
7. Dependent '751 Claim 15	458
8. Independent '751 Claim 17	461
B. For the '751 Patent, Riddle in View of Ferdinand and Further in View of Yu Renders Obvious Claims 1, 2, 5, 10, 14, 15, and 17.....	465

C.	For the '751 Patent, Riddle in View of Ferdinand and Further in View of RFC1945 Renders Obvious Claims 1, 2, 5, 10, 14, 15, and 17.....	467
XI.	THE CLAIMS OF THE '789 PATENT ARE UNPATENTABLE	469
A.	For the '789 Patent, Riddle in View of Ferdinand Renders Obvious Claims 1, 2, 13-17, 19, 20, and 42	469
1.	Reasons to Modify Riddle in View of Ferdinand	470
2.	Independent '789 Claim 1	470
3.	Dependent '789 Claim 2	473
4.	Dependent '789 Claim 13	475
5.	Dependent '789 Claim 14	476
6.	Dependent '789 Claim 15	477
7.	Dependent '789 Claim 16	477
8.	Dependent '789 Claim 17	478
9.	Independent '789 Claim 19	480
10.	Dependent '789 Claim 20	488
11.	Dependent '789 Claim 42	488
B.	For the '789 Patent, Riddle in View of Ferdinand and Further in View of Baker Renders Obvious Dependent Claim 31	490
1.	Reasons to Modify the Combination of Riddle and Ferdinand Further in View of Baker	491
2.	Dependent '789 Claim 31	491
C.	For the '789 Patent, Riddle in View of Ferdinand and Further in View of Wakeman Renders Obvious Dependent Claims 33 and 34	492
1.	Reasons to Modify the Combination of Riddle and Ferdinand Further in View of Wakeman.....	492

2. Dependent '789 Claim 33	492
3. Dependent '789 Claim 34	493
D. For the '789 Patent, Riddle in View of Ferdinand and Hasani Renders Obvious Claims 44, 48, and 49.....	493
1. Reasons to Modify the Combination of Riddle and Ferdinand Further in View of Hasani.....	493
2. Independent '789 Claim 44.....	496
3. Dependent '789 Claim 48	506
4. Dependent '789 Claim 49	508
E. For the '789 Patent, Riddle in View of Ferdinand and Further in View of Yu Renders Obvious Claims 1-2, 13-17, 19-20, and 42.....	510
F. For the '789 Patent, Riddle in View of Ferdinand and Baker and Further in View of Yu Renders Obvious Claim 31.....	512
G. For the '789 Patent, Riddle in View of Ferdinand and Wakeman and Further in View of Yu Renders Obvious Claims 33-34.	514
H. For the '789 Patent, Riddle in View of Ferdinand and Hasani and Further in View of Yu Renders Obvious Claims 44 and 48-49.	516
I. For the '789 Patent, Riddle in View of Ferdinand and Further in View of RFC1945 Renders Obvious Claims 1-2, 13-17, 19-20, and 42.	518
J. For the '789 Patent, Riddle in View of Ferdinand and Baker and Further in View of RFC1945 Renders Obvious Claim 31.	519
K. For the '789 Patent, Riddle in View of Ferdinand and Wakeman and Further in View of RFC1945 Renders Obvious Claims 33-34. .	521
L. For the '789 Patent, Riddle in View of Ferdinand and Hasani and Further in View of RFC1945 Renders Obvious Claims 44 and 48- 49.	522

Exhibit	Description
1001	U.S. Patent No. 6,651,099 (“the ’099 Patent”)
1002	U.S. Patent No. 6,665,725 (“the ’725 Patent”)
1003	U.S. Patent No. 6,771,646 (“the ’646 Patent”)
1004	U.S. Patent No. 6,839,751 (“the ’751 Patent”)
1005	U.S. Patent No. 6,954,789 (“the ’789 Patent”)
1006	Declaration of Dr. Jon B. Weissman
1007	Curriculum vitae of Dr. Weissman
1008	U.S. Patent No. 6,412,000 (“Riddle”)
1009	PCT Publication WO 92/19054 (“Ferdinand”)
1010	RFC 1945 - Hypertext Transfer Protocol -- HTTP/1.0 (“RFC1945”)
1011	U.S. Patent No. 6,625,150 (“Yu”)
1012	Provisional Patent Application No. 60/112,859 (“the ’859 Provisional”)
1013	PCT Publication WO 97/23076 (“Baker”)
1014	U.S. Patent No. 5,740,175 (“Wakeman”)
1015	U.S. Patent No. 5,805,808 (“Hasani”)
1016	Provisional Patent Application No. 60/141,903 (“the ’903 Provisional”)
1017	File History for U.S. Patent No. 6,651,099
1018	File History for U.S. Patent No. 6,665,725
1019	File History for U.S. Patent No. 6,771,646
1020	File History for U.S. Patent No. 6,771,646 – February 10, 2004, Response to Office Action
1021	File History for U.S. Patent No. 6,839,751
1022	File History for U.S. Patent No. 6,954,789

Exhibit	Description
1023	Certified Translation of German Federal Patent Court Nos. 2Ni 26/16 (EP) and 2(Ni 46/16) (July 12, 2018)
1024	Provisional Patent Application No. 60/066,864 (“the ’864 Provisional”)
1025	Redline showing a comparison of Riddle to Provisional Patent Application No. 60/066,864
1026	Claim Chart comparing claims 1, 8, and 11 of Riddle to the specification of Provisional Patent Application No. 60/066,864
1027	U.S. Patent Application 08/977,642 (“Packer Application”)
1028	U.S. Patent Application 09/198,051 (“the ’051 Application”)
1029	U.S. Patent No. 5,802,106
1030	U.S. Patent No. 6,038,216
1031	U.S. Patent No. 6,046,980 (“Packer”)
1032	<i>PointCast Inc. is Testing a New Screen-Saver Product</i> , The Wall Street Journal (April 15, 1996)
1033	Gillin, Paul. <i>Editorial</i> , Computer World (May 13, 1996)
1034	Sneider, Daniel. <i>Redefining News in the Era of Internet By Blending Print and Television, Silicon Valley Start-up Shakes up Traditional View of News</i> , The Christian Science Monitor (June 26, 1996)
1035	PointCast Inc. 1998 SEC Filings
1036	U.S. Patent No. 6,807,558
1037	RFC 765 – File Transfer Protocol (“RFC765”)
1038	RFC 791 – Internet Protocol (“RFC791”)
1039	RFC 793 – Transmission Control Protocol (“RFC793”)
1040	RFC 1543 – Instructions to RFC Authors (“RFC1543”)
1041	RFC 2026 – The Internet Standards Process – Revision 3 (“RFC2026”)
1042	RFC 2616 – Hypertext Transfer Protocol – HTTP/1.1 (“RFC2616”)

Exhibit	Description
1043	International Standard ISO/IEC 7498 – Information Processing Systems – Open Systems Interconnection – Basic Reference Model – Part 4: Management Framework (Nov. 15, 1989)
1044	Internet Archive Affidavit for RFC1945
1045	Internet Archive Affidavit for RFC 1889 – RTP: A Transport Protocol for Real-Time Applications (“RFC1889”)
1046	Internet Archive Affidavit for RFC 2326 – Real Time Streaming Protocol (RTSP) (“RFC2326”)
1047	Chart comparing Yu to Provisional Patent Application No. 60/112,859
1048	Claim Chart comparing Yu’s claim 1 to the Provisional Patent Application No. 60/112,859
1049	<i>Sandvine Corp. v. Packet Intelligence, LLC</i> , No. IPR2017-00769, Paper No. 10 (Opposition to Request for Rehearing) (September 15, 2017)
1050	<i>Sandvine Corp. v. Packet Intelligence, LLC</i> , No. IPR2017-00450, Paper No. 6 (Preliminary Response) (April 28, 2017)
1051	<i>Sandvine Corp. v. Packet Intelligence, LLC</i> , No. IPR2017-00451, Paper No. 6 (Preliminary Response) (April 28, 2017)
1052	<i>Sandvine Corp. v. Packet Intelligence, LLC</i> , No. IPR2017-00629, Paper No. 6 (Preliminary Response) (April 28, 2017)
1053	<i>Sandvine Corp. v. Packet Intelligence, LLC</i> , No. IPR2017-00630, Paper No. 6 (Preliminary Response) (April 28, 2017)
1054	<i>Sandvine Corp. v. Packet Intelligence, LLC</i> , No. IPR2017-00769, Paper No. 6 (Preliminary Response) (April 28, 2017)
1055	<i>Sandvine Corp. v. Packet Intelligence, LLC</i> , No. IPR2017-00862, Paper No. 6 (Preliminary Response) (June 5, 2017)
1056	<i>Sandvine Corp. v. Packet Intelligence, LLC</i> , No. IPR2017-00450, Paper No. 8 (Decision) (July 26, 2017)
1057	<i>Sandvine Corp. v. Packet Intelligence, LLC</i> , No. IPR2017-00451, Paper No. 8 (Decision) (July 26, 2017)

Exhibit	Description
1058	<i>Sandvine Corp. v. Packet Intelligence, LLC</i> , No. IPR2017-00629, Paper No. 8 (Decision) (July 26, 2017)
1059	<i>Sandvine Corp. v. Packet Intelligence, LLC</i> , No. IPR2017-00630, Paper No. 9 (Decision) (July 26, 2017)
1060	<i>Sandvine Corp. v. Packet Intelligence, LLC</i> , No. IPR2017-00769, Paper No. 8 (Decision) (July 26, 2017)
1061	<i>Sandvine Corp. v. Packet Intelligence, LLC</i> , No. IPR2017-00862, Paper No. 8 (Decision) (July 26, 2017)
1062	<i>Sandvine Corp. v. Packet Intelligence, LLC</i> , No. IPR2017-00863, Paper No. 6 (Decision) (August 31, 2017)
1063	<i>Sandvine Corp. v. Packet Intelligence, LLC</i> , No. IPR2017-00863, Paper No. 8 (Notice of Abandonment) (Dec. 1, 2017)
1064	<i>Sandvine Corp. v. Packet Intelligence, LLC</i> , No. IPR2017-00863, Paper No. 9 (Adverse Judgment) (Dec. 20, 2017)
1065	<i>Nokia Corp. v. Packet Intelligence, LLC</i> , No. IPR2019-01289, EX1006 (Declaration of Dr. Kevin Jeffay)
1066	<i>Packet Intelligence LLC, v. NetScout Systems, Inc. et al.</i> , E.D. Tex. Case No. 2:16-CV-230-JRG, Docket Item 55-21 (Packet Intelligence Technology Tutorial) (January 20, 2017)
1067	<i>Packet Intelligence LLC, v. NetScout Systems, Inc. et al.</i> , E.D. Tex. Case No. 2:16-CV-230-JRG, Docket Item 66 (Claim Construction Memorandum and Order) (March 14, 2017)
1068	<i>Packet Intelligence LLC, v. NetScout Systems, Inc. et al.</i> , E.D. Tex. Case No. 2:16-CV-230-JRG, Docket Item 244 (Transcript of Proceedings held Oct. 10, 2017 AM Session) (October 17, 2017)
1069	<i>Packet Intelligence LLC, v. NetScout Systems, Inc. et al.</i> , E.D. Tex. Case No. 2:16-CV-230-JRG, Docket Item 250 (Transcript of Proceedings held Oct. 12, 2017 PM Session) (October 17, 2017)

Exhibit	Description
1070	<i>Packet Intelligence LLC, v. NetScout Systems, Inc. et al.</i> , E.D. Tex. Case No. 2:16-CV-230-JRG, Docket Item 314 (NetScout’s JMOL of No Infringement) (October 5, 2018)
1071	<i>Packet Intelligence LLC, v. NetScout Systems, Inc. et al.</i> , E.D. Tex. Case No. 2:16-CV-230-JRG, Docket Item 314-1 (Declaration of Michael Lyons) (October 5, 2018)
1072	<i>Packet Intelligence LLC, v. NetScout Systems, Inc. et al.</i> , E.D. Tex. Case No. 2:16-CV-230-JRG, Docket Item 314-4 (Excerpts of Russell Dietz’s Demonstrative Slides) (October 5, 2018)
1073	<i>Packet Intelligence LLC, v. NetScout Systems, Inc. et al.</i> , E.D. Tex. Case No. 2:16-CV-230-JRG, Docket Item 323-1 (Declaration of Steven Udick) (October 26, 2018)
1074	<i>Packet Intelligence LLC, v. NetScout Systems, Inc. et al.</i> , E.D. Tex. Case No. 2:16-CV-230-JRG, Docket Item 323-2 (Excerpts from Dr. Kevin Almeroth’s Direct Testimony Demonstrative Slides) (October 26, 2018)
1075	<i>Packet Intelligence LLC, v. NetScout Systems, Inc. et al.</i> , E.D. Tex. Case No. 2:16-CV-230-JRG, Docket Item 324-1 (Declaration of Sadaf R. Abdullah) (October 26, 2018)
1076	<i>Packet Intelligence LLC, v. NetScout Systems, Inc. et al.</i> , E.D. Tex. Case No. 2:16-CV-230-JRG, Docket Item 324-2 (Dr. Kevin Almeroth’s Rebuttal Testimony Demonstrative Slides) (October 26, 2018)
1077	<i>Packet Intelligence LLC, v. Ericsson Inc. et al.</i> , E.D. Tex. Case No. 2:18-CV-00381-JRG, Docket Item 74 (Joint Claim Construction and Prehearing Statement) (June 7, 2019)
1078	<i>Packet Intelligence LLC, v. Cisco Systems, Inc.</i> , E.D. Tex. Case No. 2:14-CV-252-JRG, Docket Item 89 (Packet Intelligence LLC’s Opening Claims Construction Brief) (January 26, 2015)
1079	<i>Palo Alto Networks, Inc. v. Packet Intelligence LLC</i> , N.D. Cal. Case No. 3:19-cv-02471, Joint Claim Construction and Prehearing Statement (December 17, 2019)

Exhibit	Description
1080	Patent Trial and Appeal Board Consolidated Trial Practice Guide (November 2019)
1081	Chart of third-parties' previously-proposed terms subject to §112(6) and corresponding structure
1082	Table Comparing Claims 1, 10, and 17 of the '725 Patent

I. INTRODUCTION

1. I, Jon B. Weissman, submit this declaration in connection with *inter partes* review (“IPR”) proceedings before the United States Patent and Trademark Office for U.S. Patent Nos. 6,651,099; 6,665,725; 6,771,646; 6,839,751; and 6,954,789 (the “Challenged Patents”).

2. I have been retained on behalf of Juniper Networks, Inc. and Palo Alto Networks, Inc. to offer technical opinions with respect to the Challenged Patents and prior art cited herein.

A. Professional Background

3. I am a Full Professor of Computer Science at the University of Minnesota, where I lead the Distributed Computing Systems Group. I received my B.S. in Applied Mathematics and Computer Science from Carnegie-Mellon, and my M.S. and Ph.D. in Computer Science from the University of Virginia. My curriculum vitae is attached as Exhibit 1007.

4. For my Ph.D. thesis, I developed the first automated scheduling system for parallel and distributed applications across heterogeneous local and wide-area networks. I thereafter worked as a software engineer for five years in the area of distributed systems.

5. In 1995, I returned to academia and began my career as a professor. My re-

search has been funded by NASA, the National Science Foundation, the Department of Energy, and the Air Force, and has included the following projects related to the subject matter of the Challenged Patents (i.e., network traffic monitoring and processing, and real-time systems):

- National Science Foundation, “Scaling the IoT with Constellation”;
- National Science Foundation, “Location, location, location (L3): Support for Geo-Centric Applications”;
- National Science Foundation, “One Thousand Points of Light: Accelerating Data-Intensive Applications By Proxy”;
- National Science Foundation, “A Data Mining and Exploration Middleware for Grid and Distributed Computing”;
- National Science Foundation, “Toward Community Services: Putting Parallel Network Services On-line”;
- National Science Foundation, “Resource Management for Parallel and Distributed Systems”; and
- Air Force Office of Scientific Research, “Telecommunication Networks for Mobile and Distributed Computing and Communications.”

6. I have published over 100 peer-reviewed technical articles, including some awarded or nominated for Best Paper at highly competitive international conferences. Many of my published papers relate to the subject matter of the Challenged Patents, including this small sample (and many more listed on my CV):

- “Rethinking Adaptability in Wide-Area Stream Processing Systems,”
Albert Jonathan, Abhishek Chandra, and Jon Weissman,

- 10th USENIX Workshop on Hot Topics in Cloud Computing;*
- “Nebula: Distributed Edge Cloud for Data Intensive Computing,” Albert Jonathan, Mathew Ryden, Kwangsung Oh, Abhishek Chandra, and Jon Weissman, *IEEE Transactions on Parallel and Distributed Systems*;
 - “TripS: Automated Multi-tiered Data Placement in a Geo-distributed Cloud Environment,” Kwangsung Oh, Abhishek Chandra, and Jon Weissman, *10th ACM International Systems and Storage Conference*;
 - “Redefining Data Locality for Cross-Data Center Storage,” Kwangsung Oh, Ajaykrishna Raghavan, Abhishek Chandra, and Jon Weissman, *The 2nd International Workshop on Software-Defined Ecosystems*;
 - “Passive Network Performance Estimation for Large-scale, Data-Intensive Computing,” Jinh Kim, Abhishek Chandra, and Jon B. Weissman, *IEEE Transactions on Parallel and Distributed Systems*;
 - “DDDAS/ITR: A Data Mining and Exploration Middleware for Grid and Distributed Computing,” Jon B. Weissman, Vipin Kumar, Varun Chandola, Eric Eilertson, Levent Ertoz, Gyorgy Simon, Seonho Kim, and Jinh Kim, *Workshop on Dynamic Data Driven Application Systems – DDDAS*;
 - “Scheduling Parallel Applications in Distributed Networks,” Jon B. Weissman and Xin Zhao, *Journal of Cluster Computing*;
 - “Adaptive Resource Scheduling for Network Services,” Byoung-Dai Lee and Jon B. Weissman, *IEEE 3rd International Workshop on Grid Computing*;
 - “Eliminating the Middle-Man: Peer-to-Peer Dataflow,” Adam Barker,

Jon B. Weissman, and Jano van Hemert, *17th IEEE International Symposium on High Performance Distributed Computing*; and

- “Optimizing Remote File Access for Parallel and Distributed Network Applications,” Jon B. Weissman, Mike Gingras, and Mahesh Marina, *Journal of Parallel and Distributed Computing*.

7. Additionally, I have served on the boards of several flagship journals, including IEEE Transactions on Parallel and Distributed Systems and IEEE Transactions on Computers. I am a member and former steering committee chair of the ACM International Symposium on High Performance Parallel and Distributed Systems, the flagship conference in my area. And I serve as an investigator for both the Center for Research in Intelligent Storage (sponsored by the National Science Foundation) and the Digital Technology Center.

8. I am being compensated at my standard consulting rate for my work on this declaration. My compensation is not dependent on and I have no financial interest in the outcome of these IPRs or any related litigation.

B. Documents and Information Considered

9. I have reviewed each of the Challenged Patents, including the claims of the patent in view of the specification. In addition, I have reviewed the Challenged Patents’ prosecution histories, the prior art discussed herein, and the remaining exhibits listed herein, as well as additional records from previous IPRs relating to the Challenged Patents. For convenience, portions of my declaration are based on the

declaration of Dr. Kevin Jeffay, submitted with IPR materials filed by Nokia.¹

C. Summary of Opinions

10. In my opinion, claims 1, 2, 4, and 5 of the '099 Patent are rendered obvious by the prior art. In the remainder of this declaration, I demonstrate that:

- Riddle in view of Ferdinand renders obvious claims 1 and 2 of the '099 Patent;
- Riddle in view of Ferdinand and further in view of Baker renders obvious claims 4 and 5 of the '099 Patent;
- Riddle in view of Ferdinand and further in view of Yu renders obvious claims 1 and 2 of the '099 Patent;
- Riddle in view of Ferdinand and Baker and further in view of Yu renders obvious claims 4 and 5 of the '099 Patent;
- Riddle in view of Ferdinand and further in view of RFC1945 renders obvious claims 1 and 2 of the '099 Patent; and
- Riddle in view of Ferdinand and Baker and further in view of RFC1945 renders obvious claims 4 and 5 of the '099 Patent.

11. In my opinion, claims 10, 12, 13, 16, and 17 of the '725 Patent are rendered obvious by the prior art. In the remainder of this declaration, I demonstrate that:

- Riddle in view of Baker renders obvious claims 10, 12, 13, 16, and 17 of the '725 Patent;
- Riddle in view of Baker and further in view of Yu renders obvious claims 10, 12, 13, 16, and 17 of the '725 Patent; and
- Riddle in view of Baker and further in view of RFC1945 renders obvious

¹ Ex. 1065.

claims 10, 12, 13, 16, and 17 of the '725 Patent.

12. In my opinion, claims 1, 2, 3, 7, 16, and 18 of the '646 Patent are rendered obvious by the prior art. In the remainder of this declaration, I demonstrate that:

- Riddle in view of Ferdinand and Wakeman renders obvious claims 1-3, 7, 16, and 18 of the '646 Patent;
- Riddle in view of Ferdinand and Wakeman and further in view of Yu renders obvious claims 1-3, 7, 16, and 18 of the '646 Patent
- Riddle in view of Ferdinand and Wakeman and further in view of RFC1945 renders obvious claims 1-3, 7, 16, and 18 of the '646 Patent.

13. In my opinion, claims 1, 2, 5, 10, 14, 15, and 17 of the '751 Patent are rendered obvious by the prior art. In the remainder of this declaration, I demonstrate that:

- Riddle in view of Ferdinand renders obvious claims 1, 2, 5, 10, 14, 15, and 17 of the '751 Patent;
- Riddle in view of Ferdinand and further in view of Yu renders obvious claims 1, 2, 5, 10, 14, 15, and 17 of the '751 Patent; and
- Riddle in view of Ferdinand and further in view of RFC1945 renders obvious claims 1, 2, 5, 10, 14, 15, and 17 of the '751 Patent.

14. In my opinion, claims 1, 2, 13-17, 19, 20, 31, 33, 34, 42, 44, 48, and 49 of the '789 Patent are rendered obvious by the prior art. In the remainder of this declaration, I demonstrate that:

- Riddle in view of Ferdinand renders obvious claims 1, 2, 13-17, 19, 20, and 42 of the '789 Patent;

- Riddle in view of Ferdinand and further in view of Yu renders obvious claims 1, 2, 13-17, 19, 20, and 42 of the '789 Patent;
- Riddle in view of Ferdinand and further in view of RFC1945 renders obvious claims 1, 2, 13-17, 19, 20, and 42 of the '789 Patent;
- Riddle in view of Ferdinand and further in view of Baker renders obvious claim 31 of the '789 Patent;
- Riddle in view of Ferdinand and Baker and further in view of Yu renders obvious claim 31 of the '789 Patent;
- Riddle in view of Ferdinand and Baker and further in view of RFC1945 renders obvious claim 31 of the '789 Patent;
- Riddle in view of Ferdinand and further in view of Wakeman renders obvious claims 33 and 34 of the '789 Patent;
- Riddle in view of Ferdinand and Wakeman and further in view of Yu renders obvious claims 33 and 34 of the '789 Patent;
- Riddle in view of Ferdinand and Wakeman and further in view of RFC1945 renders obvious claims 33 and 34 of the '789 Patent;
- Riddle in view of Ferdinand and further in view of Hasani renders obvious claims 44, 48, and 49 of the '789 Patent;
- Riddle in view of Ferdinand and Hasani and further in view of Yu renders obvious claims 44, 48, and 49 of the '789 Patent; and
- Riddle in view of Ferdinand and Hasani and further in view of RFC1945 renders obvious claims 44, 48, and 49 of the '789 Patent.²

² Throughout my declaration, “the Challenged Claims” refers to all the analyzed claims in the '099, '725, '646, '751, and '789 Patents. After review of the documents and materials cited herein, I have not identified any evidence of secondary

15. In this declaration, I also offer my opinion on how a person of ordinary skill in the art (“POSITA”) would have interpreted certain claim terms recited in the Challenged Patents at the time of the purported invention. I understand that the earliest possible priority date for the Challenged Patents is June 30, 1999.

II. LEGAL STANDARDS TO BE APPLIED

16. Counsel has advised me of legal concepts that are relevant to IPRs, and I have applied these concepts in reaching my opinions in this declaration.

Claim Language

17. I understand that, during IPR, patent claim terms are given their ordinary and customary meaning to a POSITA in view of the specification and prosecution history, unless those sources show an intent to depart from such meaning.

U.S. Patent’s Priority to an Earlier Filing Date

18. I understand that a U.S. patent has an effective prior art date under pre-AIA 35 U.S.C. §102(e) based on the filing date of an earlier-filed patent application if the patent’s relevant subject matter is described in the earlier-filed application, and at least one of the patent’s claims is supported by the written description of the earlier-filed application in compliance with pre-AIA 35 U.S.C. §112, first paragraph.

considerations supporting non-obviousness of the Challenged Claims that would change my opinion that the Challenged Claims are obvious over the prior art. If and when presented, I will address any evidence and arguments that Patent Owner may raise regarding secondary considerations supporting non-obviousness.

Obviousness

19. I understand that prior art references can render a patent claim obvious to a POSITA if the differences between the claimed subject matter and the prior art are such that the claimed subject matter would have been obvious at the time of the claimed invention. In analyzing obviousness, it is important to consider the scope of the claims, the level of skill in the relevant art, the scope and content of the prior art, the differences between the prior art and the claims, and any secondary considerations of non-obviousness (such as commercial success, long-felt but unsolved needs, failure of others, unexpected results, copying, skepticism of experts, industry praise, significant effort or high cost to develop, near simultaneous invention, industry acceptance via licensing), if present.

20. I further understand that, when the claimed subject matter combines preexisting elements to yield no more than what one would have expected from such an arrangement, it is obvious. So in assessing obviousness, one must consider whether the claimed improvement is more than the predictable use of prior art elements according to their established functions. While a motivation to combine or modify the prior art must exist, and an obviousness conclusion cannot rely on hindsight, there need not be a precise teaching in the prior art directed to the specific claimed subject matter. Rather, one can take account of the inferences and creative steps

that a POSITA—a person of ordinary creativity, not an automaton—would employ. Additionally, neither the motivation nor the avowed purpose of the inventor(s) controls this inquiry, since any need or problem known in the field at the time of the invention can provide a reason for combining elements. And when there is a reason to solve a problem with a finite number of predictable solutions, a POSITA has reason to pursue the known options within his or her technical grasp. For example, I understand that it is important to consider whether there existed at the time of the invention a known problem for which there was an obvious solution, as well as design incentives and other market forces that prompt variations in the same or other fields.

III. SUMMARY OF THE CHALLENGED PATENTS

A. Technology Overview

1. Network Protocols and Protocol Layering

21. When one computer communicates with another computer over a network, hardware and software on the source computer will create and transmit a message as a series of one or more data units to the destination computer. Network devices (including the source and destination computers) process these data units according to rules that facilitate communications over the network. These rules, grouped together, define a protocol for an aspect of the communication. To communicate

messages between a source and a destination computer, multiple protocols, operating in concert, are required.

22. These protocols are organized hierarchically as a series of hardware and software “layers” and are colloquially referred to as a “protocol stack.” The layers are numbered from the lowest, most basic or primitive protocol layer, to the highest, most functional protocol layer. Each layer is responsible for providing a discrete communication service that builds upon the service(s) provided by the lower layer(s) to provide a more functional, full-featured communication service.

23. Historically, the two most dominant models of protocol layers are the OSI (Open Systems Interconnect) model (sometimes referred to as the ISO model) and the TCP/IP protocol suite. As illustrated below in Figure 1, the OSI model defines a seven-layer protocol stack.³ Whereas the TCP/IP protocol suite defines a simpler five-layer protocol stack, which was the predominant protocol stack in the networking and distributed systems communities by the mid-1990’s.

³ In Sections III’s technology and patent overviews, I include original figures (such as Figure 1) that I directed and oversaw the creation of.

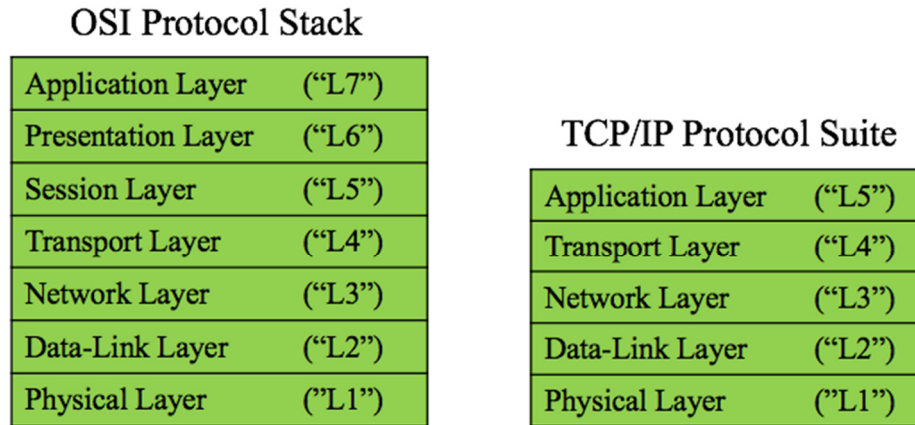


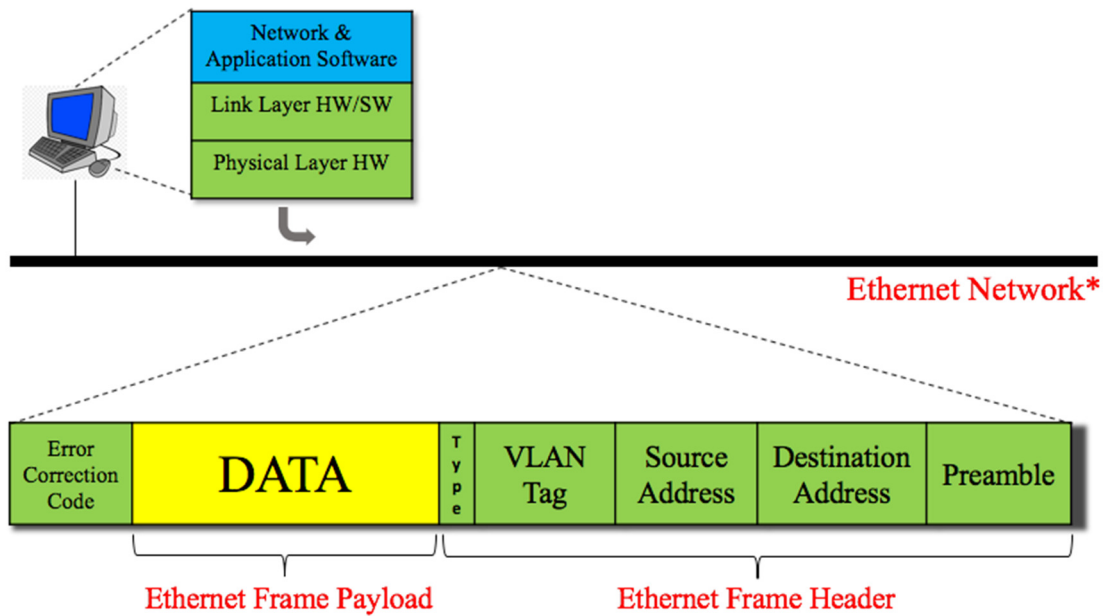
Figure 1: OSI & TCP/IP Protocol Stacks

24. In both models, the lowest layer, layer-1 or L1, is the physical layer. The physical-layer protocol is typically implemented exclusively in hardware and is concerned with the low-level details of transmitting binary data (1s and 0s) over a physical medium (fiber-optic cables, copper twisted-pair wiring, radio-frequency spectrum, etc.). Different physical media require different physical-layer protocols because the process of transmitting binary data on each medium is different.

25. The next layer, layer-2 or L2, is the data-link layer, or simply the link layer. The data-link layer defines a transmission structure, typically called a frame, and is responsible for transmitting frames between computers on the same network. The most common example of a data-link layer protocol is the Ethernet protocol. The data-link layer uses the services of a physical-layer protocol to transmit the binary data of a frame on the network's physical medium to a destination on the network.

26. A frame consists of a header and a payload. The payload is the actual data

being communicated across the network and may have additional, internal structure as described below. The header contains, among other things, a link layer address of the computer generating the frame (a source address) and a link layer address of the destination computer of the frame (a destination address). The link layer addresses in the header of a frame (shown below as the “Source Address” and “Destination Address” of Figure 2) are often referred to as hardware addresses, or Media Access Control (“MAC”) addresses.



*Not to scale

Figure 2: Ethernet Frame Structure & Major Components

27. Data-link-layer protocols transmit frames between two computers on the same local-area network (“LAN”). LANs are created by a type of interconnection device called a switch, bridge, or access point, which individual computers connect to via some physical medium (e.g., the ubiquitous blue Ethernet cable). The typical

interconnection device is capable of interconnecting a large number of computers.

28. When a computer generates and transmits a data-link-layer frame, the frame is received by the interconnection device. The device examines the destination data-link layer address in the frame to determine the destination of the frame.

Based on the data-link layer destination address carried in the frame, the frame is forwarded to the appropriate destination. “Forwarding” refers to the process used by a data-link layer network interconnection device to transmit a frame to (or towards) its ultimate destination.

29. LANs are limited in terms of both the number of computers that can be on a LAN as well as the geographic area served by the LAN. The physical media used to build the LAN and the physical-layer protocol used by the LAN constrain the size of the LAN. The type of interconnection device used to realize the LAN may also constrain the size of the LAN.

30. LANs can be interconnected via a type of interconnection device called a router. Normally, two computers on two LANs interconnected via a router cannot exchange messages with each other using only a data-link-layer protocol because the addresses in the data-link-layer frame’s header are only known on the LAN on which the frame was generated. That is, a computer on one LAN cannot transmit data to a computer on another LAN using just a data-link-layer protocol because

the source computer cannot address a frame to the destination computer. To transmit data between two computers on different LANs in an internetwork, a higher-layer protocol is required.

31. The next layer in the protocol stack, layer-3 or L3, is the network layer. Network-layer protocols solve the problem of communication between LANs by defining a new transmission structure, called a datagram, and by defining a new addressing scheme that enables computers to address other computers anywhere in a network of interconnected LANs. Like a frame, a datagram consists of a header and a payload. The datagram header contains a new type of address, a network-layer address, for routing datagrams between LANs. Thus, “routing” refers to the process of using network-layer destination addresses at an interconnection device to transmit datagrams and determine the networks onto which the datagrams should be transmitted. The most common network-layer protocol today is the Internet Protocol (IP), in which case network-layer addresses are called IP addresses. Figure 3 provides an illustration of the IP datagram structure and components.

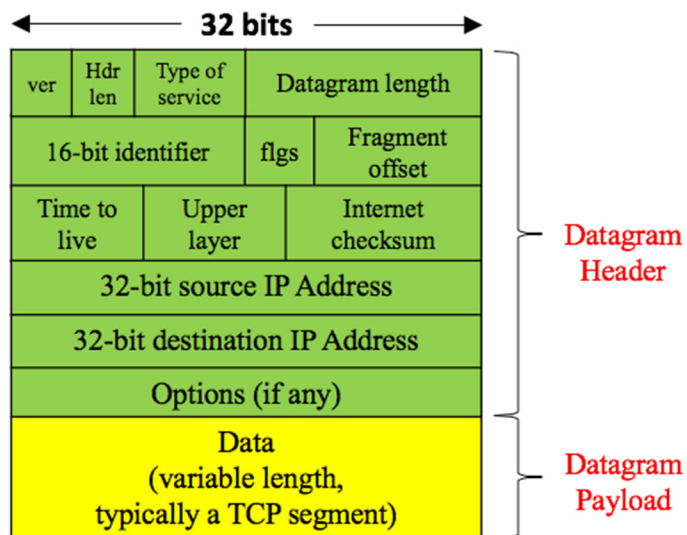


Figure 3: IP Datagram Structure & Major Components
 Showing Datagram as Series of 32-bit Words

32. Network-layer protocols can deliver datagrams between two computers on different LANs. But as datagrams transit those networks between source and destination, they are subject to a number of pathologies that can hinder communication or even make it impossible. These pathologies include the deletion, loss, reordering, or creation of duplicate datagrams.

33. Protocols in the next layer in the protocol stack—the layer-4, L4, or transport-layer protocols—build upon network-layer protocols to provide a communication service that ameliorates some or all of these pathologies. For example, in networks, such as the Internet, that use the IP network-layer protocol, the most common transport-layer protocol is the Transmission Control Protocol (“TCP”).

The TCP transport-layer protocol provides, among other things, a reliable, in-order

data delivery service that monitors transmissions to detect when datagrams have been lost and retransmits copies until they have been successfully received.

34. As with other layers, the transport layer defines a new transmission structure called a segment that is comprised of a new header and payload. In the case of TCP, the header includes control information such as a sequence number for the segment. The sequence number is used by the destination computer to detect and reorder out-of-order segments, and to detect lost segments. In the header, the source computer corresponds to the source IP address carrying the TCP segment. And in the header, the destination computer corresponds to the destination IP address. Figure 4 illustrates the structure and components of a TCP segment.

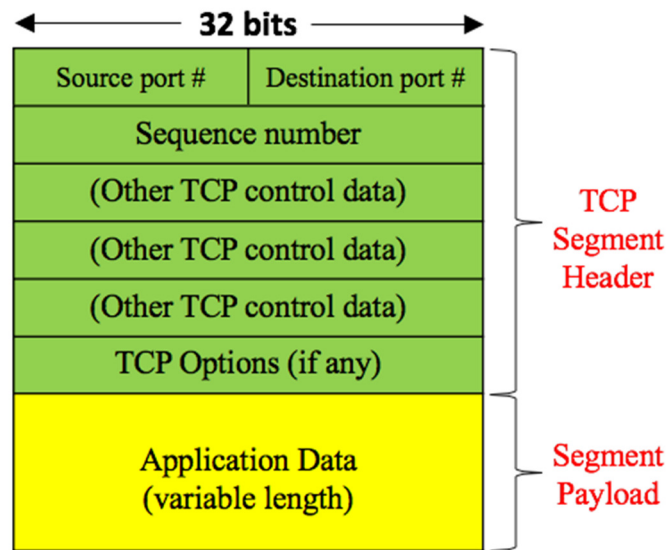


Figure 4: TCP Segment Structure & Major Components
Showing Segment as Series of 32-bit Words

35. The TCP header also contains port numbers that act like addresses identifying which underlying communication protocols, such as HTTP, Telnet, and SMTP, are associated with the transmitted datagram. In a TCP segment, the header contains two port numbers: a source port number and a destination port number. The source port number identifies which underlying communication protocol generated the datagram on the source computer. And the destination port number identifies which communication protocol should be associated with the datagram on the destination computer.

36. Port numbers can range from 0 to 65,535. But to make it easier to develop clients and servers for various protocols, the Internet Assigned Numbers Authority (IANA) has assigned port numbers between 0 and 1,023 for use by well-known protocols. For example, port 80 is reserved for the HTTP (web) protocol, so by default all web traffic is transmitted to web servers in TCP segments with port 80 as the destination port number. Similarly, port 25 is reserved for the SMTP (email) protocol, port 53 for the Domain Name System protocol (discussed below), port 21 for the FTP protocol, port 22 for the SSH protocol, and port 443 for the HTTPS protocol. By convention, programmers only use reserved port numbers for communications those numbers are associated with.

37. Combined, the IP source address, IP destination address, source port number, destination port number, along with an identifier of the transport protocol in

use (e.g., TCP), uniquely identify all the datagrams exchanged during a communication session between a source computer (e.g., a client) and a destination computer (e.g., a server). This 5-tuple, called a flow id, has been used to identify datagrams within a single end-to-end (original sender to ultimate destination) communication session since the creation of the Internet. All five components of the flow id are required to determine the communication session to which a datagram belongs.

38. The next and highest layer in the TCP/IP protocol suite model is the application layer. Application-layer protocols also define a new transmission structure, simply called a message. The message is typically comprised of a new header and payload. The contents of the header vary by application-layer protocol but typically include addresses and control information relevant to the application (and hence are application protocol-specific).

39. One common application-layer protocol is the Hypertext Transfer Protocol (“HTTP”) that is used to request and receive data from web servers. In the case of HTTP, the header contains an indication of the content that is being requested (the “address” of the content on the server) as well as other control information. The body of an HTTP message contains data such as the data for a requested web page. Two exemplary HTTP messages are (a) the HTTP request message sent from a client such as a web browser to a server to request content from that server, and (b)

the HTTP response message sent from a web server back to a client in response to previous request.

40. The seven-layer OSI reference model illustrated in Figure 1 has two additional layers before the application layer, the layer-5 session layer and the layer-6 presentation layer. The session layer controls the connections between computers, and the presentation layer establishes context between application-layer entities. The TCP/IP protocol suite does not include the session and presentation layers. Instead, the TCP/IP protocol suite assumes that the services associated with the session and presentation layers are built into the application if they are needed.

2. Network Packets

41. Layers two and higher within the protocol stack each defines a unit of data transmission and its structure. The foregoing has purposefully used different terms for the transmission units in the protocol stack in the TCP/IP protocol suite model: data-link layer transmission units are frames, network-layer units are datagrams, transport-layer units are segments, and higher-layer units are messages. These are all terms of art that networking professionals used at the time of the invention of the Challenged Patents when trying to be precise in their descriptions of network operations, as illustrated in Figure 5 below.

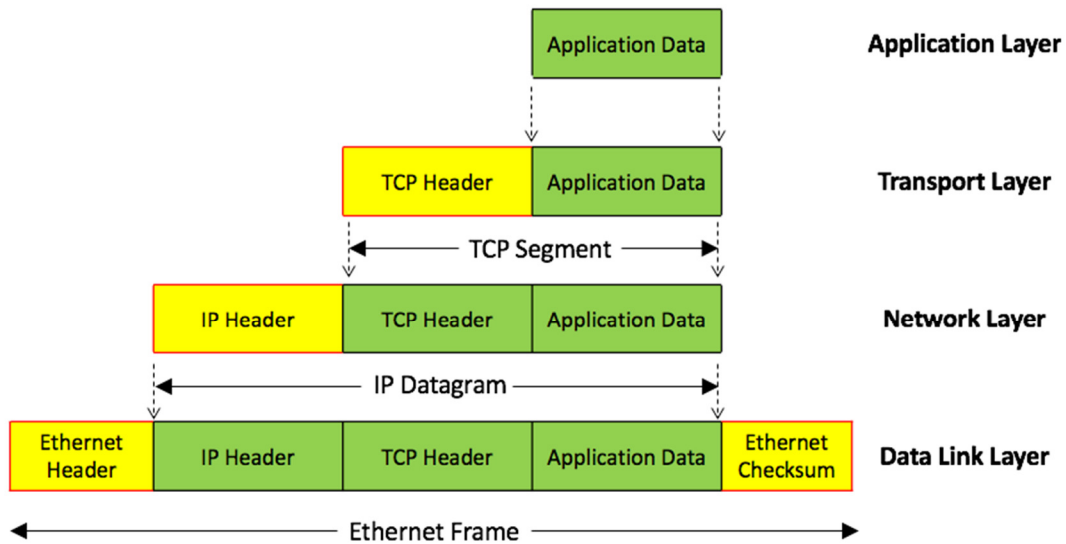


Figure 5: Layer Information for TCP/IP Protocol Suite Transmission

42. Nonetheless, networking professionals were (and are) often less precise and frequently used the term “packet” to refer generically to transmission units within the protocol stack. Thus, while it could be confusing, it was not uncommon for a POSITA to refer to link-layer transmission units as “Ethernet packets” instead of the more precise “Ethernet frames,” network-layer transmission units as “IP packets” instead of the more precise “IP datagrams,” etc. Given this informal use of “packet,” one must always consider the word’s context to understand exactly which transmission unit is being referred to.

3. Monitoring Network Traffic

43. For the technology involved in the Challenged Patents, IP datagrams are in one sense “received” only by the computer associated with the destination IP address in the datagram’s header. However, in order to deliver the IP datagram to its

destination, as described above, various data-link-layer frames containing the IP datagram will be received by intermediate devices such as routers, where the datagram will be examined and forwarded in a new data-link-layer frame to either the final destination or to another router along the path towards it. In the latter case, the network-layer IP datagram is not addressed to the router, but the router “receives” the IP datagram because the data-link-layer frame containing the datagram is addressed to the router.

44. Beyond routers, it is possible for other devices in the network to receive IP datagrams, as well as data-link-layer frames, that are not addressed to the device. That is, as a data-link-layer frame transits a LAN, a device on the LAN can receive the frame even though the frame carries a different data-link-layer destination address. This is possible because, for example, certain data-link-layer protocols (e.g., Ethernet) allow a computer to operate its network interface in so-called “promiscuous”-mode wherein the computer can receive copies of any and all frames that transit its interface. The computer can then do anything with the frame it desires, including parse them to learn which higher-layer protocols (if any) are being used and even modify any data field of any determined higher-layer protocol.

45. A common application of promiscuous-mode reception is for network monitoring. A device generically known as a network monitor is placed in a network such that it can “see” all network traffic on a network link. The device intercepts

frames and examines them to determine the source computer, the communication session to which the frame belongs, and/or the application generating the frame. To determine such information, the monitor typically needs knowledge of various protocols above the data-link-layer protocol. The monitor then can maintain various statistics concerning the computer/session/application to which frames belong such as the total number of bytes or packets transmitted.

46. An in-line monitoring device is often referred to as a “man-in-the-middle” device. Depending on the operations it performs on intercepted frames, its presence may or may not be discernable to the original source, the next receiver, or the ultimate destination. If its presence is not discernable to sources or receivers, the man-in-the-middle is said to be “transparent.” An important aspect of ensuring that a monitoring device is transparent is for the device to be able to intercept and process frames in real-time. Ideally, this means that in all cases the device can intercept and process a frame before the next frame arrives.

4. Control and Data Transmission in Network Protocols

47. Most application-layer protocols need to communicate both application data and control data. The former typically relates to data the user has requested or that is required for the user to take an action, whereas the latter relates to data required for the application to perform its function. Protocols such as HTTP transmit application and control data on the same network connection between the client and

server with the control data in the header of an application-layer message (e.g., an “HTTP GET” message) and the application data in the body or payload.

48. But for a variety of reasons, other application-layer protocols establish separate network connections for transmission of application data and control data. The best-known example of such a protocol is the file transfer protocol FTP. The FTP protocol was standardized in 1980.⁴ In the FTP protocol, a client will establish at least two connections to a server: one for communicating commands to control the file transfer and to send data about the state of the transfer, and the second to transfer the actual file.⁵ An example of the two disjointed TCP connections used for FTP is shown below with one TCP connection on port 21 for control and one TCP connection on port 20 for transport (data).

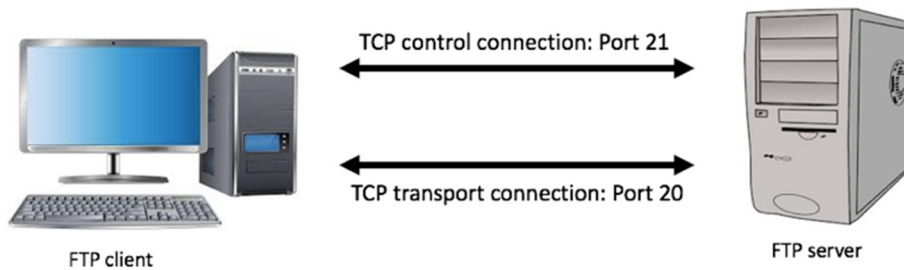


Figure 6: FTP Connections Between Server & Client

49. Certain transport protocols may also use separate connections for control and data transmission. The Real-Time Transport Protocol (“RTP”) is a transport

⁴ Ex. 1037 (RFC765 – File Transfer Protocol).

⁵ Ex. 1037 (RFC765 – File Transfer Protocol), 6-7.

protocol (commonly implemented as an application-layer protocol) used to carry real-time data such as audio and video data in multimedia applications such as a video or audio conference.⁶ In the case of RTP, a separate protocol called RTCP (“Real-Time Control Protocol”) is used to convey a variety of control and status information between conference participants.⁷ Since these protocols may convey information relating to the same application, RTP defines a session identifier that can be used to associate RTP flows and RTCP flows.⁸ For certain uses of RTP, multiple RTCP flows may exist, in which case the session identifier enables a receiver (or a network monitor) to understand which control flows relate to which data flows. Notably, the RTP specification (discussed below) defines devices that may exist in the network that are neither the source nor final destination for multimedia data but exist to monitor the performance of RTP flows (by receiving and processing the appropriate RTCP flows).

50. Strictly speaking, RTCP is not tied to RTP and can be used with other protocols that seek to transmit data in real-time, such as the Real-Time Streaming Protocol (“RTSP”) discussed in more detail below.⁹ RTCP can be used with RTSP to

⁶ RFC 1889 - RTP: A Transport Protocol for Real-Time Applications (“RFC1889”), 1.

⁷ RFC1889, 3-4.

⁸ RFC1889, 8.

⁹ RFC 2326 - Real Time Streaming Protocol (RTSP) (“RFC2326”).

create a streaming session where RTSP carries the media data and RTCP carries the control data for the session.¹⁰ RTSP, like RTP, enables a receiver (or network monitor) to associate an RTCP flow with an RTSP flow. An example of RTSP, RTP, and RTCP working together is shown below.¹¹

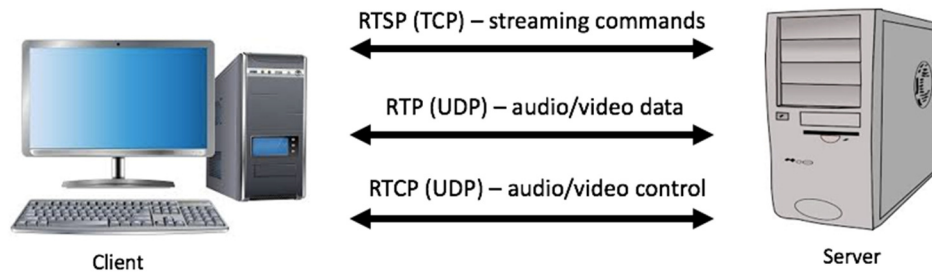


Figure 7: Real-Time Streaming Connections Between Server & Client

B. The '099 Patent Overview

51. The '099 Patent discloses a packet monitor for examining packets passing through a connection point on a computer network to determine whether a packet is of an existing flow.¹²

52. According to the '099 Patent, some prior-art packet monitors classified packets into “connection flows,” while the patent seeks to classify packets into “conversational flows.” In the '099 Patent’s words:

The term “connection flow” is commonly used to describe all the packets involved with a single connection. A conversational flow, on the other

¹⁰ RFC2326, 1.

¹¹ RFC2326, 12-13.

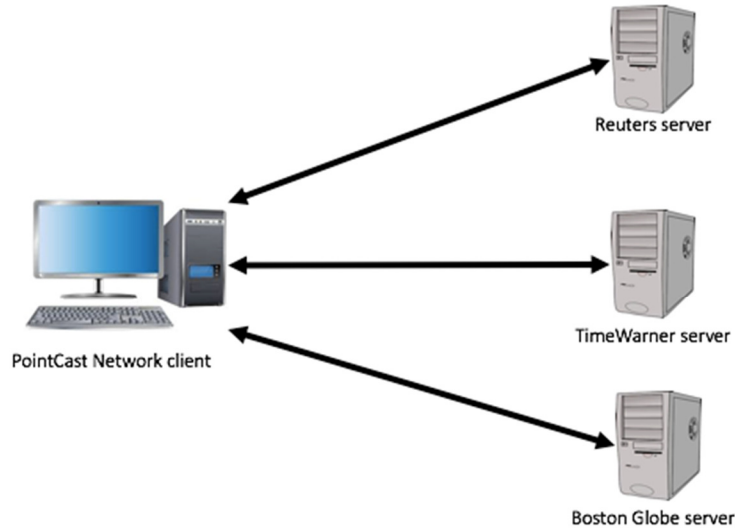
¹² '099 Patent, Abstract.

hand, is the sequence of packets that are exchanged in any direction as a result of an activity—for instance, the running of an application on a server as requested by a client.¹³

53. All the Challenged Patents, including the '099 Patent, incorporate by reference Provisional Patent Application 60/141,903 (“the '903 Provisional”), which disclosed how “PointCast” traffic created a conversational flow.¹⁴ PointCast was a screensaver program that would retrieve news, sports scores, stock quotes, weather, horoscopes, and other information from different sources on the internet to display when the computer was idle. The below exemplary figure illustrates PointCast running on a client computer and connecting to a Reuters server, a Time-Warner Server, and a Boston Globe server to retrieve information to display on the screensaver.

¹³ '099 Patent at 2:34–40.

¹⁴ E.g., '099 Patent, 1:6-11.



PointCast Network Connections Between Client & Servers

54. Thus, as Patentee’s ’903 Provisional put it, PointCast would open multiple conversations that “packet-by-packet ... might look like separate flows to prior art monitors. However, each of these connections is merely a sub-flow under the PointCast master flow.”¹⁵

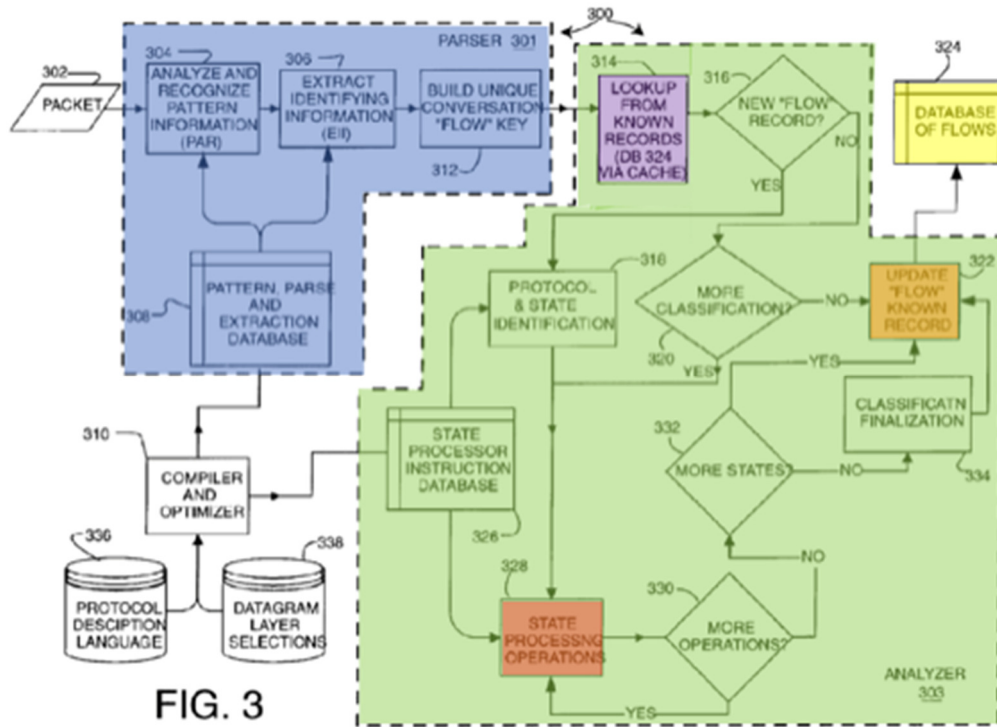
55. And according to the ’099 Patent, “[w]hat distinguishes [its claimed] invention from prior art network monitors is that it has the ability to recognize disjointed flows as belonging to the same conversational flow.”¹⁶

56. The ’099 Patent’s Figure 3, “a functional block diagram of a process embodiment of the present invention,” shows a network packet monitor 300 that includes parser 301 (blue), flow-entry database 324 (yellow), and analyzer 303 (green) with

¹⁵ Ex. 1016 (Provisional Patent Application No. 60/141,903 (“’903 Provisional”)), 7:16-25.

¹⁶ ’099 Patent, 3:48–51.

lookup engine 314 (purple), state processor 328 (red), and flow-entry updater 322 (orange), as annotated below:¹⁷



57. When monitoring packets, the monitor receives a packet 302 and determines whether the packet is part of a new flow or part of an existing flow recorded in flow-entry database 324.¹⁸ To do so, the monitor’s parser 301 parses and extracts portions of packet 302 to generate an identifying “signature” that is used to recognize the packet’s flow.¹⁹

58. The monitor’s compiler 310 sends protocol-specific information, such as

¹⁷ ’099 Patent, 11:43–45, 21:24-25:37.

¹⁸ ’099 Patent, 13:54-61, 14:14-18.

¹⁹ ’099 Patent, 11:59-65; ’646 Patent, 6:47-54; 11:59-62; 12:7-9.

packet sender and recipient, to parser 301 to identify the packet's flow.²⁰ Using the protocol-specific information, parser's pattern-structures-and-extraction-operations database 308 provides the packet's parsing-pattern-structures.²¹ Based on the parsing-pattern-structures, parser 301 parses packet 302 via pattern recognition process 304 to determine the protocol types and associated headers for each protocol layer that exists in packet 302.²²

59. The parser's extraction process 306 extracts characteristic information from packet 302 using extraction masks that depend on the protocols used in the packet.²³ Pattern-structures-and-extraction-operations database 308 supplies the extraction masks.²⁴ At block 312, the parser processes the packet's extracted characteristic information to build a unique flow signature (also called a "key") for the packet's flow.²⁵ The flow signature includes a hash of the signature.²⁶ The monitor records this information, which includes the signature, the hash, and the packet itself, as a parser record.²⁷

²⁰ '099 Patent, 11:66–12:8.

²¹ '099 Patent, 12:12–22, 12:65–13:2.

²² '099 Patent, 12:12–22, 12:65–13:2.

²³ '099 Patent, 12:12–22, 13:14–25.

²⁴ '099 Patent, 12:12–22, 13:14–25.

²⁵ '099 Patent, 13:20-23, Fig. 3.

²⁶ '099 Patent, 13:30–36.

²⁷ '099 Patent, 13:30-47.

60. Then, the monitor's analyzer 303 examines the parser record. The analyzer's lookup engine 314 determines whether the examined packet belongs to a new flow or a known flow by comparing the parser record to flow-entries stored in database 324.²⁸ Flow-entry database 324 "stores flow-entries that include the unique flow-signature, state information, and extracted information from the packet for updating flows," and statistics about flows.²⁹ If there is no flow-entry matching the signature in the parser record, then protocol-and-state-identification process 318 determines the packet's protocols and where in the state sequence for a flow for the protocol the packet belongs, by referencing database 326 of state patterns and processes.³⁰ If the packet is found to have a matching flow-entry in database 324, then process 320 determines if state processor 328 needs to further classify the flow-entry's signature.³¹ If there is no need for further classification, then process 322 updates the flow-entry in flow-entry database 324.³² If state processing is needed, then state processor 328 performs state operations according to state instructions from state-pattern-and-processes database 326.³³

²⁸ '099 Patent, 13:54–61, 14:3–13.

²⁹ '099 Patent, 14:14–18.

³⁰ '099 Patent, 14:39–46.

³¹ '099 Patent, 14:49–53.

³² '099 Patent, 14:53–54.

³³ '099 Patent, 14:58–62.

61. State processor 328 analyzes both new and existing flows in order to analyze all levels of the protocol stack, ultimately classifying flows by application (level 7 in the OSI model).³⁴ State processor 328 processes from state-to-state based on predefined state transition rules and state operations specified in state-pattern-and-processes database 326.³⁵

62. By maintaining a state of flows, monitor 300 provides for a “single packet protocol recognition of flows,” and a “multiple-packet recognition of flows.”³⁶ The analyzer’s process 334 finalizes the classification of the flow.³⁷

C. The ’725 Patent Overview

63. The ’725 Patent discloses a method of performing protocol-specific operations on a packet passing through a connection point on a computer network.³⁸ The ’725 Patent incorporates by reference U.S. Patent Application No. 09/608,237, which issued as the ’099 Patent.³⁹ The ’725 Patent also includes the same Figure 3 as the ’099 Patent and similarly discloses a network packet monitor that includes a parser 301, a flow-entry database 324, and an analyzer 303.⁴⁰

³⁴ ’099 Patent, 14:63–66.

³⁵ ’099 Patent, 14:66–15:1.

³⁶ ’099 Patent, 15:18–22.

³⁷ ’099 Patent, 15:39–41.

³⁸ ’725 Patent, 3:61–4:21.

³⁹ ’725 Patent, 2:21–30.

⁴⁰ ’725 Patent, 9:4–8, 10:58–65, Fig. 3.

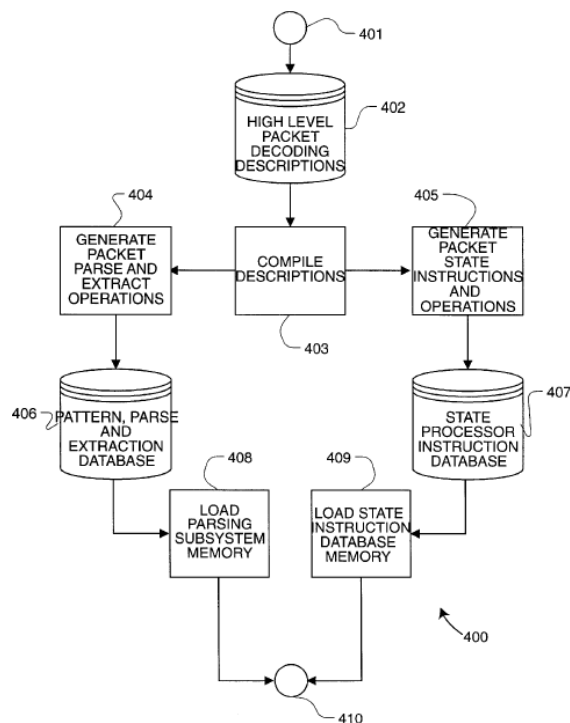
64. The '725 Patent states that the method includes performing various steps, such as receiving the packet and a set of protocol descriptions for protocols that may be used in the packet, and performing protocol-specific operations on the packet based on the protocol descriptions.⁴¹ The set of protocol descriptions may include child protocols, information related to the location of the child protocol, and any protocol-specific operations to be performed on the packet for the particular protocol at a particular layer level.⁴² The protocol-specific operations may also include state processing operations.⁴³

65. The '725 Patent's Figure 4 shows an exemplary flowchart illustrating the process of compiling the protocol descriptions into a data structure:

⁴¹ '725 Patent, Abstract, 4:63-5:12.

⁴² '725 Patent, 4:63-5:12, 41:59-65.

⁴³ '725 Patent, 21:45-22:4.



66. Much like the '099 Patent, the '725 Patent discloses that “[p]arser subsystem 301 examines the packets using pattern recognition process 304 that parses the packet and determines the protocol types and associated headers for each protocol layer that exists in the packet 302.”⁴⁴ Protocol description language (PDL) files 336:

[D]escribe[] both patterns and states of all protocols that ... occur at any layer, including how to interpret header information, how to determine from the packet header information the protocols at the next layer, and what information to extract for the purpose of identifying a flow, and ultimately, applications and services.⁴⁵

⁴⁴ '725 Patent, 9:17–20.

⁴⁵ '725 Patent, 9:29–35.

67. For each protocol, the respective PDL file provides the information needed by compiler/optimizer 310 to generate parsing/extraction database 308.⁴⁶ And the '725 Patent specifies that database 308 tells the parser how to extract packet information, including “one or more of what protocol-specific components of the packet to extract for the flow signature, how to use the components to build the flow signature, where in the packet to look for these components, where to look for any child protocols, and what child recognition patterns to look for.”⁴⁷

D. The '646 Patent Overview

68. The '646 Patent discloses a packet monitor and method of examining packets passing through a connection point on a computer network.⁴⁸ Like the '099 Patent, the '646 Patent discloses a network packet monitor that includes a parser, a flow-entry database, an analyzer with a lookup engine, a state processor, and a flow-insertion engine.⁴⁹ The '646 Patent also includes the same Figure 3 as the '099 Patent, with the same or similar descriptions.⁵⁰ In the '646 Patent, the packet

⁴⁶ '725 Patent, 41:57–59.

⁴⁷ '725 Patent, 41:59–65.

⁴⁸ '646 Patent, 1:42-3:14.

⁴⁹ '646 Patent, Abstract, 7:52-58, 9:45-52, Fig. 3.

⁵⁰ E.g., '646 Patent, Fig. 2, 8:5–9:28, 27:66–29:61 (Parser 301 “parses the packet and determines the protocol types and associated headers for each protocol layer that exists in the packet 302,” “extracts characteristic portions (signature information) from the packet 302,” and “build[s] a unique flow signature (also called a ‘key’) for this flow.”).

monitor further includes a packet acquisition device configured to receive packets passing through the connection point.⁵¹

69. Due to the high speed at which packets pass through the computer network, the '646 Patent discloses that it is advantageous to include a cache for the memory storing the flow-entry database.⁵²

One desirable property of such a cache system is a least recently used (LRU) replacement policy that replaces the LRU flow-entry when a cache replacement is needed.⁵³

Replacing least recently used flow-entries is preferred because it is likely that a packet following a recent packet will belong to the same flow.⁵⁴

70. After the monitor parses the packet, analyzer 303 determines whether the packet matches any of the previously-encountered flows by first looking in the cache of the flow-entry database 324.⁵⁵ Analyzer 303 processes the packet by, for example, determining whether the packet belongs to an existing flow or a new, previously unencountered flow.⁵⁶ If the latter, analyzer 303 performs state processing to determine whether the packet has been “fully characterized” and

⁵¹ '646 Patent, 4:67-5:8.

⁵² '646 Patent, 2:37-62.

⁵³ '646 Patent, 2:53-56.

⁵⁴ '646 Patent, 2:56-58.

⁵⁵ '646 Patent, 9:45-52.

⁵⁶ '646 Patent, 11:51-12:34, Figs. 8, 12.

whether the flow's classification can be "finalized."⁵⁷

E. The '751 Patent Overview

71. The '751 Patent discloses "[a] method of and monitor apparatus for analyzing a flow of packets passing through a connection point on a computer network."⁵⁸ Like the '099 and '646 Patents, the '751 Patent describes a network packet monitor that includes a flow-entry database and an analyzer.⁵⁹ And like the '646 Patent, the '751 Patent discloses a packet acquisition device configured to receive packets passing through the network's connection point.⁶⁰

72. The '751 Patent states that analyzer subsystem 303 processes each received packet to determine whether the packet matches any flow-entry in the flow-entry database 324, and, therefore, belongs to an existing flow.⁶¹ If so, the analyzer will perform state operations to update the flow-entry of the existing flow, including storing statistical measures kept in the flow-entry.⁶² If the packet does not match any flow-entry in the flow-entry database, the packet is part of a new flow and the analyzer will perform state operations for the initial state of the new flow and store

⁵⁷ '646 Patent, 9:45–12:34, 19:46–20:2, 30:13–36:28.

⁵⁸ '751 Patent, Abstract.

⁵⁹ '751 Patent.

⁶⁰ '751 Patent.

⁶¹ '751 Patent, 10:56-13:44.

⁶² '751 Patent, Abstract, 4:18-26, 18:1-8.

the new flow, including statistical measures, in the flow-entry database.⁶³

73. The '751 Patent discloses that the flow-entry database 324 stores flow entries using a plurality of protocols and at plurality of layer levels, including above the network layer.⁶⁴

F. The '789 Patent Overview

74. The '789 Patent discloses “[a] monitor for and a method of examining packets passing through a connection point on a computer network.”⁶⁵ On its cover, the '789 Patent identifies that it is a continuation of the application that issued as the '099 Patent.

75. Like the '099 and '646 Patents, the '789 Patent discloses a network packet monitor that includes a parser, a flow-entry database, an analyzer with a lookup engine, a state processor, and a flow-insertion engine.⁶⁶ And like the '646 and '751 Patents, the '789 Patent discloses a packet acquisition device configured to receive packets passing through the network's connection point.⁶⁷

76. The '789 Patent's Figure 11 shows an exemplary monitor configuration:

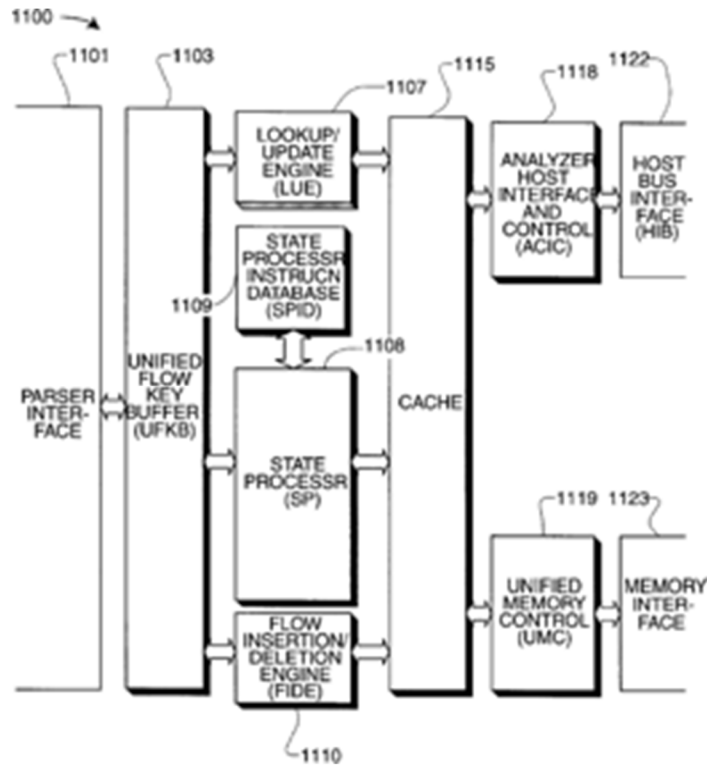
⁶³ '751 Patent, 13:11-27.

⁶⁴ '751 Patent, 11:16-36, 21:43-46.

⁶⁵ '789 Patent, Abstract.

⁶⁶ '789 Patent, Abstract, 11:66-12:5, 13:60-67, Fig. 3.

⁶⁷ '789 Patent, 9:11-20.



77. Like the '099, '646, '725, and '751 Patents, the '789 Patent discloses that the monitor receives a packet and determines whether the packet is part of an existing flow from a flow-entry database.⁶⁸ For this determination, the '789 Patent describes building a signature from portions of the packet for use to determine whether future packets are part of an existing flow.⁶⁹ If the packet is of a new flow, then the monitor performs states operations indicative of an initial state that in-

⁶⁸ '789 Patent, 13:5-54; 13:60-67.

⁶⁹ '789 Patent, 13:5-54; 13:60-67.

clude creating a signature and storing a new flow-entry in the flow-entry database.⁷⁰ If a packet is part of a known flow, then the monitor performs state operations to update the flow-entry in the flow-entry database.⁷¹

G. Prosecution History Overview

1. The '099 Patent's Prosecution History

78. The '099 Patent was filed on June 30, 2000, with 49 claims.⁷² On June 25, 2003, the Examiner allowed claims 1-10.⁷³ The Examiner noted that the prior art allegedly did not teach the claimed “state patterns/operations memory” or “state processor.”⁷⁴ The Examiner rejected claims 11-59 under 35 U.S.C §102(e) as being anticipated by U.S. Patent No. 6,483,804 to Muller.⁷⁵ In response to the rejection on July 8, 2003, the Applicants cancelled claims 11-59.⁷⁶ The '099 Patent then issued November 18, 2003.⁷⁷

2. The '725 Patent's Prosecution History

79. The '725 Patent was filed on June 30, 2000, with 18 claims.⁷⁸ On June 4,

⁷⁰ '789 Patent, 14:44-47, 15:21-23.

⁷¹ '789 Patent, 14:54-62.

⁷² '099 Prosecution History, 3 (Bib Data Sheet).

⁷³ '099 Prosecution History, 210-211 (06/25/2003 Office Action, cover).

⁷⁴ '099 Prosecution History, 212 (06/25/2003 Office Action, p.2).

⁷⁵ '099 Prosecution History, 213 (06/25/2003 Office Action, p.3).

⁷⁶ '099 Prosecution History, 584-590 (07/08/2003 Response to Office Action, pp.6-7).

⁷⁷ '099 Patent, cover.

⁷⁸ '725 Prosecution History, 1 (File Wrapper Cover).

2003, the Examiner rejected claims 1 and 16 under 35 U.S.C. §112 as indefinite and claims 1-3, 13-14, and 17-18 under 35 U.S.C. §102(b) as anticipated by U.S. Patent No. 5,860,585 to Bruell (“Bruell”).⁷⁹ The Examiner indicated that claims 4-11 and 15-16 contained allowable subject matter.⁸⁰

80. On June 13, 2003, the Applicants amended the claims based on the allowable subject matter indicated by the Examiner.⁸¹ The Applicants also argued that Bruell did not disclose the claimed invention.⁸² On June 27, 2003 the Applicants filed a supplemental response.⁸³ In these responses the Applicants argued in pertinent part that Bruell does not disclose that “the state of a flow” being “an indication of all previous events in the flow.”⁸⁴

81. On July 1, 2003, the Examiner allowed the amended claims without providing any reasons for allowance.⁸⁵

⁷⁹ '725 Prosecution History, 262-266 (06/04/2003 Office Action, cover).

⁸⁰ '725 Prosecution History, 267-268 (06/04/2003 Office Action, p.4).

⁸¹ '725 Prosecution History, 275-277 (06/13/2003 Response to Office Action, p.2).

⁸² '725 Prosecution History, 289-292 (06/13/2003 Response to Office Action, p.16).

⁸³ '725 Prosecution History, 293-298 (06/27/2003 Applicants' Supplemental Response, p.1).

⁸⁴ '725 Prosecution History, 290-291, (06/13/2003 Response to Office Action, p.17), 293-298 (06/27/2003 Applicants' Supplemental Response, p.1).

⁸⁵ '725 Prosecution History, 316-317 (10/27/2003 Response to Rule 312 Communication, cover).

3. The '646 Patent's Prosecution History

82. On June 30, 2000, the '646 Patent was filed with 20 claims.⁸⁶ On September 10, 2003, the Examiner rejected claims 7-11, 19, and 20 under 35 U.S.C. §102(e) as anticipated by U.S. Patent No. 4,458,310 to Chang (“Chang”) and claims 1 and 2 under 35 U.S.C. §102(e) as anticipated by U.S. Patent No. 5,917,821 to Gobuyan (“Gobuyan”). The Examiner also rejected claims 3-6 under 35 U.S.C. §103(a) as being obvious over Gobuyan in view of Chang and claims 12-18 as being obvious over Chang in view of U.S. Patent No. 6,003,123 to Carter.⁸⁷

83. On February 10, 2004, the Applicants amended claim 1 to include “a conversational flow being an exchange of one or more packets in any direction as a result of an activity corresponding to the flow” and “a state processor ... being to perform any state operations specified for the state of the flow”⁸⁸ The Applicants also added new claims 21-33.⁸⁹ The Applicants distinguished the claimed invention by arguing that Gobuyan did not disclose a “cache subsystem” or a “state processor.”⁹⁰ The Applicants further argued:

The present invention includes a process that recognizes *a conversational flow*. Gobuyan does not recognize a conversational flow, but instead looks

⁸⁶ '646 Prosecution History, 1 (File Wrapper Cover).

⁸⁷ '646 Prosecution History, 190-203 (09/10/2003 Office Action, cover).

⁸⁸ '646 Prosecution History, 342 (02/10/2004 Response to Office Action, p.2).

⁸⁹ '646 Prosecution History, 344-346 (02/10/2004 Response to Office Action, p.4).

⁹⁰ '646 Prosecution History, 348 (02/10/2004 Response to Office Action, cover).

up only each packet's destination address and source address. A conversational flow is not identified simply by the stations that are involved in a communication, but rather by the nature of the communication. E.g., the application program being invoked. Thus, even for the same two stations, the present invention identifies different conversational flows between two stations and maintains a different entry for each different conversational flow in a database.

It is important to be able to distinguish between packets that are exchanged between a source and a destination, and a *con[v]ersational flow* as used in the present invention. A conversational flow is the sequence of packets that are exchanged in any direction as a result of a particular[sic] activity-for instance, the running of an application on a server as requested by a client. Different conversational flows may occur between the same two addresses. Each of these would have a separate entry in the flow database. ...

Unlike Gobuyan, *the present invention is able to identify and classify conversational flows rather than only connection flows*. The reason for this is that some conversational flows involve more than one connection, and some even involve more than one exchange of packets between a client and server. Thus, there may be different *states* to a flow. This is particularly true when using client/server protocols such as RPC, DCOMP, and SAP, which enable a service to be set up or defined prior to any use of that service.⁹¹

⁹¹ Ex. 1020 ('646 Prosecution History, Feb. 10, 2004 Response to Office Action), 9 (bold and italics in original and underlining added). It is my understanding that the certified file history for the '646 Patent contains only the first page of the Feb. 10, 2004, Response. '646 Prosecution History, 341. The next page of the certified file

84. The Applicants went on to describe examples of conversational flows:

An example of such a case is the SAP (Service Advertising Protocol), a NetWare (Novell Systems, Provo, Utah) protocol used to identify the services and addresses of servers attached to a network. In the initial exchange, a client might send a SAP request to a server for print service. The server would then send a SAP reply that identifies a particular address-for example, SAP#5-as the print service on that server. Such responses might be used to update a table in a router, for instance, known as a Server Information Table. A client who has inadvertently seen this reply or who has access to the table (via the router that has the Service Information Table) would know that SAP#5 for this particular server is a print service. Therefore, in order to print data on the server, such a client would not need to make a request for a print service, but would simply send data to be printed specifying SAP#5. *Like the previous exchange, the transmission of data to be printed also involves an exchange between a client and a server, but requires a second connection and is therefore independent of the initial exchange. In order to eliminate the possibility of disjointed conversational exchanges, it is desirable for a network packet monitor to be able to “virtually concatenate” -that is, to link-the first exchange with the second. If the clients were the same, the two packet exchanges would then*

history includes the Feb. 20, 2004, Supplemental Response. This is clear from the fax mail markings on the top and bottom of each page as well as the signature. '646 Prosecution History, 342-347. Further, the Feb. 20, 2004 Supplemental Response states a “response to an office action was filed 10 Feb. 2004.” Ex. 1020 (Feb. 10, 2004 Response). Additionally, it is my understanding that this document was used in prior litigation, and that Patentee never disputed this document was the Feb. 10, 2004, Response.

*be correctly identified as being part of the same conversational flow.*⁹²

85. Regarding the claimed “existing flow” and “new flow,” Applicant specified these flows refer to “conversational flows”:

The analyzer subsystem . . . , for each packet, looks up a database of flow records for *previously encountered conversational flows* to determine whether a signature is from an existing flow. . . . [T]he analyzer further *identifies the state of the existing flow*, and performs *any state processing operations* specified for the state. In the case of a *newly encountered flow*, the analyzer includes a flow insertion and deletion engine for inserting new flows into the database of flows.⁹³

86. On April 14, 2004, the Examiner issued a Notice of Allowance that states “the closest prior art, Chang (U.S. Patent 4,458,310) discloses a cache memory subsystem that utilizes the use of flow entries, but fails to show that the ability to distinguish conversational data flow.”⁹⁴

4. The '751 Patent's Prosecution History

87. On June 30, 2000, the '751 Patent was filed with 21 claims.⁹⁵ On July 10,

⁹² Ex. 1020 ('646 Prosecution History, Feb. 10, 2004 Response to Office Action), 9 (throughout my declaration, all emphasis added unless otherwise noted).

⁹³ Ex. 1020 ('646 Prosecution History, Feb. 10, 2004 Response to Office Action), 8.

⁹⁴ '646 Prosecution History, 377 (04/14/2004 Notice of Allowance, p.4).

⁹⁵ '751 Prosecution History, 82-85 (06/30/2000 Patent Claims, p.78).

2003, the Examiner rejected claims 1-21 under 35 U.S.C. §102(e) as being anticipated by U.S. Patent No. 5,850,388 to Anderson (“Anderson”).⁹⁶

88. On November 3, 2003, the Applicants amended independent claims 1 and 17 to include, for example, “identifying the last encountered state of the flow” and “performing any state operations.”⁹⁷ The Applicants also asserted the purported invention was distinguishable from the teachings of Anderson.⁹⁸ For example, the Applicants asserted:

The present invention includes a process that recognizes *a conversational flow* and then generates statistics for the conversational flow. Anderson does not recognize a conversational flow, but instead compiles statistics for particular stations, and/or for particular network protocols used. A conversational flow is not identified simply by the stations that are involved in a communication, but rather by the nature of the communication, e.g., the application program being invoked.... It is important to be able to distinguish between the term “*connection flow*” commonly used to describe all the packets involved with a single connection, and a *con[v]ersational flow* as used in the present invention. A conversational flow is the sequence of packets that are exchanged in any direction as a result of an activity—for instance, the running of an application on a server as requested by a client. Unlike Anderson, *the present invention is able to identify and classify*

⁹⁶ ’751 Prosecution History, 172-179 (07/10/2003 Office Action, cover).

⁹⁷ ’751 Prosecution History, 221-235, (11/03/2003 Response to Office Action, cover).

⁹⁸ ’751 Prosecution History, 227-234 (11/03/2003 Response to Office Action, p.6).

conversational flows rather than only connection flows, including gathering statistics on the flows.⁹⁹

89. Regarding the claimed “existing flow,” Applicant specified these flows refer to “conversational flows”:

As an aspect of the present invention includes, for any packet ascertained to belong to an existing flow by looking up the database, identifying the state of the flow, and carrying out any state operations defined that that [sic] state; [prior art] Anderson has no concept of state of the flow, or even of a conversational flow, so that no such state operations are therefore carried out.¹⁰⁰

Anderson’s “previous session” is not a previously encountered conversational flow. Furthermore, Applicant’s lookup is to determine if a packet is part of an existing conversational flow.... Anderson’s “prior entries” are not the same as previously encountered conversational flows.¹⁰¹

90. On December 23, 2013, the Examiner found the Applicants’ arguments unpersuasive and again rejected claims 1-21 as being anticipated by Anderson.¹⁰² The Examiner also rejected claims 1-21 under 35 U.S.C. §102(e) as being anticipated by U.S. Patent No. 6,330,226 to Chapman and U.S. Patent No. 6,625,657 to

⁹⁹ ’751 Prosecution History, 229 (11/03/2003 Response to Office Action, p.8).

¹⁰⁰ ’751 Prosecution History, 230-31 (11/03/2003 Response to Office Action, p.9-10).

¹⁰¹ ’751 Prosecution History, 232 (11/03/2003 Response to Office Action, p.11).

¹⁰² ’751 Prosecution History, 243 (12/23/2003 Office Action, p.7).

Bullard.¹⁰³

91. On April 15, 2004, there was a telephone interview between the Examiner and the Applicants.¹⁰⁴ On April 19, 2004, the Applicants proposed amendments, which were discussed during the telephone interview.¹⁰⁵ Those amendments included, for example, adding the following limitations to claim 1 and similar limitations to claim 17:

“a conversational flow including an exchange of a sequence of one or more packets in any direction between two network entities as a result of a particular activity using a particular layered set of one or more network protocols, a conversational flow further having a set of one or more states, including an initial state” and

“wherein at least one step of the set consisting of of [sic] step (a) and step (b) includes identifying the protocol being used in the packet from a plurality of protocols at a plurality of protocol layer levels, such that the flow-entry database is to store flow entries for a plurality of conversational flows using a plurality of protocols, at a plurality of layer levels, including levels above the network layer.”¹⁰⁶

¹⁰³ '751 Prosecution History, 248, 253 (12/23/2003 Office Action, p.12, 17).

¹⁰⁴ '751 Prosecution History, 637 (04/15/2004 Interview Summary, cover).

¹⁰⁵ '751 Prosecution History, 639-648 (04/19/2004 Response to Final Office Action, cover).

¹⁰⁶ '751 Prosecution History, 642 (04/19/2004 Response to Final Office Action, p.3).

92. On June 4, 2004, the Examiner allowed the amended claims without providing any reasons for allowance.¹⁰⁷

5. The '789 Patent's Prosecution History

93. On October 14, 2003, the '789 Patent was filed with 59 claims.¹⁰⁸ That same day, Applicants filed a preliminary amendment cancelling claims 1-10.¹⁰⁹ On October 1, 2004, the Examiner rejected claims 11-59 under 35 U.S.C. §102(e) as being anticipated by U.S. Patent No. 6,483,804 to Muller ("Muller").¹¹⁰

94. On March 2, 2005, the Applicants submitted a declaration of inventor Russell Dietz and accompanying exhibits that purportedly shows that the Applicants had conceived of and reduced to practice independent claims 11, 29, and 54 prior to Muller.¹¹¹ The dates on the exhibits were redacted making it impossible to determine their exact date.¹¹² On May 3, 2005, the Examiner allowed the claims.¹¹³

H. Sandvine's IPR Petitions

95. I understand that Sandvine Corporation ("Sandvine") filed petitions requesting IPR of all of the Challenged Patents, as summarized below.

¹⁰⁷ '751 Prosecution History, 651-652 (10/14/2003 Notice of Allowability, cover).

¹⁰⁸ '789 Prosecution History, 2 (10/14/2003 Transmittal of New Application, cover).

¹⁰⁹ '789 Prosecution History, 104-112 (10/14/2003 Preliminary Amendment, p.1).

¹¹⁰ '789 Prosecution History, 174-178 (10/01/2004 Office Action, cover).

¹¹¹ '789 Prosecution History, 191-195 (03/02/2005 Declaration of Russell Dietz, cover).

¹¹² '789 Prosecution History, 195-437 (Mr. Dietz Declaration Exhibits, p.1).

¹¹³ '789 Prosecution History, 445-447 (05/03/2005 Notice of Allowability, cover).

IPR	Patent	Challenge Grounds
IPR2017-00450	'646	<ul style="list-style-type: none"> • Obviousness of claims 1-3, 7-9, 12, and 15-20 over U.S. Pat. No. 6,115,393 (“Engel”) and U.S. Pat. No. 5,530,834 (“Colloff”) • Obviousness of claims 13 and 14 over Engel, Colloff, and U.S. Pat. No. 6,182,146 (“Graham-Cumming”) • Obviousness of claims 10-11 over Engel, Colloff, and WO 97/23076 (“Baker”)
IPR2017-00451	'751	<ul style="list-style-type: none"> • Anticipation of claims 1-14, 17, and 19-21 by Engel • Obviousness of claims 15 and 16 over Engel and Graham-Cumming • Obviousness of claim 18 over Engel and Colloff
IPR2017-00629	'789	<ul style="list-style-type: none"> • Anticipation of claims 19-22, 25-26, 29-31, 36-40, and 42 by Engel • Obviousness of claims 23-24 over Engel and Baker • Obviousness of claims 27-28, 32, 41, and 43 over Engel and Graham-Cumming • Obviousness of claims 33-35 over Engel and Colloff
IPR2017-00630	'789	<ul style="list-style-type: none"> • Anticipation of claims 1-8, 11, 13, and 15 by Engel • Obviousness of claims 12, 44-45, and 47-48 over Engel and Baker • Obviousness of claims 14 and 16-18 over Engel and Graham-Cumming • Obviousness of claims 46 and 49 over Engel, Baker, and Graham-Gumming
IPR2017-00769	'099	<ul style="list-style-type: none"> • Obviousness of claims 1-5 and 9-10 over Engel, Baker, and Graham-Cumming • Obviousness of claims 6-8 over Engel, Baker, Graham-Cumming, and U.S. Pat. No. 4,532,606 (“Phelps”)
IPR2017-00862	'725	Anticipation of claims 10, 12-13, and 15-17 by Engel

IPR	Patent	Challenge Grounds
IPR2017-00863	'725	Anticipation of claims 1 and 2 by Baker

96. In IPR2017-00863, the Patent Trial and Appeal Board (the “Board”) instituted IPR for claims 1 and 2 of the ’725 Patent based on Baker, and Patentee thereafter abandoned those claims.¹¹⁴

I. Nokia’s IPR Petitions

97. I understand that Nokia Corporation and Nokia of America Corporation (collectively “Nokia”) filed petitions requesting IPR of all the Challenged Patents, as summarized below.

IPR	Patent	Challenge Grounds
IPR2019-01289	'751	<ul style="list-style-type: none"> • Anticipation or obviousness of claims 1, 2, 5, 10, and 14-15 by U.S. Pat. No. 6,412,000 (“Riddle”) or Riddle and U.S. Pat. No. 6,308,148 (“Bruins”) • Obviousness of claims 1, 2, 5, 10, and 14-15 over Riddle, Bruins, and RFC1945 - Hypertext Transfer Protocol -- HTTP/1.0 (“RFC1945”) • Obviousness of claims 1, 2, 5, 10, and 14-15 over Riddle, Bruins, RFC1889 - RTP: A Transport Protocol for Real-Time Applications (“RFC1889”), and RFC2326 - Real Time Streaming Protocol (RTSP) (“RFC2326”)
IPR2019-01290	'099	<ul style="list-style-type: none"> • Obviousness of claims 1, 2, 4, and 5 over Riddle, U.S. Pat. No. 6,091,725 (“Cheriton”), and Bruins • Obviousness of claims 1, 2, 4, and 5 over Riddle and RFC1945

¹¹⁴ Ex. 1062 (IPR2017-00863, Institution Decision); Ex. 1063 (Abandonment of Contest).

IPR	Patent	Challenge Grounds
		<ul style="list-style-type: none"> • Obviousness of claims 1, 2, 4, and 5 over Riddle, Cheriton, Bruins, RFC1889, and RFC2326
IPR2019-01291	'725	<ul style="list-style-type: none"> • Obviousness of claims 10, 12-13, and 16-17 over Riddle and Baker • Obviousness of claims 10, 12-13, and 16-17 over Riddle and RFC1945 • Obviousness of claims 10, 12-13, and 16-17 over Riddle, Baker, RFC1889, and RFC2326
IPR2019-01292	'646	<ul style="list-style-type: none"> • Obviousness of claims 1-3, 16, and 18 over Riddle, U.S. Pat. No. 5,740,175 (“Wakeman”), and Bruins • Obviousness of claim 7 over Riddle, Wakeman, Cheriton, and Bruins • Obviousness of claims 1-3, 16, and 18 over Riddle, Wakeman, Bruins, and RFC1945 • Obviousness of claim 7 over Riddle, Wakeman, Cheriton, Bruins, and RFC1945
IPR2019-01293	'789	<ul style="list-style-type: none"> • Anticipation or obviousness of claims 1-2 and 13-17 by Riddle or Riddle and Bruins • Anticipation or obviousness of claims 44, and 48-49 by Riddle or Riddle, Bruins, and U.S. Pat. No. 5,805,808 • Obviousness of claims 19-20, 31, and 42 over Riddle, Cheriton, and Bruins • Obviousness of claims 33-34 over Riddle, Cheriton, Bruins, and Wakeman

98. I understand that Patentee did not file responses to Nokia’s IPR petitions. Instead, Nokia and Patentee filed joint motions to terminate the IPR proceedings, which the Board granted on September 26, 2019.

J. German Nullity Proceeding

99. I understand that a European counterpart, EP 1196856 B1, to the Challenged Patents was subject to a Nullity Action in Germany, where the German Federal Patent Court found it “null and void.”¹¹⁵

100. The table below compares EP 1196856 claim 1 to ’646 Patent claim 16, and shows that EP 1196856 claim 1 includes limitations not found in ’646 Patent claim 16—i.e., the German court found patent claims narrower than the Challenged Claims invalid in view of the prior art.

EP 1 196 856 B1 Claim 1	The ’646 Patent Claim 16
A method of recognizing one or more conversational flows for packets passing through a connection point (121) on a computer network (102), each packet conforming to at least one protocol, wherein at least one said protocol defines one or more conversational flows that each includes a plurality of states of the flow including an initial state, and transitions from the initial state to at least one of the plurality of states of the flow, the method comprising:	[16. Pre] A method of examining packets passing through a connection point on a computer network, each packets [sic] conforming to one or more protocols, the method comprising:
receiving a packet (302) from a packet acquisition device;	[16.1] (a) receiving a packet from a packet acquisition device;

¹¹⁵ Ex. 1023 (Certified Translation of German Federal Patent Court, Case Nos. 2Ni 26/16 (EP) and 2Ni 46/16 (EP) (July 12, 2018)), 2.

EP 1 196 856 B1 Claim 1	The '646 Patent Claim 16
performing at least one parsing operation (304) and/or at least one extraction operation (306) on the packet to create a parser record comprising a function (312) of selected portions of the packet;	[16.2] (b) performing one or more parsing/extraction operations on the packet to create a parser record comprising a function of selected portions of the packet;
wherein at least one of the parsing and/or extraction operations depend on one or more of the protocols to which the packet conforms;	[16.6] wherein the parsing/extraction operations depend on one or more of the protocols to which the packet conforms. ¹¹⁶
looking up (314) in a flow-entry database (324) comprising flow entries for any previously encountered conversational flows, the look up using at least some of the selected packet portions and determining (316) if the packet is of an existing conversational flow;	[16.3] (c) looking up a flow-entry database comprising none or more flow-entries for previously encountered conversational flows, the looking up using at least some of the selected packet portions and determining if the packet is of an existing flow, the lookup being via a cache;
if the packet is of an existing conversational flow, classifying the packet as belonging to the found existing conversational flow and performing (328, 330) any state operation or operations specified in a database (326) for the state of the conversational flow; and	[16.4] (d) if the packet is of an existing flow, classifying the packet as belonging to the found existing flow; and

¹¹⁶ Note this limitation is presented out of order.

EP 1 196 856 B1 Claim 1	The '646 Patent Claim 16
<p>if the packet is of a new conversational flow, storing (322) a new flow-entry for the new conversational flow in the flow-entry database, including identifying information for future packets to be identified with the new flow-entry, determining the state of the flow (318) using the database (326) and performing (328, 330) any state operation or operations specified in the database (326) for the state of the flow;</p>	<p>[16.5] (e) if the packet is of a new flow, storing a new flow-entry for the new flow in the flow-entry database, including identifying information for future packets to be identified with the new flow-entry,</p>
<p>wherein each conversational flow that includes a plurality of states is recognized by transitioning through a</p>	
<p>plurality of states of the conversational flow, and at each state, carrying out one or more state operations specified in the database (326) for the state of the flow.”</p>	

101. While I understand that these findings do not bind the Board, I find the German court’s findings informative. First, the German court found that an FTP communication, which includes two TCP connections, taught a “conversational flow.”

As the German court wrote:

The person skilled in the art was familiar with the fact that an **FTP** communication is triggered via an initial TCP connection (“control channel”) and the associated user data transfer then takes place via a second TCP connection (“data channel”). That is, a normal FTP communication (OSI Layer 7) customarily comprises two TCP “connections” (OSI Layer 4) which are distinguished by the ports used (cf. above **I 7.1.2**). The **basic**

idea of the Patent in Suit consists in that the claimed method is supposed to produce an assignment of these two connections to one another; i.e., it recognizes two apparently different connections (of a lower layer, here TCP) as linked (to a single connection of a higher layer, here FTP).¹¹⁷

102. I agree with the German court that prior art teachings toward FTP communications meet the claimed “conversational flow,” at least under Patentee’s proposed construction of that term. As I detail below in Sections IV.A.4 and VII.A.2.e, such FTP communications include two or more TCP connections that result in the claimed “conversational flow.”

103. Second, the German court found invalid Patentee’s proposed claim amendments (termed “Alternative Applications”) that would have narrowed the alleged invention. In Alternative Applications 1 to 4, for example, Patentee sought to narrow the claims by “stating that such a conversational flow should comprise ‘more than one connection.’”¹¹⁸ And in Alternative Applications 5 to 9, Patentee sought to narrow the claims by stating that the conversational flow should comprise “more than one disjointed sub-flow.”¹¹⁹ Ultimately, the German court found Patentee’s proposed “Alternative Applications are also unsuccessful.”¹²⁰ I agree with the German court’s finding that the proposed amendments would not have successfully

¹¹⁷ Ex. 1023 (German Court Translation), 35-36 (emphasis in original).

¹¹⁸ Ex. 1023 (German Court Translation), 24, §7.1.

¹¹⁹ Ex. 1023 (German Court Translation), 24, §7.1.

¹²⁰ Ex. 1023 (German Court Translation), 34, §3.

differentiated the claims over the prior art.

IV. SUMMARY OF THE PRIOR ART

104. In Section III.A above, I described the state of the pertinent art in 1999. The following prior art references show that every element of the Challenged Claims was known in the art by June 30, 1999.

A. Riddle Overview

105. U.S. Patent No. 6,412,000 (“Riddle”) is titled “Method for Automatically Classifying Traffic in a Packet Communications Network.” Riddle was not considered by the USPTO during the original prosecution of the Challenged Patents.

106. I understand that a U.S. patent has an effective prior art date under pre-AIA 35 U.S.C. §102(e) based on the filing date of an earlier-filed patent application if the patent’s relevant subject matter is described in the earlier-filed application, and at least one of the patent’s claims is supported by the earlier-filed application’s written description in compliance with pre-AIA 35 U.S.C. §112, first paragraph.

107. The application that issued as Riddle was filed on November 23, 1998. Riddle claims priority to U.S. Provisional Patent Application No. 60/066,864 (“’864 Provisional”), which was filed on November 25, 1997.¹²¹ I understand that Riddle qualifies as prior art under at least 35 U.S.C. §102(e) based on its filing date, and under U.S.C. §102(e) based upon the filing date of the ’864 Provisional.

¹²¹ Riddle, cover.

108. I understand that Exhibit 1025 is a redlined comparison of Riddle's disclosures and the '864 Provisional's disclosures. This comparison shows that the two disclosures are substantially the same. The differences between the two specifications are minor, such as changed straight quotations to smart quotations. Aside from the minor differences shown in Exhibit 1025, the '864 Provisional contains every one of my declaration's citations to Riddle.

109. Further, Riddle's claims 1, 8, and 11, for example, are supported by the written description of the '864 Provisional in compliance with 35 U.S.C. §112, first paragraph. Exhibit 1026 is a table showing how the '864 Provisional's disclosures support Riddle's claims 1, 8, and 11. Further, many of Riddle's claims, such as claims 1 and 8, are similar to the '864 Provisional's claims, such as claims 1 and 7.

1. Overview of Riddle

110. Riddle explains that the physical components of communication networks, including computer clients, servers, and routers, were well-known in the prior art by 1997.¹²² Riddle states that, in the 1990's, there was a need to prioritize bandwidth.¹²³ According to Riddle, one solution to bandwidth issues was "by applying 'policies' to control traffic classified as to type of service required in order to more

¹²² Riddle, Figs. 1A-1C.

¹²³ Riddle, 3:40-42.

efficiently match resources with traffic.”¹²⁴ In order to apply such policies, Riddle teaches a technique to manage “network bandwidth based on information ascertainable from multiple layers of OSI network model.”¹²⁵

111. As explained in Section III.A.1, the Open Systems Interconnection (“OSI”) basic framework model is a standardized framework for network communications that was developed in the 1980s.¹²⁶ Riddle’s Figure 1D shows the OSI model:

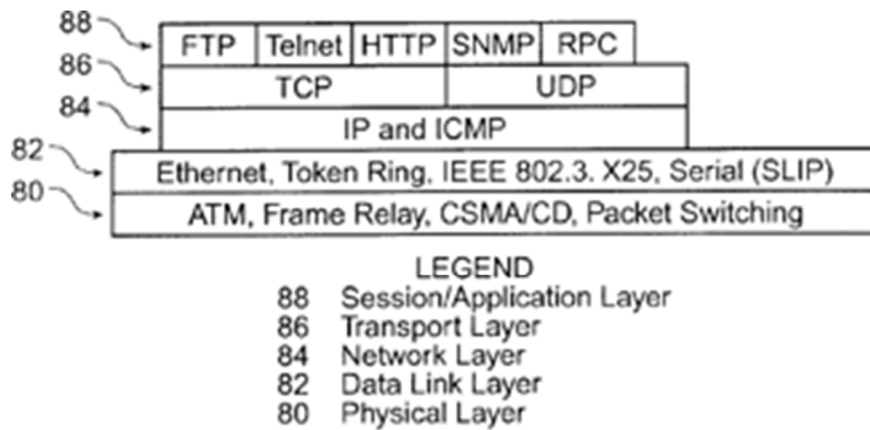


FIG. 1D
(PRIOR ART)

112. As shown in Riddle’s Figure 1D, examples of well-known protocols that are prior art to the Challenged Patents include FTP, HTTP, TCP, UDP, IP, and Ethernet. And as discussed above in Section IV.A.6, RTP and RTSP are other examples of well-known protocols that are prior art to the Challenged Patents.

¹²⁴ Riddle, 3:43-45.

¹²⁵ Riddle, 1:54-57.

¹²⁶ Ex. 1043 (ISO/IEC 7498 Part 4: Management Framework).

113. Riddle discloses that “there is no teaching in the prior art of methods for automatically classifying packet traffic based upon information gathered from a [sic] multiple layers in a multi-layer protocol network.”¹²⁷ In order to apply policies to control traffic, Riddle teaches that traffic can be classified a number of ways including by application or protocol. For example, Riddle states:

Traffic may be classified by type, e.g. E-mail, web surfing, file transfer, at various levels. For example, to classify by network paradigm, examining messages for an IEEE source/destination service access point (SAP) or a sub-layer access protocol (SNAP) yields a very broad indicator, i.e., SNA or IP. More specific types exist, such as whether an IP protocol field in an IP header indicates TCP or UDP. Well known connection ports provide indications at the application layer, i.e., SMTP or HTTP.¹²⁸

114. Riddle states that classifying traffic by application was known in the art.¹²⁹ Riddle teaches automatically classifying packet flows for use in allocating bandwidth resources by assigning service levels to packet flows.¹³⁰ To do so, Riddle discloses “applying individual instances of traffic classification paradigms to packet network flows based on selectable information obtained from a plurality of layers of a multi-layered communication protocol in order to define a characteristic

¹²⁷ Riddle, 3:36-39.

¹²⁸ Riddle, 3:46-54.

¹²⁹ Riddle, 3:55-4:2.

¹³⁰ Riddle, Abstract, 4:7-10.

class, then mapping the flow to the defined traffic class.”¹³¹ This “automatic classification is sufficiently robust to classify a complete enumeration of the possible traffic.”¹³²

115. Further, Riddle discloses “techniques to automatically classify a plurality of heterogeneous packets in a packet telecommunications system for management of network bandwidth in systems such as a private area network, a wide area network or an internetwork.”¹³³ As a result, “network managers need not know the technical aspects of each kind of traffic in order to configure traffic classes and service aggregates bundle traffic to provide a convenience to the user, by clarifying processing and enables the user to obtain group counts of all parts comprising a service.”¹³⁴

116. Riddle and the related ’864 Provisional incorporate-by-reference the following patent applications in their entirety:

- U.S. Patent Application No. 09/198,051 (Ex. 1028);
- U.S. Patent Application No. 08/762,828, issued as U.S. Patent No. 5,802,106 (Ex. 1029);
- U.S. Patent Application No. 08/977,642 (Ex. 1027), having attorney docket number 17814-5.10, and issued as U.S. Patent No. 6,046,980 (Ex.

¹³¹ Riddle, 4:10-15.

¹³² Riddle, 4:15-17.

¹³³ Riddle, 4:55-60.

¹³⁴ Riddle, 4:18-23.

1031); and

- U.S. Patent Application No. 08/742,994, issued as U.S. Patent No. 6,038,216 (Ex. 1030).¹³⁵

It is my understanding that, because of this incorporation by reference, the disclosures of each of the above patent applications are a part of and fully included in the disclosures of Riddle and the '864 Provisional.

2. Riddle's Hardware Components

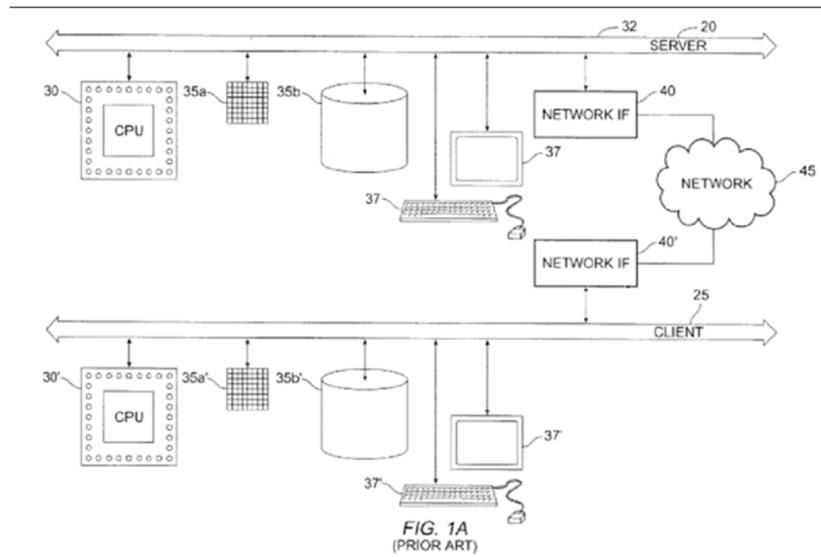
117. Riddle teaches that its monitoring method is “implemented in the C programming language and is operational on a computer system such as shown in FIG. 1A.”¹³⁶ And Riddle states that its “invention may be implemented in a client-server environment, but a client-server environment is not essential.”¹³⁷ As shown below in Figure 1A, Riddle shows that network systems include network interfaces 40, storage subsystems 35, and network connection 45.¹³⁸

¹³⁵ Riddle, 1:13-18, 1:29-51; Ex. 1024 (U.S. Provisional Patent Application No. 60/066,864 (“’864 Provisional”)), 3-4.

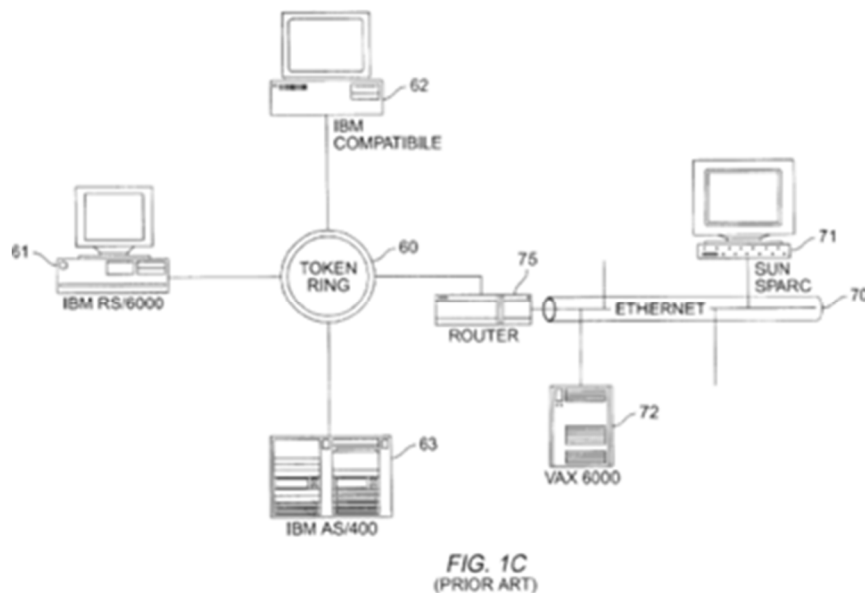
¹³⁶ Riddle, 5:55-57.

¹³⁷ Riddle, 5:57-59.

¹³⁸ Riddle, 6:1-7:34, Fig. 1B.



118. Riddle’s monitoring system can include router 75, “a network access point” with token ring and ethernet adapters, as shown below in Figure 1C.¹³⁹ Riddle specifies that such routers are aware of inter-network protocols (e.g., ICMP, RIP).



¹³⁹ Riddle, 7:29-34, Fig. 1C.

3. Riddle's Parsing of Packets

119. Riddle discloses that traffic classifier 304 detects the protocols and services in each packet.¹⁴⁰ As shown below in Figure 3, Riddle's traffic classifier 304 parses packets to identify the packet's flow specification.¹⁴¹ Traffic classifier 304 stores those packets' flow specifications, including traffic-type, in list 308 for classification.¹⁴²

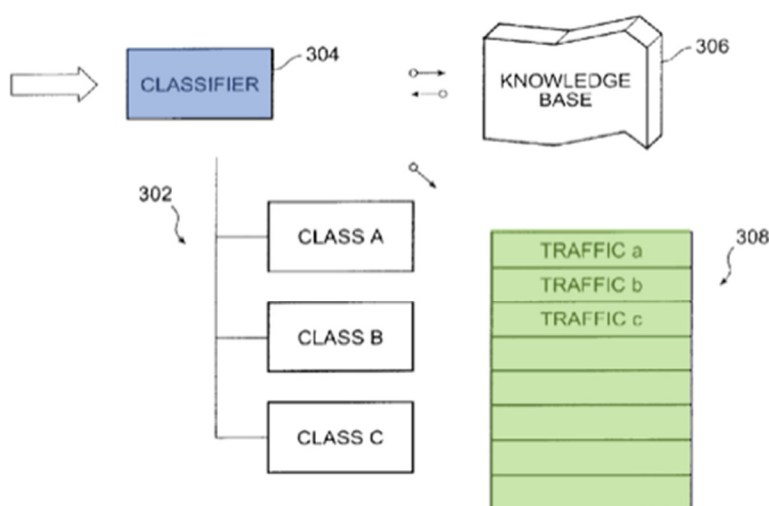


FIG. 3

120. Riddle illustrates the parsing and classifying of packet traffic in flow chart 401, as shown below in Figure 4A.¹⁴³ At step 402, Riddle discloses parsing the

¹⁴⁰ Riddle, 11:47-67, 12:27-32, 14:28-41, Fig. 3.

¹⁴¹ Riddle, 11:47-67, 12:37-41.

¹⁴² Riddle, 12:37-53.

¹⁴³ Riddle, 4:48-50, 12:42-63, Fig. 4A.

flow specification from the packet flow being classified.¹⁴⁴ Examples of information extracted from parsed packet flows include:

- Protocol family;
- Direction of packet flow;
- Protocol type;
- Pair of hosts (*i.e.*, source and destination network-layer addresses);
- Pair of ports (*i.e.*, source and destination transport-layer port numbers); and
- HTTP protocol packets MIME type.¹⁴⁵

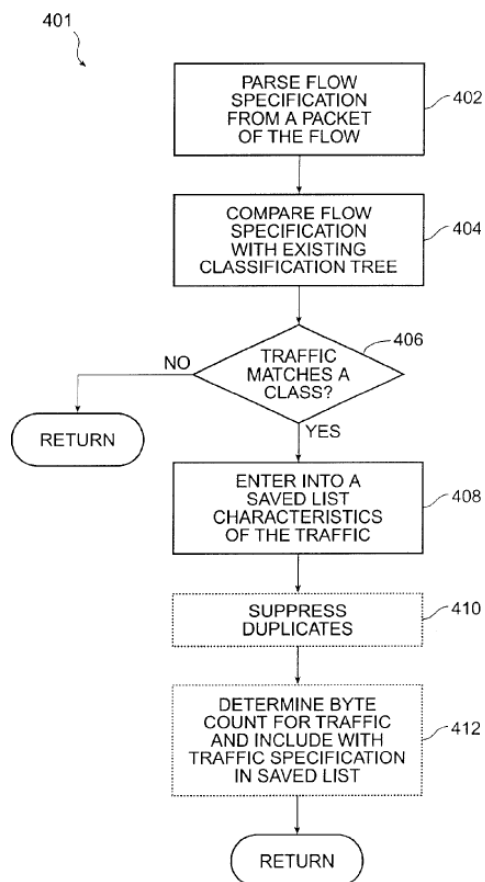


FIG. 4A

¹⁴⁴ Riddle, 12:43-44, Fig. 4A.

¹⁴⁵ Riddle, 12:50-53, claims 1, 8, 11.

121. As shown above, at step 406, Riddle's classifier analyzes whether the packet's flow specification matches a traffic class.¹⁴⁶ If the flow specification matches, at step 408, Riddle's classifier enters the identifying characteristics of the traffic into the corresponding flow-entry in the saved list.¹⁴⁷ And at steps 410 and 412, a POSITA would have understood that Riddle's classifier checks if the flow is a new flow or an existing flow (e.g., suppressing duplicates for existing flows).¹⁴⁸

4. Riddle's Classifying Flows Based on Conversations

122. Riddle teaches using a classification tree to organize the relationships between packet traffic passing through the monitor.¹⁴⁹ Riddle details that "[e]ach node of the classification tree represents a class, and has a traffic specification, i.e., a set of attributes or characteristics describing the traffic associated with it."¹⁵⁰ An exemplary classification tree 302 is shown below in Riddle's Figure 3. Riddle's classifier 304 classifies pending traffic flows (e.g., a, b, and c) under particular member class nodes (e.g., classes A, B, and C) within tree 302.¹⁵¹

¹⁴⁶ Riddle, 12:48-53, Fig. 4A.

¹⁴⁷ Riddle, 12:48-53, Fig. 4A.

¹⁴⁸ Riddle, 12:53-60, Fig. 4A.

¹⁴⁹ Riddle, 9:29-32, 12:37-41, Fig. 3.

¹⁵⁰ Riddle, 9:29-32.

¹⁵¹ Riddle, 12:27-30.

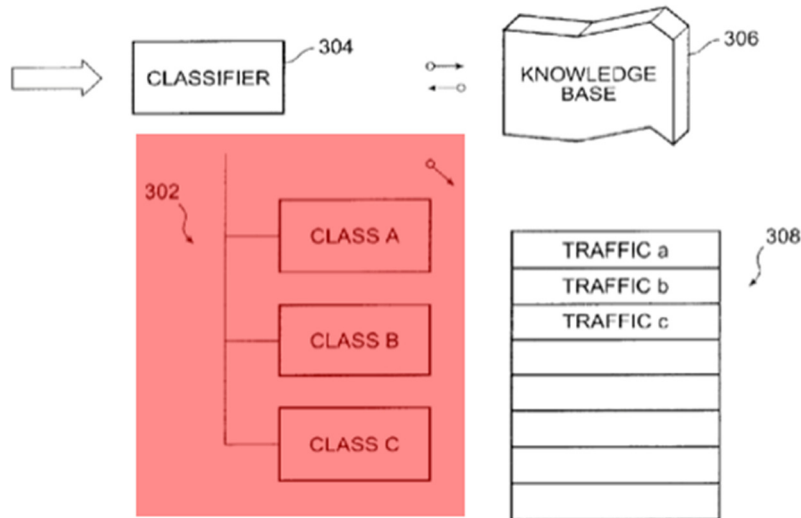


FIG. 3

123. Riddle illustrates the classification process at Figure 4’s steps 404 to 412. At steps 404 and 406, Riddle teaches comparing a given traffic’s parsed specification with the classification tree to determine if the traffic matches a class.¹⁵² Riddle specifies that a traffic class is “[a]ll traffic between a client and a server endpoints. A single instance of a traffic class is called a flow. Traffic classes have properties or class attributes such as, directionality, which is the property of traffic to be flowing inbound or outbound.”¹⁵³ Riddle further explains that “[a] flow is a single instance of a traffic class. For example, all packets in a TCP connection belong to the same flow. As do all packets in a UDP session.”¹⁵⁴

¹⁵² Riddle, 12:44-50.

¹⁵³ Riddle, 5:42-45.

¹⁵⁴ Riddle, 5:17-20.

124. Using traffic classes and classification trees, Riddle teaches a highly customizable way to detect and classify traffic: “The present invention provides a method for classifying traffic according to a definable set of classification attributes selectable by the manager, including selecting a subset of traffic of interest to be classified. The invention provides the ability to classify and search traffic based upon multiple orthogonal classification attributes.”¹⁵⁵

125. Riddle discloses defining traffic classes by application-level attributes:

Traffic classes may be defined at any level of the IP protocol as well as for other non-IP protocols. For example, at the IP level, traffic may be defined as only those flows between a specified [sic] set of inside and outside IP addresses or domain names. An example of such a low level traffic class definition would be all traffic between my network and other corporate offices throughout the Internet. *At the application level, traffic classes may be defined for specific URIs within a web server. Traffic classes may be defined having “Web aware” class attributes.* For example, a traffic class could be created such as all URIs matching “*.html” for all servers, or all URI patterns matching “*.gif” for server X, or for access to server Y with URI pattern “/sales/*” from client Z, wherein “*” is a wildcard character, i.e., a character which matches all other character combinations. Traffic class attributes left unspecified will simply match any value for that attribute. For example, a traffic class that accesses data objects within a certain directory path of a web server is specified by a URI

¹⁵⁵ Riddle, 9:14-19.

pattern of the directory path to be managed, e.g. “/sales/*”.¹⁵⁶

126. Riddle discloses Table 2, which includes examples of information for building traffic classes.¹⁵⁷ Table 2 specifies exemplary client-side, server-side, and global components for a traffic class. As an example, one traffic class is a global FTP application (a client-server software program for transferring files using the FTP protocol) using a specific client-side IP address and a specific server-side IP address, as shown below in Table 2.

Components of a Traffic Class Specifier

Inside (Client or Server)	Global	Outside (Server or Client)
IP Address/Domain Name	TCP or UDP Service	IP Address/Domain Name
Port Number	e.g., WWW, FTP, RealAudio, etc.	Port Number
MAC Address	URI pattern for Web Service, MIME type for Web Service IPX Service SNA Service LAT Service IP precedence	MAC Address

127. As discussed above in Section IV.A, minor differences exist between Riddle’s Table 2 and the related ’864 Provisional’s Table 2.¹⁵⁸ For purposes of my analysis, I rely on Table 2 materials as they appear in the provisional application.

¹⁵⁶ Riddle, 8:58-9:11.

¹⁵⁷ Riddle, 9:64-65.

¹⁵⁸ Ex. 1024 (’864 Provisional), 19; Ex. 1025 (Redline comparing Riddle to ’864 Provisional), 16.

Moreover, my understanding is the '864 Provisional provides support for all of the changes appearing in Riddle's Table 2. For example, Riddle's Table 2 includes classifying based upon port numbers, which is shown throughout the provisional application at section 3.1.6 "Dynamic Ports" and other areas.¹⁵⁹

128. Riddle also describes a traffic class being based on a "service aggregate" for applications with multiple connections in a particular conversation flow between computers:

A service aggregate is provided *for certain applications that use more than one connection in a particular conversation between a client and a server*. For example, an FTP client in *conversation* with an FTP server employs a command channel and a transfer channel, which are distinct TCP sessions on two different ports. In cases where two or three TCP or UDP sessions exist *for each conversation* between one client and one server, it is useful to provide a common traffic class i.e., the service aggregate, containing *the separate conversations*. In practice, these types of *conversations* are between the same two hosts, but use different ports. According to the invention, a class is created with a plurality of traffic specifications, each matching various component *conversations*.¹⁶⁰

As described in greater detail below, a "service aggregate" satisfies the construction of "conversational flow" under either party's proposed construction because it

¹⁵⁹ Ex. 1024 ('864 Provisional), 22.

¹⁶⁰ Riddle, 11:10-23.

links multiple disjointed flows for a common application. Further, Riddle teaches its classifier defines a service aggregate class as part of the initial classification tree or while analyzing the traffic.¹⁶¹

129. Riddle classifies flows by “a series of steps through a traffic class tree, with the last step (i.e., at the leaves on the classification tree) mapping the flow to a policy.”¹⁶² These classification steps include:

[T]he classification process checks at each level if the flow being classified matches the attributes of a given traffic class. If it does, processing continues down to the links associated with that node in the tree. If it does not, the class at the level that matches determines the policy for the flow being classified. If no policy specific match is found, the flow is assigned the default policy.

In a preferred embodiment, *the classification tree is an N-ary tree with its nodes ordered by specificity*. For example, in classifying a particular flow in a classification tree ordered first by organizational departments, the attributes of the flow are compared with the traffic specification in each successive department node and if no match is found, then processing proceeds to the next subsequent department node. If no match is found, then the final compare is a default “match all” category. If, however, a match is found, then classification moves to the children of this department node. The child nodes may be ordered by an orthogonal paradigm such as, for example, “service type.” Matching proceeds according to the order of

¹⁶¹ Riddle, 10:31-33, 13:54-59.

¹⁶² Riddle, 9:20-25.

specificity in the child nodes. Processing proceeds in this manner, traversing downward and from left to right in FIGS. 2A and 2B, which describe a classification tree, searching the plurality of orthogonal paradigms. ***Key to implementing this a hierarchy is that the nodes are arranged in decreasing order of specificity. This permits search to find the most specific class for the traffic before more general.***¹⁶³

130. Riddle illustrates an exemplary classification tree in Figure 2A, below, where Riddle's packet classifier tests whether the parsed flow is for Department A resources (202) by comparing the packet's source (client) IP to the range of IP addresses defined for subnet A.¹⁶⁴ If it is, then the packet classifier updates the state and tests whether the flow is for an FTP outside port 2.0 (206) or for a worldwide web server (208).¹⁶⁵ If the flow was not for Department A resources, then Riddle's packet classifier tests whether the flow is for Department B resources (204) by comparing the packet's source (client) IP to the range of IP addresses defined for subnet B.¹⁶⁶ If it is, then the packet classifier updates the state and tests whether the flow is for an FTP server (210) or another worldwide web server (212).¹⁶⁷ And if the flow was not for Department A or B resources, then the flow falls into a default

¹⁶³ Riddle, 9:28-63.

¹⁶⁴ Riddle, 10:19-39.

¹⁶⁵ Riddle, 10:21-33, Fig. 2.

¹⁶⁶ Riddle, 10:19-39.

¹⁶⁷ Riddle, 10:21-26, 10:36-39, Fig. 2A.

classification (205).¹⁶⁸

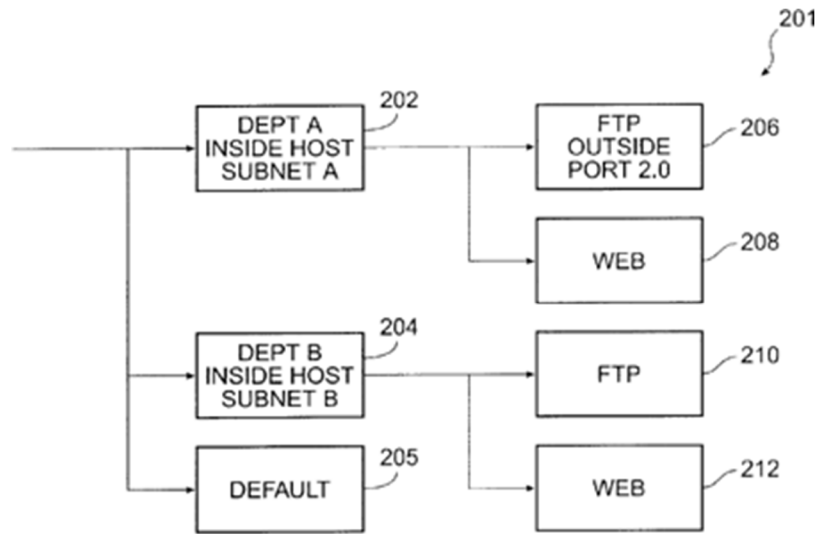


FIG. 2A

131. Riddle illustrates another exemplary classification tree in Figure 2B, as provided below. In Figure 2B, Riddle's packet classifier tests whether the parsed flow is web traffic (220). If it is, then the packet classifier updates the state and tests whether the flow is for Department A (226) or Department B (228).¹⁶⁹ If the flow is not web traffic (220), then Riddle's packet classifier updates the state and tests whether the flow is for TCP traffic (224). If it is, then the packet classifier updates that state and tests whether the flow is for Department A (230) or Department B (232).¹⁷⁰ And if the flow is not web or TCP traffic, then the flow falls into a default classification (225).¹⁷¹

¹⁶⁸ Riddle, 10:52-56, Fig. 2A.

¹⁶⁹ Riddle, 10:40-47, Fig. 2B.

¹⁷⁰ Riddle, 10:40-51, Fig. 2B.

¹⁷¹ Riddle, 10:52-56, Fig. 2B.

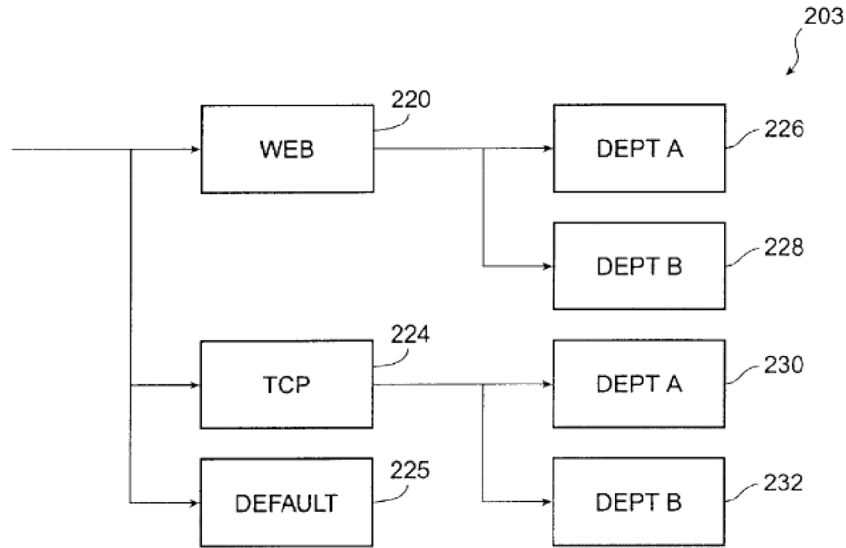


FIG. 2B

5. Riddle's Conversational Flow Analyzer

132. As shown in Figure 4A's flowchart, Riddle's step 406 analyzes whether the packet's flow specification matches a traffic class.¹⁷² If it does, then Riddle adds an entry to the traffic class's list of identifying characteristics (step 408).¹⁷³ And at steps 410 and 412, Riddle checks if the flow is a new flow or an existing flow. For example, Riddle may suppress duplicate flows and just update the traffic class's list for the number of duplicates, byte count, and the time at which the most recent flow was encountered.¹⁷⁴ This operation can be explained using an instance of an FTP application pre-defined as a traffic class. When Riddle's monitor receives an initial FTP-command packet for a new FTP transfer, the monitor extracts the

¹⁷² Riddle, 12:48-53, Fig. 4A.

¹⁷³ Riddle, 12:50-53, Fig. 4A.

¹⁷⁴ Riddle, 12:53-60, Fig. 4A.

packet's flow specification and enters it into saved list 308. If that flow specification matches the FTP application traffic classification, the monitor classifies the traffic type as an instance of the FTP application. Riddle's monitor checks subsequently received FTP-command and FTP-data packets to see if they belong to the same FTP application classification, based on their respectively extracted flow specifications. If those flow specifications relate to the same instance of the FTP application, the monitor updates the identifying characteristics and statistics in the original entry in saved list 308 to reflect receipt of the new packets and suppresses duplicate traffic flow entries. Suppressing duplicates in this manner allows the user to obtain a group count of all parts of the transfer, and to track the bandwidth consumed by the transfer.¹⁷⁵

¹⁷⁵ Riddle, 4:6-10, 4:18-23.

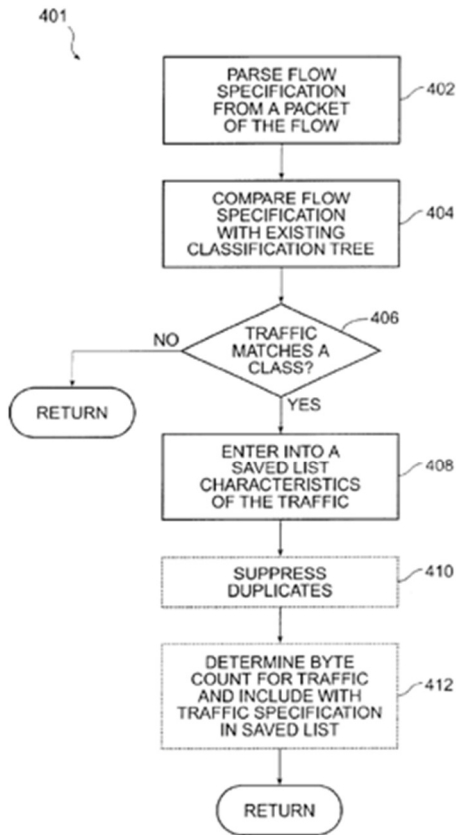


FIG. 4A

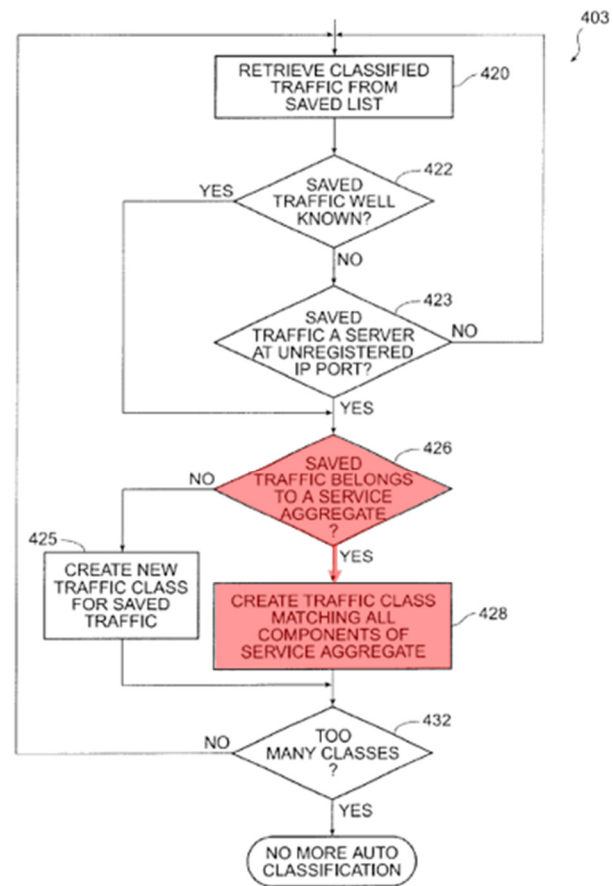


FIG. 4B

133. And as shown above in Figure 4B's flowchart, Riddle teaches analyzing whether the packet's flow belongs to a service aggregate (step 426).¹⁷⁶ Riddle describes analyzing whether the flow belongs to a conversation composed of multiple connection flows, and if so, creating a traffic class to match all connection flows of the conversation (step 428):

In decisional step 426, the instance of saved traffic is examined to determine whether it belongs to a service aggregate. For example, an FTP session has one flow that is used to exchange commands and responses and a

¹⁷⁶ Riddle, 13:43-62, Fig. 4B.

second flow that is used to transport data files. If the traffic does belong to a service aggregate, then in a step 428, a traffic class is created which will match all components of the service aggregate.¹⁷⁷

The '864 Provisional further illustrates how a POSITA would have understood Riddle's "service aggregate" relates multiple connection flows based on an FTP application's specific software program activity:

[T]he concept of "service aggregates" (service groups) [is] different traffic types that are associated together (ex. FTP has one stream that it uses to exchange commands and responses, and a second that the data files are actually sent over). Whenever we recognize the signature of one of these types of traffic, we create a traffic class (or class hierarchy) that can match all the components of the aggregate. This bundling is mainly a convenience to the user, makes it clearer what's going on, but also permits you to get group counts of all the parts that make up what the user thinks as the service.¹⁷⁸

6. Riddle's Traffic Identification Based on RTP and RTSP

134. Riddle describes identifying a traffic class based on the type of resource used by the traffic.¹⁷⁹ Riddle provides examples of resources creating such connections, such as detecting Real Time Protocol and/or Real Time Steaming Protocol flows.¹⁸⁰

¹⁷⁷ Riddle, 13:54-61, 11:10-23, Fig. 4B.

¹⁷⁸ '864 Provisional, 69.

¹⁷⁹ Riddle, 12:1-12.

¹⁸⁰ Riddle, 7-12.

135. As I detail below regarding two pertinent RFC publications, RTP and RTSP are protocols that support functions suitable for applications transmitting real-time data, such as audio or video data. As with other protocols, such as FTP, both RTP and RTSP use a separate control flow with one or more linked dataflows.

a. *RFC 1889 - RTP: A Transport Protocol for Real-Time Applications*

136. RFC 1889 - RTP: A Transport Protocol for Real-Time Applications (“RFC1889”) is a printed publication available to the public on or around January 1996. It is my understanding that RFC1889 is prior art to each of the Challenged Patents.

137. H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson authored RFC1889. Like other RFCs, RFC1889 was the product of an IETF working group. I obtained a copy of RFC1889, and made use of it, on or around 1996.¹⁸¹

138. RFC1889 describes RTP, the real-time transport protocol, which provides:

[E]nd-to-end network transport functions suitable for applications transmitting real-time data, such as audio, video or simulation data, over multicast or unicast network services. RTP does not address resource reservation and does not guarantee quality-of-service for real-time services. The data transport is augmented by a control protocol (RTCP) to allow monitoring of the data delivery in a manner scalable to large multicast networks,

¹⁸¹ Ex. 1045 (RFC1889 Internet-Archive Affidavit) (illustrating RFC1889 was publicly accessible and available for download by May 30, 1998).

and to provide minimal control and identification functionality. RTP and RTCP are designed to be independent of the underlying transport and network layers. The protocol supports the use of RTP-level translators and mixers.¹⁸²

139. RFC1889 explains that RTP has two “closely-linked parts.”¹⁸³ Part one is “the real-time transport protocol (RTP), to carry data that has real-time properties.”¹⁸⁴ And part two is “the RTP control protocol (RTCP), to monitor the quality of service and to convey information about the participants in an on-going session.”¹⁸⁵

140. RFC1889 also discloses that audio and video are transmitted separately as “sessions” and that the protocol provides a means by which related sessions can be associated. For example, RFC1889 states:

If both audio and video media are used in a conference, they are transmitted as separate RTP sessions RTCP packets are transmitted for each medium using two different UDP port pairs and/or multicast addresses. There is no direct coupling at the RTP level between the audio and video sessions, except that a user participating in both sessions should use the same distinguished (canonical) name in the RTCP packets for both so that the sessions can be associated.¹⁸⁶

¹⁸² Ex. 1045 (RFC1889 Internet-Archive Affidavit), 4.

¹⁸³ Ex. 1045 (RFC1889 Internet-Archive Affidavit), 5.

¹⁸⁴ Ex. 1045 (RFC1889 Internet-Archive Affidavit), 5.

¹⁸⁵ Ex. 1045 (RFC1889 Internet-Archive Affidavit), 5.

¹⁸⁶ Ex. 1045 (RFC1889 Internet-Archive Affidavit), 6.

141. RFC1889 defines an RTP session as:

The association among a set of participants communicating with RTP. For each participant, the session is defined by a particular pair of destination transport addresses (one network address plus a port pair for RTP and RTCP). The destination transport address pair may be common for all participants, as in the case of IP multicast, or may be different for each, as in the case of individual unicast network plus a common port pair. In a multimedia session, each medium is carried in a separate RTP session with its own RTCP packets. The multiple RTP sessions are distinguished by different port number pairs and/or different multicast addresses.¹⁸⁷

142. Moreover, RFC1889 details that RTP and RTCP are separate flows:

RTP relies on the underlying protocol(s) to provide demultiplexing of RTP data and RTCP control streams. *For UDP and similar protocols, RTP uses an even port number and the corresponding RTCP stream uses the next higher (odd) port number.* If an application is supplied with an odd number for use as the RTP port, it should replace this number with the next lower (even) number.¹⁸⁸

b. *RFC 2326 – Real Time Streaming Protocol (RTSP)*

143. RFC 2326 - Real Time Streaming Protocol (RTSP) (“RFC2326”) is a printed publication available to the public on or around April 1998. It is my understanding that RFC2326 is prior art to each of the Challenged Patents.

¹⁸⁷ Ex. 1045 (RFC1889 Internet-Archive Affidavit), 7.

¹⁸⁸ Ex. 1045 (RFC1889 Internet-Archive Affidavit), 25.

144. H. Schulzrinne, A. Rao, and R. Lanphier authored RFC2326. Like other RFCs, RFC2326 was the product of an IETF working group. I obtained a copy of RFC2326, and made use of it, on or around 1998.¹⁸⁹

145. RFC2326 describes RTSP, the real time streaming protocol, as:

[A]n application-level protocol for control over the delivery of data with real-time properties. RTSP provides an extensible framework to enable controlled, on-demand delivery of real-time data, such as audio and video. Sources of data can include both live data feeds and stored clips. This protocol is intended to control multiple data delivery sessions, provide a means for choosing delivery channels such as UDP, multicast UDP and TCP, and provide a means for choosing delivery mechanisms based upon RTP (RFC 1889).¹⁹⁰

146. RFC2326 details that RTSP “establishes and controls either a single or several time-synchronized streams of continuous media such as audio and video.”¹⁹¹

RTSP “does not typically deliver the continuous streams itself, although interleaving of the continuous media stream with the control stream is possible In other words, RTSP acts as a ‘network remote control’ for multimedia servers.”¹⁹²

147. Regarding the interplay between RTP and RTSP, RFC2326 specifies that

¹⁸⁹ Ex. 1046 (RFC2326 Internet-Archive Affidavit) (illustrating RFC2326 was publicly accessible and available for download by May 30, 1998).

¹⁹⁰ Ex. 1046 (RFC2326 Internet-Archive Affidavit), 4.

¹⁹¹ Ex. 1046 (RFC2326 Internet-Archive Affidavit), 5-6.

¹⁹² Ex. 1046 (RFC2326 Internet-Archive Affidavit), 5-6.

“streams controlled by RTSP may use RTP ..., but the operation of RTSP does not depend on the transport mechanism used to carry continuous media.”¹⁹³ The RTSP protocol “is intentionally similar in syntax and operation to HTTP/1.1 ... so that extension mechanisms to HTTP can in most cases also be added to RTSP.”¹⁹⁴

148. RFC2326 explains that RTSP has separate control and data channels and requires tracking of session states:

RTSP controls a stream which may be sent via a separate protocol, independent of the control channel. For example, RTSP control may occur on a TCP connection while the data flows via UDP. Thus, data delivery continues even if no RTSP requests are received by the media server. Also, during its lifetime, a single media stream may be controlled by RTSP requests issued sequentially on different TCP connections. *Therefore, the server needs to maintain “session state” to be able to correlate RTSP requests with a stream. The state transitions are described in Section A.*¹⁹⁵

149. Further, RFC2326 teaches that RTSP requires state monitoring: “An RTSP server needs to maintain state by default in almost all cases....”¹⁹⁶ Further, “RTSP requests are also not stateless; they may set parameters and continue to control a

¹⁹³ Ex. 1046 (RFC2326 Internet-Archive Affidavit), 5-6.

¹⁹⁴ Ex. 1046 (RFC2326 Internet-Archive Affidavit), 5-6.

¹⁹⁵ Ex. 1046 (RFC2326 Internet-Archive Affidavit), 9.

¹⁹⁶ Ex. 1046 (RFC2326 Internet-Archive Affidavit), 5-6.

media stream long after the request has been acknowledged.”¹⁹⁷ Thus, both the RTSP client and server require a state machine.¹⁹⁸

B. Ferdinand Overview

150. WO 92/19054 (“Ferdinand”) is titled “Network Monitoring,” and published on October 29, 1992.¹⁹⁹ Ferdinand was not considered by the USPTO during the original prosecution of the Challenged Patents.

151. As I discuss above, the German Federal Patent Court found a counterpart to the Challenged Patents “null and void.”²⁰⁰ I understand that, in so doing, the German court found Ferdinand to be similar prior art.²⁰¹

152. Ferdinand relates to “monitoring and managing communication networks for computers.” For example, Ferdinand describes network monitor 10 analyzing and extracting information about packets passing through the network in real-time:

Network Monitor 10 ... is the data collection module which is attached to the LAN. It is a high performance real time front end processor which collects packets on the network and performs some degree of analysis to search for actual or potential problems and to maintain statistical infor-

¹⁹⁷ Ex. 1046 (RFC2326 Internet-Archive Affidavit), 9.

¹⁹⁸ Ex. 1046 (RFC2326 Internet-Archive Affidavit), 36-37.

¹⁹⁹ Ferdinand, cover.

²⁰⁰ Ex. 1023 (German Court Translation), 2.

²⁰¹ Ex. 1023 (German Court Translation), 8, 30-32 (referring to Ferdinand as reference E2).

mation for use in later analysis. In general, it performs the following functions. It operates in a promiscuous mode to capture and analyze all packets on the segment and it extracts all items of interest from the frames. It generates alarms to notify the Management Workstation of the occurrence of significant events. It receives commands from the Management Workstation, processes them appropriately and returns responses.²⁰²

153. As part of network monitor 10, Ferdinand discloses separate memories devoted to separate functionalities, such as a packet-buffer memory and system control blocks.²⁰³ And as shown below in Figures 5, Ferdinand describes network monitor 10 may include separate hardware components, including real time parser 32, boot loader 22, and database 36 for storing flow-entry information of the examined packets.²⁰⁴

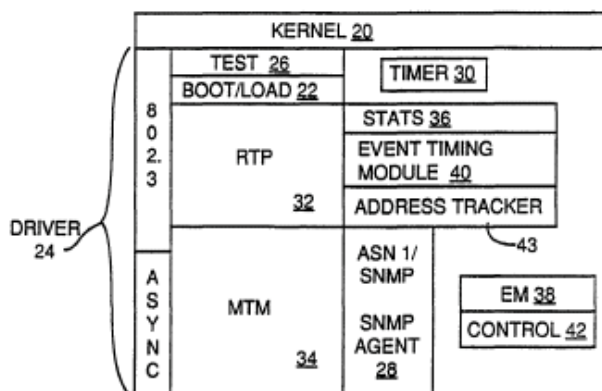


FIG 5

²⁰² Ferdinand, 12:3-16.

²⁰³ Ferdinand, 26:2-17.

²⁰⁴ Ferdinand, 19:8-12, 27:16-28:17.

154. As part of network monitor 10, Ferdinand discloses separate memories devoted to separate functionalities, such as a packet-buffer memory and system control blocks.²⁰⁵ And as shown below in Figures 5, Ferdinand describes network monitor 10 may include separate hardware components, including real time parser 32, boot loader 22, and database 36 for storing flow-entry information of the examined packets:²⁰⁶

155. The Challenged Patents, such as the '099 Patent, describe storing flow-entry information such as protocol identifiers, source and destination addresses, and hashes.²⁰⁷ Similarly, Ferdinand teaches storing flow-entry information that includes unique flow signatures for identifying subsequent flows.²⁰⁸ Ferdinand teaches this flow-entry information includes, for example, protocol identifiers (shown below in green), source and destination addresses (shown in yellow), and hashes (shown in blue). Ferdinand discloses storing flow-entry information in database 36.²⁰⁹

²⁰⁵ Ferdinand, 26:2-17.

²⁰⁶ Ferdinand, 19:8-12, 27:16-28:17.

²⁰⁷ '099 Patent, 13:22-36.

²⁰⁸ Ferdinand 34:11-20 (addresses and hashes), 36:13-22 (protocols).

²⁰⁹ Ferdinand, 26:15-18, 28:1-20, 35:34-36:4.

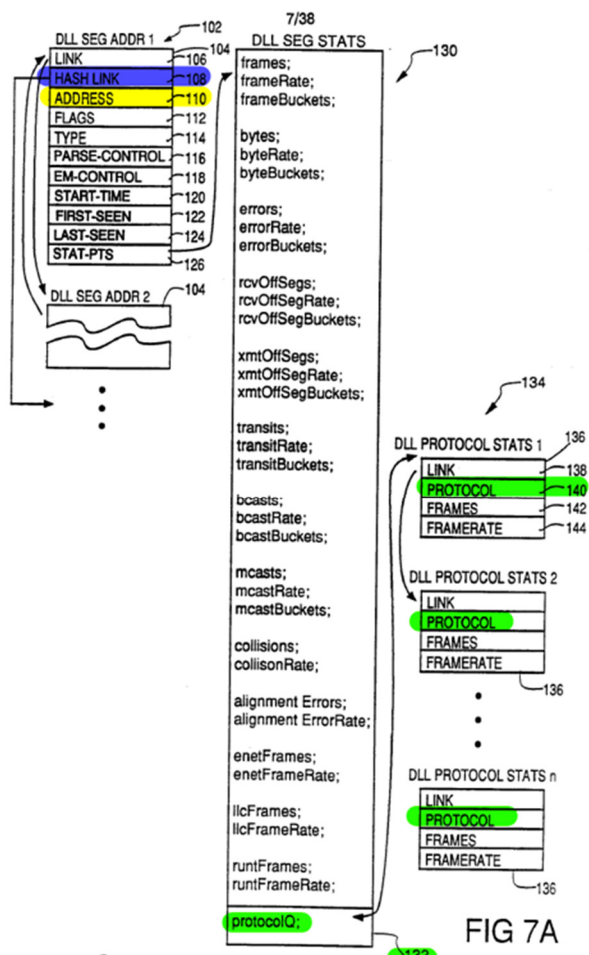


FIG 7A

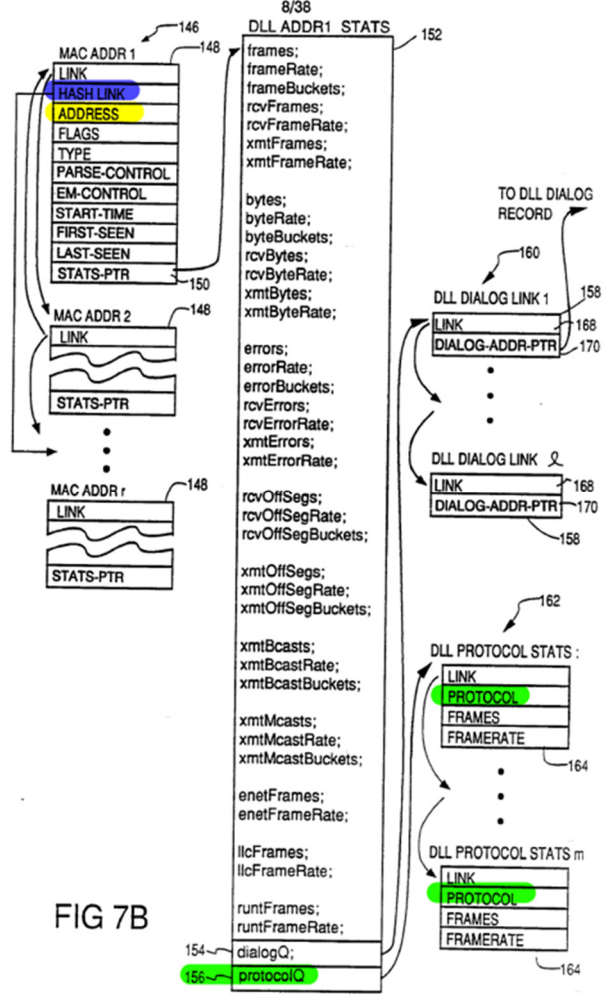


FIG 7B

SUBSTITUTE SHEET

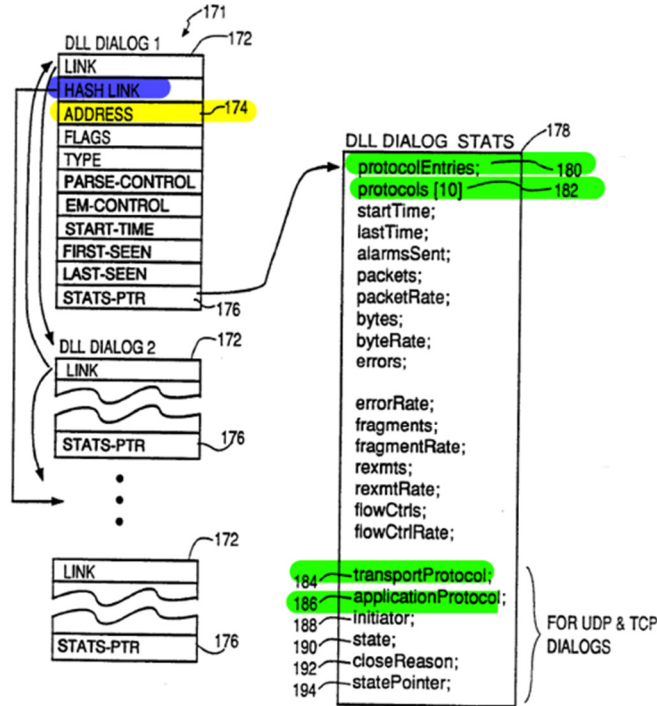


FIG 7C

156. Further, Ferdinand teaches classifying the same types of traffic, such as FTP and other protocol types like TCP and UDP.²¹⁰ And Ferdinand discloses displaying the results of its analysis to a user.²¹¹

C. Yu Overview

157. U.S. Patent No. 6,625,150 (“Yu”) is titled “Policy Engine Architecture.” Yu was not considered by the USPTO during the original prosecution of the Challenged Patents.

²¹⁰ Ferdinand, 29:4-30:10, 39:23-40:16; Riddle, 10:1-18 (Table 2 showing TCP and UDP protocols).

²¹¹ Riddle, 12:64-13:9, 14:1-5; Ferdinand, 60:10-15, Fig. 22.

158. As discussed with respect to Riddle, I understand that a U.S. patent has an effective prior art date under pre-AIA 35 U.S.C. §102(e) based on the filing date of an earlier-filed patent application if the patent's relevant subject matter is described in the earlier-filed application, and at least one of the patent's claims is supported by the earlier-filed application's written description in compliance with pre-AIA 35 U.S.C. §112, first paragraph.

159. The application that issued as Yu was filed on December 16, 1999. Yu claims priority to U.S. Provisional Patent Application No. 60/112,859 ("the '859 Provisional"), which was filed on December 17, 1998. I understand that Yu qualifies as prior art under at least 35 U.S.C. §102(e) prior art based on the '859 Provisional's filing date.

160. I understand that Exhibit 1047 is a table showing how Yu's disclosures are supported by the '859 Provisional's disclosures. This comparison shows that the two disclosures are substantially the same.

161. Further, Yu's single claim 1 is supported by the written description of the '859 Provisional in compliance with 35 U.S.C. §112, first paragraph. Exhibit 1048 is a table showing how the '859 Provisional's disclosures support Yu's single claim.

162. Yu sets forth a “policy engine for handling incoming data packets” in a network.²¹² Yu discloses that its “policy-based application examines every packet coming in from the network along the data path, compares it against flow classification criteria, and performs the necessary actions based upon the policies defined in a policy database.”²¹³ Further, Yu teaches that a flow classification criterion is “a rule to identify packets with a pattern of any random byte within a packet, and/or across many packets.”²¹⁴

163. As shown below in Figure 3, Yu’s policy architecture includes policy database 202 and flow classifier logic 204 which “uses the policy database 202 to determine the action specifications 203b that correspond to the policies of the flow to which the stream belongs.”²¹⁵

²¹² Yu, Abstract.

²¹³ Yu, 1:22-26.

²¹⁴ Yu, 1:50-54.

²¹⁵ Yu, 3:23-25, 4:46-55, Fig. 3.

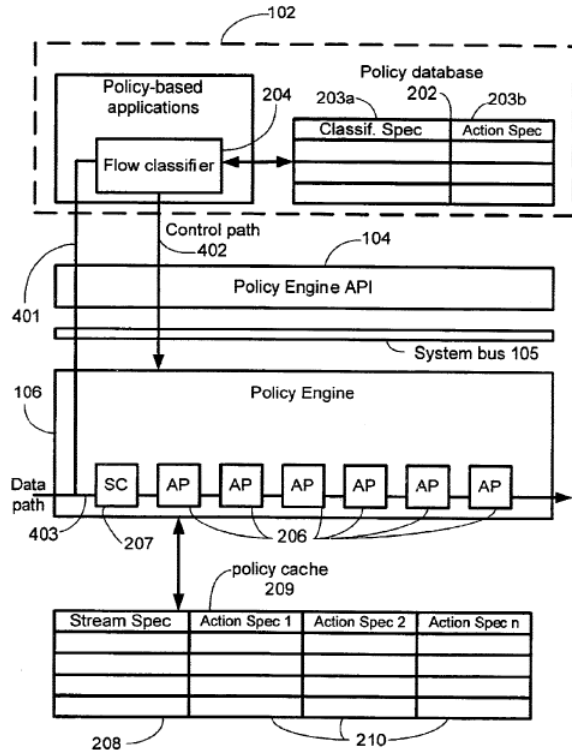


Fig. 3

D. RFC1945 Overview

164. RFC 1945 - Hypertext Transfer Protocol -- HTTP/1.0 ("RFC1945") is a printed publication available to the public on or around May 1996. RFC1945 was not considered by the USPTO during the original prosecution of the Challenged Patents. It is my understanding that RFC1945 is prior art to each of the Challenged Patents.

165. In my declaration, I discuss various publications that are "Request For Comments" (or "RFCs") prepared and distributed under a formalized publication process that is managed by the Internet Engineering Task Force ("IETF"). The IETF is

an open international community of network designers, operators, vendors, and researchers concerned with the evolution of the Internet architecture and protocols and the smooth operation of the Internet. IETF Working Groups perform the community's technical activities, which are organized by topic into several areas.

These working groups handle much of their work via mailing-lists to which anyone can subscribe. The IETF holds public meetings multiple times per year.

166. Often, an RFC publication is the result of a working group tasked with addressing a particular problem of interest. Each working group's work is in the open, and any networking community member is free to participate in these groups.

167. The IETF prepares and releases RFC publications to the public in a formalized and structured process, as described in RFC1543, dated October 1993, and RFC2026, dated October 1996. RFC1543 explains that RFCs are announced to members of mailing lists and distributed to the public online:

RFCs are distributed online by being stored as public access files, and a short message is sent to the distribution list indicating the availability of the memo.

The online files are copied by the interested people and printed or displayed at their site on their equipment. This means that the format of the online files must meet the constraints of a wide variety of printing and display equipment. (RFCs may also be returned via e-mail in response to an e-mail query, or RFCs may be found using information and database

searching tools such as Gopher, Wais, WWW, or Mosaic.)²¹⁶

168. Three years later, RFC2026 reaffirmed that RFC publications are widely disseminated on the Internet. For example, § 2.1 of RFC2026 explains:

Each distinct version of an Internet standards-related specification is published as part of the “Request for Comments” (RFC) document series. This archival series is the official publication channel for Internet standards documents and other publications of the IESG, IAB, and Internet community. RFCs can be obtained from a number of Internet hosts using anonymous FTP, gopher, World Wide Web, and other Internet document-retrieval systems.²¹⁷

169. RFC2026 further explains that, once a standard is adopted, it will be formally published:

If a standards action is approved, notification is sent to the RFC Editor and copied to the IETF with instructions to publish the specification as an RFC. The specification shall at that point be removed from the Internet-Drafts directory.²¹⁸

170. Each RFC document is uniquely numbered. The publication date of each RFC is contained in the RFC, typically in the top right corner of the first page of the document. Section 4a of RFC1543 describes:

This is the Month and Year of the RFC Publication. Indicated on the third

²¹⁶ Ex. 1040 (RFC1543), 2.

²¹⁷ Ex. 1041 (RFC2026), 5.

²¹⁸ Ex. 1041 (RFC2026), 17.

line on the right side.²¹⁹

171. Thus, before the priority date of the Challenged Patents and today, RFCs are well-known documents in the networking community and are commonly reviewed by people ranging from researchers, to industry professionals, to students, to government personnel. In my own research and teaching, I commonly track developments in various IETF working groups and obtain, read, and use both Internet Drafts as well as RFCs.

172. RFCs typically contain technical specifications and organizational notes and can cover many different topics, including standardized network protocols. RFCs most typically represent the definitive statement of, or definition of, a protocol, architecture, or use of a technology. For this reason, I, and numerous others, frequently cite to RFCs in our scholarly writing.

173. RFC1945 is the specification for version 1.0 of the HTTP protocol authored by T. Berners-Lee, R. Fielding, and H. Frystyk. RFC1945 states that it was published in May 1996. This is consistent with my recollection as to when the RFC for HTTP/1.0 was publicly available. Although I do not recall precisely when I first obtained a copy for RFC1945 (from a publicly accessible website), I am certain I

²¹⁹ Ex. 1040 (RFC1543), 6.

had obtained a copy before 1997.²²⁰ More generally, I am not aware of any publication date stated on an RFC being inaccurate. Thus, I understand that RFC1945 qualifies as prior art under at least pre-AIA 35 U.S.C. §102(b).

174. RFC1945 detailed the requirements, parameters, and overall operation of HTTP/1.0 for consideration by all those of skill in the art. Although the RFC, like all RFCs, solicited comments from interested parties and was not necessarily the final version of HTTP/1.0, it was the *de facto* standard until the RFC was updated by RFC2068, in January 1997, which formally defined version 1.1 of Hypertext Transfer Protocol (“HTTP”).

175. RFC1945 reflects common usage of the protocol referred to as “HTTP/1.0” (version 1.0 of the HTTP protocol).²²¹ HTTP has been in use by the World-Wide Web global information initiative since 1990. HTTP “is an application-level protocol with the lightness and speed necessary for distributed, collaborative, hypermedia information systems.”²²²

176. RFC1945’s Section 10 provides the “Header Field Definitions.”²²³ Section 10 “defines the syntax and semantics of all commonly used HTTP/1.0 header

²²⁰ Ex. 1044 (Internet-Archive Affidavit) (illustrating RFC1945 was publicly accessible and available for download at least by May 30, 1998).

²²¹ RFC1945, 1.

²²² RFC1945, 1.

²²³ RFC1945, 37.

fields.”²²⁴ Examples of HTTP header fields disclosed in RFC1945 include: “Content-Type,” “User-Agent,” and “Referer.”²²⁵ In RFC1945, “Referrer” was misspelled “Referer.” The standard’s later versions did not correct this misspelling.

177. RFC1945 describes the “Referer” as follows:

The Referer request-header field allows the client to specify, for the server’s benefit, the address (URI) of the resource from which the Request-URI was obtained. This allows a server to generate lists of back-links to resources for interest, logging, optimized caching, etc. It also allows obsolete or mistyped links to be traced for maintenance. The Referer field must not be sent if the Request-URI was obtained from a source that does not have its own URI, such as input from the user keyboard.

Referer = “Referer” “:” (absoluteURI | relativeURI)

Example:

Referer: <http://www.w3.org/hypertext/DataSources/Overview.html>

If a partial URI is given, it should be interpreted relative to the Request URI. The URI must not include a fragment.

Note: Because the source of a link may be private information or may reveal an otherwise private information source, it is strongly recommended that the user be able to select whether or not the Referer field is sent. For example, a browser client could have a toggle switch for browsing openly/anonymously, which would re-

²²⁴ RFC1945, 37.

²²⁵ RFC1945, 40-46.

spectively enable/disable the sending of Referer and From information.²²⁶

178. As discussed above, Patentee has alleged that the use of the Referer field results in a “conversational flow.” Notably, the HTTP Referer field has not changed in subsequent versions of HTTP.²²⁷

E. Baker Overview

179. WO 97/23076 (“Baker”) is titled “System and Method for General Purpose Network Analysis,” and published on June 26, 1997. Baker was not considered by the USPTO during the original prosecution of the Challenged Patents. It is my understanding that Baker is prior art to each of the Challenged Patents.

180. Baker’s teachings relate to “a network interface system” that has “one or more programmably configurable protocol descriptions which may be stored in and retrieved from an associated memory.”²²⁸ Baker further described its system as being directed to:

[I]mproved systems and methods for parsing, filtering, generating and analyzing data (or frames of data) transmitted over a data communications network. In one particularly innovative aspect of the present invention, a single logic control module, which may be implemented in hard-

²²⁶ RFC1945, 44-45.

²²⁷ Ex. 1042 (RFC2616), 140-41.

²²⁸ Baker, Abstract.

ware or software, is utilized to perform any of a number of data manipulation functions (for example, parsing, filtering, data generation or analysis functions) based upon *one or more programmably configurable protocol descriptions which may be stored in and retrieved from an associated memory.*²²⁹

181. Baker also discloses that its approach accommodates changes to existing network protocols and newly added protocols:

The use of common control logic (i.e. the use of a single logic control module) and programmably configurable protocol descriptions allows changes to existing protocols to be made and support for new protocols to be added to a system in accordance with the present invention through configuration only--without the need for hardware and/or software system modifications. Thus, those skilled in the art will appreciate that a network interface in accordance with the present invention may be configured and reconfigured, if necessary, in a highly efficient and cost effective manner to implement numerous data manipulation functions and to accommodate substantial network modifications (for example, the use of different data transmission hardware, protocols or protocol suites) without necessitating substantial system changes.²³⁰

182. Baker further discloses storing protocol descriptions as “protocol description files (PDF).”²³¹ As provided below, Baker’s Figure 1 shows database 14 of PDFs

²²⁹ Baker, 3:32-4:6.

²³⁰ Baker, 4:7-21.

²³¹ Baker, 19:6-10.

22, which network device control logic 16 uses to retrieve network frames based on extracted field values and filtering criteria contained in PDFs 22.²³²

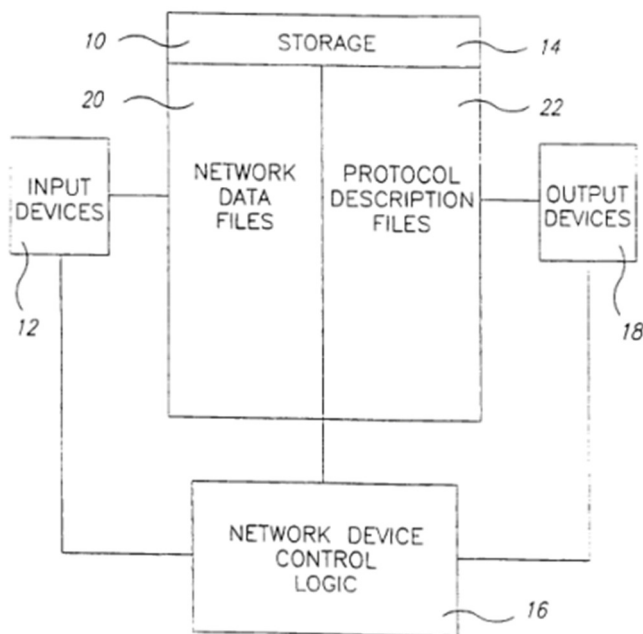


FIG. 1

183. Baker's PDFs include a protocol control record that defines the overall structure of a network protocol and references other information relating to the network protocol.²³³ Each PDF includes (i) a total bit length of the protocol header; (ii) a number of fields required to describe the header; and (iii) field records, each describing a protocol header field, including a byte offset from the start of the protocol header and, if appropriate, an associated lookup structure for determining the next protocol control record to use.²³⁴

²³² Baker 10:10-35, Fig. 1.

²³³ Baker, 12:25-32.

²³⁴ Baker, 12:25-15:17, Tables 1, 2, and 4.

184. Upon system initialization, a “ProtocolList” is constructed from the protocol and associated control record information extracted from all PDFs.²³⁵ The ProtocolList is a sorted vector of all protocol records.²³⁶

F. Wakeman Overview

185. U.S. Patent No. 5,740,175 (“Wakeman”) is titled “Forwarding Database Cache For Integrated Switch Controller,” and issued on April 14, 1998. Wakeman was not considered by the USPTO during the original prosecution of the Challenged Patents. It is my understanding that Wakeman is prior art under at least 35 U.S.C. §102(b) to each of the Challenged Patents.

186. Wakeman describes a network switch that includes a content addressable memory (CAM) cache:

A LAN network switch includes a RAM forwarding database which contains the address-to-port mappings for all the workstations or other devices connected to the switch’s plurality of ports and further includes at least one CAM-cache connected to respective one or more of the switch’s ports. The CAM-cache, having an access time much faster than that of the forwarding database, stores selected ones of the address-to-port mappings. When it is desired for the switch to forward a packet, the destination address is extracted and the CAM-cache is accessed and searched. If the correct mapping is contained in the CAM- cache, the packet is immediately forwarded

²³⁵ Baker, 20:25-21:11.

²³⁶ Baker, 20:25-21:11.

to the destination port without accessing the much larger and slower forwarding database. Only if the CAM-cache does not contain the correct mapping is the forwarding database accessed to retrieve the correct mapping. The packet is then forwarded to the destination port, and the CAM-cache is updated with this mapping so that succeeding packets having the same destination address-to-port mapping may be forwarded to the destination port by accessing only the fast CAM-cache and, by eliminating the need to access the much slower forwarding database, increasing the forwarding speed of the switch.²³⁷

187. Wakeman discusses prior-art network switch 10 as having forwarding database (FSB) 12, which is shown below in Figure 1. Wakeman explains that databases, such as FSB 12, are “typically implemented either as a hardware content addressable memory (CAM) or as RAM.”²³⁸ And Wakeman states that a hardware CAM “is very fast and can typically retrieve mappings in less than 100 ns. A RAM, on the other hand, requires a searching algorithm and typically requires several micro- seconds to locate the correct mapping and, thus, is typically too slow to keep up with [switch engine] SE 11.”²³⁹

²³⁷ Wakeman, Abstract.

²³⁸ Wakeman, 1:55-56.

²³⁹ Wakeman, 1:56-61.

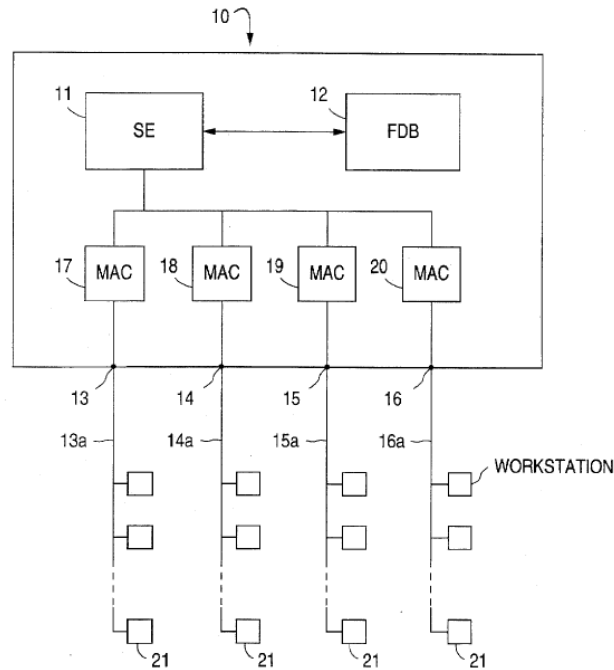


FIG. 1
(PRIOR ART)

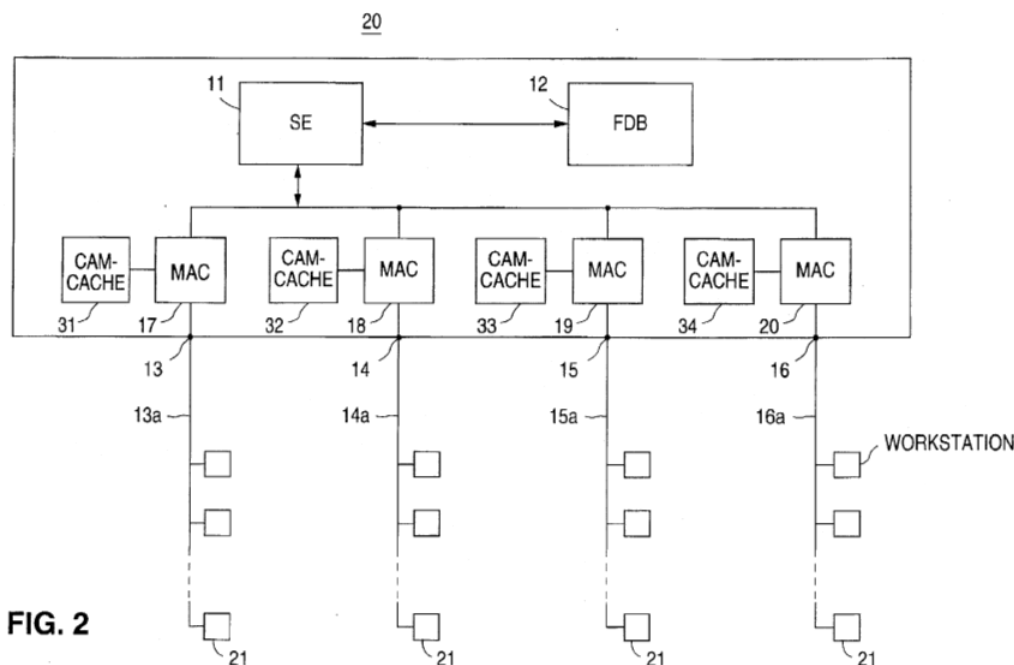
188. Wakeman discloses improving a CAM system by including a CAM cache:

Upon receiving an incoming data packet, the MAC associated with the source port will, after extracting the destination address from the packet, access its associated CAM-cache to find the correct address-to-port mapping. If the correct mapping is contained in the CAM-cache, the packet may be immediately forwarded to the destination port without having to access the much larger and slower forwarding database.

Where the CAM-cache does not contain the correct mapping, the MAC then causes the correct mapping to be retrieved from the forwarding database. The packet may then be forwarded to the correct destination port. The CAM-cache is then updated with this mapping so that succeeding packets having the same destination address-to-port mapping may be

quickly forwarded to the destination port by accessing only the fast CAM-cache, thereby eliminating the need to access the much slower forwarding database.²⁴⁰

189. As shown below in Figure 2, Wakeman teaches network switch 20 having switch engine 11, forwarding database 12, media access controllers 17-20, and CAM-Caches 31-34.²⁴¹



190. Wakeman describes that “CAM caches 31-34 may be distributed across ports 13-16 of switch 20, where one of the CAM caches described above may service more than one port.”²⁴² And Wakeman describes the CAM cache’s internal architecture and operation as follows:

²⁴⁰ Wakeman, 2:31-51.

²⁴¹ Wakeman, 1:20-28, 3:26-28, Fig. 2.

²⁴² Wakeman, 5:28-31.

CAM cache 31 which, in accordance with the preferred embodiment, includes a FIFO 35, a memory 36, a learning and aging block 38, and logic 37. These elements are well understood in the art and thus will not be discussed below. The extracted source and destination addresses of the first packet are queued in FIFO 35 which, in turn, provides the destination address to memory 36. If the correct destination mapping is contained in memory 36, there is thus a match and memory 36 provides the correct destination port to logic 37 which, in turn, forwards the port location and a “hit” signal to MAC 17. MAC 17 then alerts SE 11 of the correct destination port. SE 11 informs MAC 18 that a packet is “addressed” thereto and directs the first packet to MAC 18 which, in turn, forwards the packet to segment 14a where it will be accepted by the workstation having the correct destination address. Thus, where the correct destination mapping is contained in CAM cache 31, accessing and searching FDB 12 is wholly unnecessary. Since the accessing speed of CAM cache is much faster than that of FDB 12, the inclusion of CAM caches 31-34 in a network switch as described above results in an increase in forwarding speed. Note that although the FDB 12 in switch 20 is preferably a RAM, CAM caches 31-34 will decrease the access time and thus increase forwarding speeds irrespective of the particular construction of FDB 12 (e.g., where FDB 12 is a hardware CAM as opposed to RAM).

If the correct destination mapping is not contained in memory 36, logic 37 sends a “miss” signal to MAC 17 which then alerts SE 11 of the destination address extracted from the packet. SE 11 then searches FDB 12 to locate the correct destination mapping and, having retrieved the correct destination mapping, forwards the packet as described earlier with reference to

prior art switch 10.²⁴³

191. As provided below, Wakeman's Figure 3 illustrates the internal architecture of the CAM-Cache.

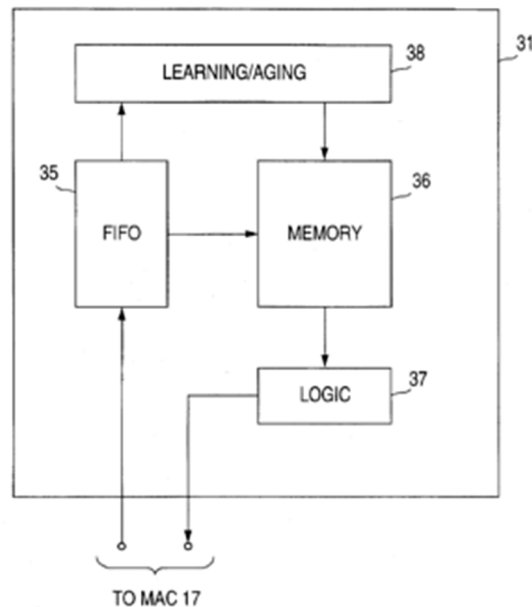


FIG. 3

G. Hasani Overview

192. U.S. Patent No. 5,805,808 (“Hasani”) is titled “Real Time Parser For Data Packets In A Communications Network.” The patent application that issued as Hasani was filed on April 9, 1997 and ultimately issued on September 8, 1998.

Hasani claims to be a continuation of U.S. patent applications filed on December 29, 1994 and December 27, 1991. I understand that Riddle qualifies as prior art under at least 35 U.S.C. §102(a) and §102(e) based on its filing and issue dates and

²⁴³ Wakeman, 3:54-4:20.

the filing dates of the related applications. It is my understanding that Hasani is prior art to each of the Challenged Patents.

193. The '099, '725, '646, and '789 Patents list Hasani in their respective References Cited sections. But it is my understanding that Hasani was not cited in any rejection by the Examiners during the prosecution of the Challenged Patents.

194. Hasani teaches a packet parser and details that it was known in the art to store parsing and extraction operations in a database.²⁴⁴ Hasani describes that its parser database contains memory locations for different parsing and extraction operations.²⁴⁵ For example, Hasani shows how its parsing/extraction operations parse and extract information from an IEEE 802.2 medium access control (MAC) packet.²⁴⁶ And Hasani discloses that its teachings apply to “any standard packet format.”²⁴⁷

V. A PERSON OF ORDINARY SKILL IN THE ART

195. It is my understanding that I must analyze and apply the prior art cited above from the perspective of a person having ordinary skill in the art (“POSITA”) as of the time of the invention, which I understand is June 30, 1999 for purposes of IPR proceedings. When forming my opinions herein, I analyzed and applied the prior

²⁴⁴ Hasani, 8:1-53, Figures 3, 4 (parser database 182).

²⁴⁵ Hasani, 6:12-57, Fig. 5B.

²⁴⁶ Hasani, 5:9-24, 6:27-57, Fig. 5B.

²⁴⁷ Hasani, 5:16-20.

art and claim constructions from the perspective of a POSITA as of the time of the invention.

196. I understand that Patentee has previously taken the position:

[A POSITA] in the late 1990s would have the equivalent of a four-year degree from an accredited institution (usually denoted as a B.S. degree) in computer science, computer engineering or the equivalent and experience with, or exposure to, packet analysis techniques. A [POSITA] would also have approximately 1-2 years of professional experience with packet network communication protocols. Additional graduate education could substitute for professional experience, while significant experience in the field might substitute for formal education.²⁴⁸

197. I also understand that the Board previously determined in IPR proceedings that a POSITA “had a bachelor’s degree in electrical engineering, computer engineering, computer science, or a related field (or its equivalent), and one to two years of experience working in networking environments, including at least some experience with network traffic monitors and/or analyzers.”²⁴⁹

198. For purposes of my analysis herein, I agree with the Board’s determination regarding a POSITA’s education and experience.

199. I am well aware of the qualifications of such a skilled artisan because I have worked with, supervised, and hired engineers with similar capabilities. By the year

²⁴⁸ E.g., Ex. 1050 (IPR2017-00450 Prelim. Resp.), 27-28.

²⁴⁹ E.g., Ex. 1056 (IPR2017-00450 Institution Decision), 13-14.

1999, I had been awarded a Ph.D. in computer science and had several years of practical experience in both industry (5 years) and academia (10 years, including M.S. and Ph.D.). As of the year 1999, I was teaching and working with individuals who met the above criteria of persons of ordinary skill in the art. In particular, I have taught and worked with distinct groups of graduate students. These students and colleagues possessed basic knowledge regarding networking environments, including at least some experience with network traffic monitors and/or analyzers. One group entered the graduate program with B.S. degrees in CS/CE/EE and several years of industry training (3 years was typical). Another group entered the program with an M.S. degree to obtain a Ph.D. having gained significant experience during their M.S. coursework. Finally, I have worked with and taught advanced Ph.D. students that had at least two years of post-BS experience and knowledge gained while in the graduate program. During my time in industry, many of my colleagues possessed at least a B.S. in the relevant fields and had several years of work experience.

200. These students and colleagues possessed knowledge of networking environments, including some experience with network traffic monitors and/or analyzers. Further, many of these students ultimately found employment at companies that had an expressed interest in and need for skills relating to computer networking environments, particularly network traffic monitoring and analysis, in this time

frame, further corroborating that these were ordinarily skilled artisans.

201. Thus, I am familiar with the understanding and knowledge of persons of ordinary skill in the art as of 1999, and was at least as qualified as the POSITA that I have identified above. Thus, I understand the perspective of a POSITA, which I have applied in my analysis.

VI. CLAIM CONSTRUCTION

202. For IPR petitions filed after November 13, 2018, I understand the Board construes claim terms as having their ordinary and customary meaning to a POSITA at the time of invention (i.e., the same standard used in district court). I have applied this standard in rendering my opinions on claim construction. In addition, I have been asked to assume that the claims are not indefinite.

203. For IPR petitions filed before November 13, 2018, I understand the Board construed claim terms as having their broadest reasonable interpretation to a POSITA at the time of invention. Because Sandvine filed its IPR petitions in 2017, the Board used the broadest reasonable interpretation standard in those IPRs.²⁵⁰

A. “Conversational Flow” / “Conversational Flow-Sequence”

204. The terms “conversational flow” and “conversational flow-sequence” appear in every independent Challenged Claim, as well as many of their dependents.

205. I understand that Palo Alto Networks has argued in district court that due to

²⁵⁰ E.g., Ex. 1062 (IPR2017-00863 Institution Decision).

contradictory positions Patentee has taken regarding the scope of this term, the term is indefinite as arguably susceptible to a broader (but indeterminate) scope than I understand Petitioners are proposing here. As noted above, I have been asked to assume the claims are not indefinite. I have also been asked to assume that statements made in the intrinsic record that clearly and unambiguously limit the meaning and scope of “conversational flow” have a limiting effect. On that basis, and in view of the disclosures in the Challenged Patents, their prosecution histories, and prior IPR proceedings, it is my opinion that a POSITA would have understood the terms “conversational flow” and “conversational flow-sequence” to mean:

The sequence of packets that are exchanged in any direction as a result of specific software program activity, where such packets form multiple connection flows that are linked based on that activity.

206. This construction is consistent with the specifications of the Challenged Patents. For example, the '099 Patent, which is incorporated-by-reference by each of the other Challenged Patents, describes the difference between a “connection flow” and a “conversational flow” as follows:

The term “connection flow” is commonly used to describe all the packets involved with a single connection. A conversational flow, on the other hand, is the sequence of packets that are exchanged in any direc-

tion as a result of an activity—for instance, the running of an application on a server as requested by a client.²⁵¹

What distinguishes this invention from prior art network monitors is that it has the ability to recognize disjointed flows as belonging to the same conversational flow.²⁵²

207. Consistent with the Challenged Patents, the above construction differentiates “conversational flow” from a “connection flow.”

208. In addition, Patentee has explicitly described that the exchanging of packets for a “conversational flow” results from specific software program activity:

Essentially, the network monitor disclosed in the '099 patent categorizes network transmissions into “conversational flows,” which relate individual packets and connection flows based on *specific application activity*.²⁵³

An “*application*” is a *software program* that runs on a computer, for example, a web browser, word processor, Skype, etc.²⁵⁴

As noted above, the conversational flows of the '099 patent relate packets, and ultimately connection flows, when they are the result of an application activity.²⁵⁵

A conversational flow relates packets and flows between the client

²⁵¹ E.g., '099 Patent, 2:35-40.

²⁵² '099 Patent, 3:47-51.

²⁵³ Ex. 1054 (IPR2017-00769 Prelim. Resp.), 8, 50.

²⁵⁴ Ex. 1049 (IPR2017-00769 Opp'n to Request for Rehearing), 13.

²⁵⁵ Ex. 1054 (IPR2017-00769 Prelim. Resp.), 40.

and server as related to specific application activities. For example, with conversational flows, packets and flows related to one video stream are distinguished from a different video stream, even under the same protocol, from a video conference, or from Facebook—assuming all of the applications call to the same server.²⁵⁶

[T]he conversational flows claimed require processing the information in each packet column (i.e., across OSI levels within a packet) and determining if a given packet is related to existing flows. For example, *Packet 1 and Packet 2, above, may be video and audio traffic, respectively, originating from a Skype call*, whereas Packet 3 might be generated by a different application (for example, an email message) and Packet 4 from still another application (for example, a web browser). *Packets 1 and 2 are related because they are data streams originating from the same instance of an application (i.e., Skype)*, where Packets 3 and 4 are data packets from separate applications.²⁵⁷

209. While the above statements are directed to the prior IPR proceedings regarding the '099 Patent, Patentee made the same or substantially similar statements in the other prior IPR proceedings.²⁵⁸

210. Further, in prior IPR proceedings, Patentee has acknowledged that a “conversational flow” is more than merely a single connection or “connection flow”:

²⁵⁶ Ex. 1054 (IPR2017-00769 Prelim. Resp.), 55.

²⁵⁷ Ex. 1054 (IPR2017-00769 Prelim. Resp.), 46-47.

²⁵⁸ E.g., Ex. 1050 (IPR2017-00450), 47, 51-52; Ex. 1051 (IPR2017-00451 Prelim. Resp.), 52, 57; Ex. 1053 (IPR2017-00630), 44-45, 46, 49; Ex. 1055 (IPR2017-00862), 43-44, 47, 51-52.

The '099 patent treats packets as complete units, such that information is extracted from the packets, entire packets are related to each other as part of a connection flow, ***and ultimately connection flows are related to each other when they are part of an application activity (i.e., a conversational flow)***.²⁵⁹

According to the Patentees: “[i]n order to eliminate the possibility of disjointed conversational exchanges, it is desirable for a network packet monitor to be able to ‘***virtually concatenate***’—***that is, to link—the first exchange with the second***. If the clients were the same, the two packet exchanges would then be correctly identified as being part of the same ***conversational flow***.”²⁶⁰

[Prior art] stands in stark contrast with the concept of “connection flows” and “conversational flows” in '099 patent where each packet is part of a single connection flow, and ***different connection flows are related to each other into conversational flows***.²⁶¹

[... T]o monitor those flows to ***establish relationships between individual flows to create conversational flows*** taught in the '099 patent.”²⁶²

The problem with only tracking connection flows is that certain applications and protocols may generate multiple connections. In other words, a single application may spawn multiple connections for a single

²⁵⁹ E.g., Ex. 1054 (IPR2017-00769 Prelim. Resp.), 43-44.

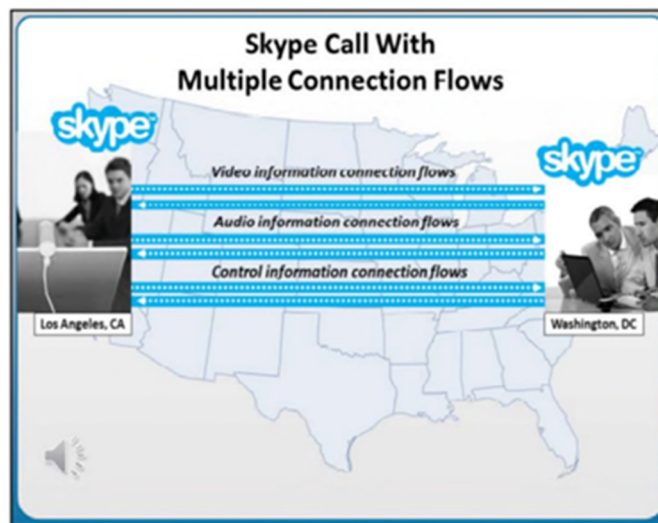
²⁶⁰ E.g., Ex. 1054 (IPR2017-00769 Prelim. Resp.), 16 (citing '099 Patent, 3:1-6).

²⁶¹ E.g., Ex. 1054 (IPR2017-00769 Prelim. Resp.), 45.

²⁶² E.g., Ex. 1054 (IPR2017-00769 Prelim. Resp.), 48.

activity.²⁶³

211. In a prior tutorial to a district court, Patentee used the software program Skype to demonstrate the alleged problem in the art the Challenged Patents addressed.²⁶⁴ Citing the below two figures, Patentee stated that a given Skype call generates multiple separate connection flows for video, audio, and control information. And Patentee asserted that linking those separate connection flows based on that specific software program activity (i.e., the Skype call) creates one “conversational flow.”

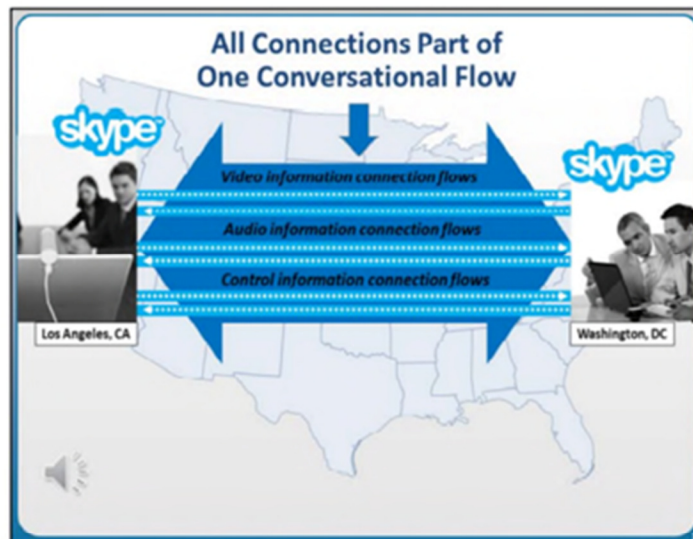


Some of the differences between connection flows and conversational flows can be illustrated through our Skype call example. During the Skype call, the Skype application may use multiple connections to facilitate the call. For example, it may use a connection for sending control information, a connection for sending audio data, and a connection for sending video data.

Patentee’s Tutorial Slide in NetScout District Court Case

²⁶³ E.g., Ex. 1054 (IPR2017-00769 Prelim. Resp.), 16.

²⁶⁴ Ex. 1066 (Patentee’s Tutorial in *NetScout* district court case), 18-19.



Each of these connections maps to a separate connection flow because each connection uses a different address and port for the source and destination. However, all of these connections are part of the same conversational flow because all of the connections facilitate the application-level activity of making a Skype call.

Patentee’s Tutorial Slide in NetScout District Court Case

212. In current district-court proceedings, I understand that the Patentee has proposed “conversational flow” means:

[T]he sequence of packets that are exchanged in any direction as a result of an activity—for instance, the running of an application on a server as requested by a client—and where some conversational flows involve more than one connection, and some even involve more than one exchange of packets between a client and server.²⁶⁵

²⁶⁵ 1079 (Joint Claim Construction and Prehearing Statement in *Palo Alto Network* district court case); Ex. 1067 (Claim Construction Memo and Order in *NetScout* district court case) (Patentee adopting substantially similar construction); Ex. 1054 (IPR2017-00769 Prelim. Resp.), 29;.

213. However, the Patentee’s proposed construction does not properly differentiate a “conversational flow” from either a “connection flow.” And that proposed construction includes optional examples: “... *for instance*, the running of an application on a server as requested by a client ... *some* conversational flows involve more than one connection, and *some* even involve more than one exchange of packets between a client and server.” After removing those optional examples, Patentee’s proposal distills to “the sequence of packets exchanged as a result of an activity.” This does not differentiate “conversational flow” from “flow” or “connection flow.”

214. The prior art discussed below invalidates the Challenged Patents under both proposed constructions. As my below analysis demonstrates, the prior art teaches or renders obvious a “conversational flow” forming multiple connection flows that are linked by specific software program activity or under Patentee’s proposed construction.

B. “State Of The Flow” / “State Of The Conversational Flow”

215. The term “state of the flow” appears in the following Challenged Claims: ’099 claims 1 and 2; ’725 claim 16; ’646 claim 1; ’751 claims 1 and 10; and ’789 claims 13 and 44. The term “state of the conversational [flow/flows]” appears in the following Challenged Claims: ’099 claim 1 and ’725 claim 17.

216. I understand that in prior IPR proceedings, Patentee asserted that “state of

the flow” means “an indication of all previous events in the flow that lead to recognition of the content of all of the protocol levels.”²⁶⁶ Patentee asserted the same meaning for “state of the conversational flow.”²⁶⁷ In support of this meaning, Patentee wrote:

[D]epending on a packet’s relationship to other previously encountered packets, the disclosed monitor will perform certain operations: The exact operation of the state processor and functions performed by it will vary depending on the current packet sequence in the stream of a conversational flow. The state processor moves to the next logical operation stored from the previous packet seen with this same flow signature. If any processing is required on this packet, the state processor will execute instructions from a database of state instruction for this state until there are either no more left or the instruction signifies processing.²⁶⁸

217. At the time of the purported invention, it was well known in the art that a “state of the flow” describes the status of the flow based on prior events in the flow. When discussing the “state of the flow,” the Challenged Patents specify: “[o]ne aspect of monitor 108 is its ability to maintain the state of a flow. The state of a flow is an indication of all previous events in the flow that lead to recognition

²⁶⁶ Ex. 1050 (IPR2017-00450 Prelim. Resp.), 31; Ex. 1051 (IPR2017-00451 Prelim. Resp.), 32; Ex. 1052 (IPR2017-00629 Prelim. Resp.), 31; Ex. 1053 (IPR2017-00630 Prelim. Resp.), 31; Ex. 1054 (IPR2017-00769 Prelim. Resp.), 31; Ex. 1055 (IPR2017-00862 Prelim. Resp.), 30-31.

²⁶⁷ Ex. 1055 (IPR2017-00862 Prelim. Resp.), 30-31.

²⁶⁸ E.g., Ex. 1050 (IPR2017-00450 Prelim. Resp.), 30-31.

of the content of all the protocol levels, e.g., the ISO model protocol levels.”²⁶⁹ The Challenged Patents also disclose that this indication of previous events includes parameters like time, length of the conversational flow, and data rate:

The preferred embodiment forms and remembers the state of any conversational flow, which is determined by the relationship between individual packets and the entire conversational flow over the network. By remembering the state of a flow in this way, the embodiment determines the context of the conversational flow, including the application program it relates to and parameters such as the time, length of the conversational flow, data rate, etc.²⁷⁰

218. My analysis herein is the same if Patentee’s construction is used or if the ordinary and customary meaning is used.

C. “The Flow” / “New Flow” / “Existing Flow”

219. The term “the flow” appears in the following Challenged Claims: ’099 claims 1-2; ’725 claim 16; ’646 claim 1; ’751 claims 1, 5, 10, and 15; and ’789 claims 13-14, 16, and 44. The term “new flow” appears in ’646 claims 1, 7, and 16; ’751 claims 1, 10, and 17; and ’789 claims 1, 13, 19, and 44. Further, the term “existing flow” appears in ’646 claims 1, 7, and 16; ’751 claims 1, 10, and 17; and ’789 claims 1, 13, 19, and 44.

220. In view of the language and logical operation of the Challenged Claims, the

²⁶⁹ E.g., ’099 Patent, 10:28-32.

²⁷⁰ E.g., ’099 Patent, 5:27-34.

prosecution history, and prior IPR proceedings, it is my opinion that a POSITA would have these terms to respectively mean “the conversational flow,” “new conversational flow,” and “existing conversational flow.”

221. With respect to “the flow,” the only logical antecedent for this term is “conversational flow” previously recited in the Challenged Claims. For example, the last line of ’725 claim 10 recites “the packet as belonging to a conversational flow,” and ’725 claim 16, which depends from claim 10, further recites “the state of the flow of the packet.” The only logical antecedent for ’725 claim 16’s “the flow” is the previously recited “conversational flow.”

222. With respect to “new flow” and “existing flow,” the logical operation of the claims results in each of these flows referring to a “conversational flow.” For example, ’646 claim 16 recites looking up a database of “previously encountered conversational flows to which a received packet may belong.” The consequence of this look up is a determination whether the received packet belongs to an existing conversational flow (included in the database) or a new conversational flow (not included in the database). The subsequent conditional operations recited in the claim depend on this determination: “(d) if the packet is of an existing flow, classifying the packet...(e) if the packet is of a new flow, storing a new flow entry for the new flow.” This logical operation also applies to the other claims reciting “new

flow” and “existing flow.”²⁷¹

223. Further, the Patentee confirmed this understanding for “new flow” and “existing flow” during the prosecution of the ’646 Patent:

The analyzer subsystem ..., for each packet, looks up a database of flow records for *previously encountered conversational flows* to determine whether a signature is from an existing flow. The present invention described the lookup mechanism in more detail. In one aspect, the analyzer further *identifies the state of the existing flow*, and performs any state processing operations specified for the state. In the case of a *newly encountered flow*, the analyzer includes a flow insertion and deletion engine for inserting new flows into the database of flows.²⁷²

224. During the prosecution of the ’751 Patent, the Patentee also confirmed this understanding for “[new/existing] flow”:

An aspect of the present invention includes, for any packet ascertained to belong to an existing flow by looking up the database, identifying the state of the flow Anderson has no concept of state of the flow, or even of a conversational flow, so that no such state operations are therefore carried out....

Furthermore, Applicant’s lookup is to determine if a packet is part of an existing conversational flow.²⁷³

²⁷¹ E.g., ’646 claims 1 and 7; ’751 claims 1, 17; ’789 claims 1, 19, 44.

²⁷² Ex. 1020 (2/10/2004 Office Action Response), 8.

²⁷³ Ex. 1021 (11/03/2003 Office Action Response, 9-11), 228, 230-31. Ex. 1020 (2/10/2004 Office Action Response for ’646 Patent), 8.

225. Additionally, in prior IPR proceedings, the Board found that the claimed “new flow” and “existing flow” mean “new conversational flow” and “existing conversational flow”:

Similar to [’646] claim 1, independent claims 7 and 16 each recite a database of flow-entries for previously encountered “conversational flows ...” [C]laim [16] requires “conversational flows” at least due to the language of limitations (d) and (e). *Limitation (d) covers the situation where there is an “existing” conversational flow and limitation (e) covers storing a new flow-entry in the database for a “new” conversational flow.*²⁷⁴

226. Moreover, in another prior IPR proceedings, the Board found that the claimed “new flow” and “existing flow” mean “new conversational flow” and “existing conversational flow”:

[’789 C]laim 19 ... requires “conversational flows” at least due to the language of limitation (f). *Limitation (f) covers the situation where there is an “existing” conversational flow and the situation of storing a new flow-entry in the database for a “new” conversational flow.*²⁷⁵

227. In current district-court proceedings, I understand that the Patentee has proposed each of these terms to mean “[n]o construction necessary; Alternatively: a

²⁷⁴ Ex. 1056 (IPR2017-00450 Institution Decision), 24, n.9.

²⁷⁵ Ex. 1058 (IPR2017-00629 Institution Decision), 11, n.5.

stream of packets being exchanged between any two addresses in the network”²⁷⁶

My analysis herein is the same if Patentee’s construction is used or if “the conversational flow,” “new conversational flow, and “existing conversational flow” are used.

D. “State Operations” / “State Processing Operations”

228. The term “state operations” or “state processing operations” appears in the following Challenged Claims: ’099 claim 1; ’725 claims 16-17; ’646 claim 1; ’751 claims 1, 10, and 14; and ’789 claims 13, 15, 17, 44, and 49.

229. I understand that in prior IPR proceedings, Patentee asserted that “state operation” means “an operation to be performed while the state processor is in a particular state.”²⁷⁷

230. Regarding “state operations,” the Challenged Patents disclose:

The state processor 328 analyzes both new and existing flows in order to analyze all levels of the protocol stack, ultimately classifying the flows by application (level 7 in the ISO model). It does this by proceeding from state-to-state based on predefined state transition rules and state operations as specified in state processor instruction database 326. *A state transition*

²⁷⁶ Ex. 1079 (Joint Claim Construction and Prehearing Statement in *Palo Alto Network* district court case).

²⁷⁷ Ex. 1050 (IPR2017-00450 Prelim. Resp.), 34; Ex. 1051 (IPR2017-00451 Prelim. Resp.), 34; Ex. 1052 (IPR2017-00629 Prelim. Resp.), 33; Ex. 1053 (IPR2017-00630 Prelim. Resp.), 33; Ex. 1054 (IPR2017-00769 Prelim. Resp.), 33; Ex. 1055 (IPR2017-00862 Prelim. Resp.), 34 (addressing “state processing operations”).

*rule is a rule typically containing a test followed by the next-state to proceed to if the test result is true. An operation is an operation to be performed while the state processor is in a particular state—for example, in order to evaluate a quantity needed to apply the state transition rule. The state processor goes through each rule and each state process until the test is true, or there are no more tests to perform.*²⁷⁸

231. Patentee has acknowledged that “the claims themselves often define specific state operations required for a given claim.”²⁷⁹ For example, the following are state operations recited in the claims of the Challenged Patents:

- ’646 claim 13, and ’789 claim 27: “wherein the set of possible state operations that the state processor is configured to perform includes *searching for one or more patterns in the packet portions*”;
- ’646 claim 15, ’751 claim 14, ’789 claim 15, and ’789 claims 30, 45: “wherein the state operations include *updating the flow-entry*, including [storing] identifying information for future packets to be identified with the flow-entry”;
- ’751 claim 13: “wherein the *reporting* is part of the state operations for the state of the flow”;
- ’751 claim 15, and ’789 claims 14 and 16: “wherein the state processing of each received packet of a flow *further the identifying of the application program of the flow*”;

²⁷⁸ E.g., ’099 Patent, 14:63-15:9.

²⁷⁹ Ex. 1050 (IPR2017-00450 Prelim. Resp.), 34; Ex. 1051 (IPR2017-00451 Prelim. Resp.), 34-35; Ex. 1052 (IPR2017-00629 Prelim. Resp.), 33; Ex. 1053 (IPR2017-00630 Prelim. Resp.), 33; Ex. 1054 (IPR2017-00769 Prelim. Resp.), 33; Ex. 1055 (IPR2017-00862 Prelim. Resp.), 34.

- ’751 claim 16: “wherein one or more *metrics related to the state of the flow are determined* as part of the state operations specified for the state of the flow”;
- ’789 claims 17 and 46: “wherein the state operations include *searching the parser record for the existence of one or more reference strings*”;
and
- ’789 claim 47: “wherein one of the state operations specified for at least one of the states includes *creating a new flow-entry* for future packets to be identified with the flow, the new flow-entry including identifying information for future packets to be identified with the flow-entry.”

232. My analysis herein is the same if Patentee’s construction is used or if the ordinary and customary meaning is used.

E. “Flow-Entry Database ...” Terms

233. The “flow-entry database ...” terms appear in the following Challenged Claims: ’099 claim 1; ’646 claims 1, 7, and 16; ’751 claim 1 and 17; and ’789 claims 1, 19, 33, and 44.

234. In view of the disclosures in the Challenged Patents, their prosecution histories, and prior IPR proceedings, it is my opinion that a POSITA would have understood the “flow-entry database ...” terms to require a “database having a separate entry for each encountered conversational flow.”

235. In every embodiment of the Challenged Patents, the specification describes the flow-entry database as having a separate entry for each known conversational

flow and creating a new entry for each new conversational flow. For example, the '099 Patent states:

The parser record is passed onto lookup process 314 which looks in an internal data store of records of known flows that the system has already encountered, and decides (in 316) whether or not this particular packet belongs to *a known flow as indicated by the presence of a flow-entry matching this flow in a database of known flows 324. A record in database 324 is associated with each encountered flow....*

The flow-entry database 324 stores flow-entries that include the unique flow-signature, state information, and extracted information from the packet for updating flows, and one or more statistical [sic] about the flow. *Each entry completely describes a flow.*²⁸⁰

The cache subsystem 1115 is an associated cache Whenever there is a cache miss, the contents of the cache memory pointed to by the bottom CAM are replaced by the flow-entry from the flow-entry database 324²⁸¹

... If the flow needs to be inserted or deleted from the database of flows, control is then passed on to the flow insertion/deletion engine 1110 for that flow signature and packet entry.²⁸²

236. Similarly, the '903 Provisional, which each Challenged Patent incorporates-

²⁸⁰ '099 Patent, 13:54-14:18.

²⁸¹ '099 Patent, 24:9-18.

²⁸² '099 Patent, 26:37-40.

by-reference, describes the flow-entry database as “*For every flow that has already been encountered, as indicated by a flow entry being present in the flow database*, there are various criteria for recognizing a packet’s particular state level.”²⁸³

237. Further, the prosecution history confirms that Patentee viewed its purported invention as requiring a database having a separate entry for each encountered conversational flow. For example, in response to a rejection of the ’646 claims by the Patent Office, Patentee summarized the invention by explaining the “flow-entry database” requirements:

The present invention includes a process that recognizes *a conversational flow*. [Examiner’s relied-upon art] Gobuyan does not recognize a conversational flow, but instead looks up only each packet’s destination address and source address.... [E]ven for the same two stations, the *present invention* identifies different conversational flows between two stations and *maintains a different entry for each different conversational flow in a database*....

It is important to be able to distinguish between packets that are exchanged between a source and a destination, and a *con[v]ersational flow* as used in the present invention.... *Different conversational flows may occur between the same two addresses. Each of these would have*

²⁸³ ’903 Provisional, 26:26-28; ’903, 43:1-8.

a separate entry in the flow database. Gobuyan, however, being concerned with routing and bridging, cannot distinguish between different conversational flows.²⁸⁴

It is my understanding that it is proper to interpret claims consistently across patents derived from same parent application and sharing common terms. Based on the Patentee's statements made during the prosecution of the '646 Patent, a POSITA would have understood that Patentee represented that each conversational flow as having a separate entry in the flow-entry database.

238. In current district-court proceedings, I understand that Patentee has proposed "flow-entry database" to mean "a database configured to store entries, where each entry describes a flow."²⁸⁵ This proposed construction appears broad as it fails to address what flow-entries the "database" must store to practice the claims.

239. My analysis herein is the same if Patentee's construction is used or if the "flow-entry database" terms require a "database having a separate entry for each encountered conversational flow."

F. "Parser Record"

240. The term "parser record" appears in the following Challenged Claims: '646 claims 7 and 16, and '789 claims 1, 17, 19, 42, and 44.

²⁸⁴ Ex. 1020 ('646 History), 9-10 (some emphasis in original) (i

²⁸⁵ Ex. 1067 (Claim Construction Memo and Order in *NetScout* district court case), 7-10.

241. In a previous district court proceeding, I understand that the court construed “parser record” to have its plain meaning.²⁸⁶

242. For purposes of *inter partes* review, I have applied the district court’s construction to the term “parser record.”

G. “Child Protocol”

243. The term “child protocol” appears in the challenged claims 10 and 17 of the ’725 Patent.

244. In a previous district court proceeding, I understand that the court construed “child protocol” to mean “a protocol that is encapsulated within another protocol.”²⁸⁷ And in another district court proceeding, I understand that Patentee agreed to this same construction.²⁸⁸

245. My analysis herein is the same if the district court’s construction is used or if the ordinary and customary meaning is used.

H. “Slicer”

246. The term “slicer” appears in the challenged claim 19 of the ’789 Patent.

247. It is my understand that Palo Alto Networks and Patentee have agreed

²⁸⁶ Ex. 1067 (Claim Construction Memo and Order in *NetScout* district court case), 10-16.

²⁸⁷ Ex. 1067 (Claim Construction Memo and Order in *NetScout* district court case), 5

²⁸⁸ Ex. 1077 (Joint Claim Construction and Prehearing Statement in *Ericsson* district court case), 1.

“slicer” means “component or process that performs extraction operations on a packet.”²⁸⁹

248. My analysis herein is the same if the agreed-upon construction is used or if the ordinary and customary meaning is used.

I. Means- and Step-Plus-Function Terms

249. In prior district court proceedings, I understand that the accused infringers proposed that the following terms are means- and step-plus-function terms:

Claim	Means- & Step-Plus-Function Terms	Corresponding Structure
'099 claim 1	“a parser subsystem ... the parser subsystem configured to examine the packet accepted by the buffer, extract selected portions of the accepted packet, and form a function of the selected portions sufficient to identify that the accepted packet is part of a conversational flow-sequence”	Pattern recognition engine 1006, extraction engine (slicer) 1007, and parser output buffer memory 1010 of the hardware parser subsystem of Fig. 10, as described at 21:60-67 and 22:14-63 of '099 Patent
'099 claim 1	“a lookup engine ... configured to determine using at least some of the selected portions of the accepted packet if there is an entry in the flow-entry database for the conversational flow sequence of the accepted packet”	Lookup/update engine (LUE) 1107 of Fig. 11, as described at 23:29-62 of '099 Patent
'099 claim 1	“a protocol/state identification mechanism ... the protocol/state identification engine configured to determine the protocol and state of the conversational flow of the packet”	state processor instruction database 326 and hardware or processor running the algorithm, as described at 14:38-46 of '099 Patent

²⁸⁹ Ex. 1079 (Joint Claim Construction and Prehearing Statement in *Palo Alto Network* district court case).

Claim	Means- & Step-Plus-Function Terms	Corresponding Structure
		<p>In addition, I understand that Palo Alto Networks has argued in district court that this claim term is an indefinite means-plus-function term because there is no adequate disclosure of a corresponding structure. As noted above, I have been asked to assume the claims are not indefinite, and therefore express no opinion here on whether this term is definite.</p>
'099 claim 1	<p>“a state processor ... the state processor, configured to carry out any state operations specified in the state patterns/operations memory for the protocol and state of the flow of the packet, the carrying out of the state operations furthering the process of identifying which application program is associated with the conversational flow-sequence of the packet, the state processor progressing through a series of states and state operations until there are no more state operations to perform for the accepted packet, in which case the state processor updates the flow- entry, or until a final state is reached that indicates that no more analysis of the flow is required, in which case the result of the analysis is announced”</p>	<p>State processor (SP) 1108 of Figs. 11 and 13, as described at 25:3-26:44 of '099 Patent</p>
'646 claim 1	<p>“a lookup engine ... configured to lookup whether a received packet belongs to a flow-entry in the flow- entry database, to looking up being the</p>	<p>Lookup/update engine (LUE) 1107 of Fig. 11, as described at 19:10-43 of '646 Patent</p>

Claim	Means- & Step-Plus-Function Terms	Corresponding Structure
	cache subsystem”	
’646 claim 1	“a state processor ... the state processor being to perform any state operations specified for the state of the flow starting from the last encountered state of the flow in the case that the packet is from an existing flow, and to perform any state operations required for the initial state of the new flow in the case that the packet is from an existing flow”	State processor (SP) 1108 of Figs. 11 and 13, as described at 20:49-22:20 of ’646 Patent
’646 claim 2	“a parser subsystem ... the parser subsystem configured to extract identifying information from a received packet”	Pattern recognition engine (PRE) 1006, extraction engine (slicer) 1007, and parser output buffer memory 1010 of the hardware parser subsystem of Fig. 10, as described at 17:43-51 and 17:64-18:43 of ’646 Patent
’646 claim 7	“a parser subsystem ... the parsing subsystem configured to extract selected portions of the accepted packet and to output a parser record containing the selected portions... wherein the operation of the parser subsystem depends on one or more of the protocols to which the packet conforms”	Pattern recognition engine (PRE) 1006, extraction engine (slicer) 1007, and parser output buffer memory 1010 of the hardware parser subsystem of Fig. 10, as described at 17:43-51 and 17:64-18:43 of ’646 Patent

Claim	Means- & Step-Plus-Function Terms	Corresponding Structure
'646 claim 7	“a lookup engine ... configured to lookup whether the particular packet whose parser record is output by the parser subsystem has a matching flow- entry, the looking up using at least some of the selected packet portions and determining if the packet is of an existing flow... the lookup engine configured such that if the packet is of an existing flow, the monitor classifies the packet as belonging to the found existing flow”	Lookup/update engine (LUE) 1107 of Fig. 11, as described at 19:10-43 of '646 Patent
'646 claim 7	“a flow insertion engine ... configured to create a flow-entry in the flow-entry database, the flow-entry including identifying information for future packets to be identified with the new flow-entry”	Flow insertion/deletion engine (FIDE) 1110 of Figs. 11-12, as described at 19:30-35 and 22:21-23:17 of '646 Patent
'646 claim 16	“performing one or more parsing/extraction operations on the packet to create a parser record comprising a function of selected portions of the packet”	Acts of processing packet components (steps 503-512), extracting packet components and building a packet signature (steps 603-610), and hashing signature buffer element based on hash elements in pattern node of element database (steps 703-708) of Figs. 5-7, as described at 3:48-57 and 14:40-15:64 of '646 Patent
'751 claim 17	“an analyzer subsystem ... configured to lookup for each received packet whether a received packet belongs to a flow-entry in the flow-entry database, to update the flow-entry of the existing flow including storing one or more statistical measures kept in the flow-	UFKB buffer, a lookup/update engine (LUE), a state processor (SP), a flow insertion and deletion engine (FIDE), a memory for storing the database of flows, a cache

Claim	Means- & Step-Plus-Function Terms	Corresponding Structure
	entry in the case that the packet is of an existing flow, and to store a new flow-entry for the new flow in the flow-entry database, including storing one or more statistical measures kept in the flow-entry if the packet is of a new flow”	coupled to the memory containing the flow database, and a host bus interface, as described in 2:30-43, 20:18-53 of ’751 Patent
’789 claim 1	“performing one or more parsing/extraction operations on the packet to create a parser record comprising a function of selected portions of the packet”	Acts of processing packet components (steps 503-512), extracting packet components and building a packet signature (steps 603-610), and hashing signature buffer element based on hash elements in pattern node of element database (steps 703-708) of Figs. 5-7, as described at 7:65-8:6 and 18:61-20:20 of ’789 Patent
’789 claim 19	“a parser subsystem ... the parsing subsystem configured to extract selected portions of the accepted packet and to output a parser record containing the selected portions ... wherein the operation of the parser subsystem depends on one or more of the protocols to which the packet conforms”	Pattern recognition engine (PRE) 1006, extraction engine (slicer) 1007, and parser output buffer memory 1010 of the hardware parser subsystem of Fig. 10, as described at 22:2-8 and 22:22-23:3 of ’789 Patent

Claim	Means- & Step-Plus-Function Terms	Corresponding Structure
'789 claim 19	“a lookup engine ... configured to lookup whether the particular packet whose parser record is output by the parser subsystem has a matching flow- entry, the looking up using at least some of the selected packet portions and determining if the packet is of an existing flow... the lookup engine configured such that if the packet is of an existing flow, the monitor classifies the packet as belonging to the found existing flow”	Lookup/update engine (LUE) 1107 of Fig. 11, as described at 23:38-24:3 of '789 Patent
'789 claim 19	“a flow insertion engine ... configured to create a flow-entry in the flow-entry database, the flow-entry including identifying information for future packets to be identified with the new flow-entry ... if the packet is of a new flow, the flow insertion engine stores a new flow-entry for the new flow in the flow-entry database, including identifying information for future packets to be identified with the new flow-entry”	Flow insertion/deletion engine (FIDE) 1110 of Figs. 11-12, as described at 23:58-63 and 26:53-27:50 of '789 Patent

250. In prior district court proceedings, I understand that Patentee has stated that no construction is necessary for each of the terms listed in the above table because none of the terms is a means- or step-plus-function term.²⁹⁰

251. For IPR purposes only, I have applied the Patentee’s proposed construction,

²⁹⁰ E.g., Ex. 1077 (Joint Claim Construction and Prehearing Statement in *Ericsson* district court case), 18-31; Ex. 1078 (Patentee’s Opening Claim Construction Brief in *Cisco* district court case), 20-22, 33-34.

i.e., that each of these terms is not a means- or step-plus-function terms and that the ordinary and customary meaning applies, since the prior art invalidates the Challenged Claims under either construction.

252. For all remaining claim terms, I used the ordinary and customary meaning to a POSITA at the time of the purported invention.

253. As explained below in Sections VII to XI, the prior art discloses or renders obvious every claim element even if any of these terms are interpreted more narrowly.

VII. THE CLAIMS OF THE '099 PATENT ARE UNPATENTABLE

254. For the '099 Patent, the Challenged Claims include independent claim 1 and claims 2 and 4, which each depends from claim 1, and claim 5, which depends from claim 4. As I detail below, it is my opinion:

- Riddle in view of Ferdinand renders obvious '099 claims 1 and 2;
- Riddle in view of Ferdinand and further in view of Baker renders obvious '099 claims 4 and 5;
- Riddle in view of Ferdinand and further in view of Yu renders obvious '099 claims 1 and 2;
- Riddle in view of Ferdinand and Baker and further in view of Yu renders obvious '099 claims 4 and 5;
- Riddle in view of Ferdinand and further in view of RFC1945 renders obvious '099 claims 1 and 2;
- Riddle in view of Ferdinand and Baker and further in view of RFC1945 renders obvious '099 claims 4 and 5.

A. For the '099 Patent, Riddle in View of Ferdinand Renders Obvious Claims 1 and 2.

255. It is my opinion that a POSITA would have recognized that each and every limitation of the '099 claims 1 and 2 is disclosed or rendered obvious by the prior art. Specifically, it is my opinion that '099 claims 1 and 2 are obvious in light of Riddle in view of Ferdinand.

1. Reasons to Modify Riddle in View of Ferdinand

256. Riddle and Ferdinand are in the same field of endeavor and contain overlapping disclosures with similar purposes. Riddle discloses a packet monitor that connects to a network for parsing and examining traffic packets.²⁹¹ And Riddle teaches its monitor stores flow-entry lists of packet identifying information and looks up flow-entries stored in the flow-entry lists.²⁹² Further, Riddle's monitor performs state operations to identify previously-encountered conversational flows or to store a new flows.²⁹³

257. Similarly, Ferdinand discloses a packet monitor. For example, Ferdinand teaches that network monitor 10 having a "processor which collects packets on the network and performs some degree of analysis ... to maintain statistical information for use in later analysis."²⁹⁴

²⁹¹ Riddle, 4:7-17, 12:27-41, Fig. 3, 4A-4B.

²⁹² Riddle, 12:37-59, Figs. 3, 4A-4B.

²⁹³ Riddle, 9:14-27, 12:44-53, claim 8, Figs. 4A-4B.

²⁹⁴ Ferdinand, 12:3-9.

258. At the time of the Challenged Patents' priority date, it was well-known and ubiquitous for networking devices to include database storage structures, buffers, caches, and distinct processing engines. Ferdinand illustrates this fact by describing a packet monitor having these features.²⁹⁵ For example, Ferdinand describes that its monitor includes a database for storing information about parsed packets.²⁹⁶ And Ferdinand teaches its monitor includes a packet buffer, database cache, and distinct processing engines.²⁹⁷

259. As I describe below regarding each relevant claim element, a POSITA would have been motivated and found it obvious to store Riddle's hierarchical classification tree and flow-entries in a database. Before the time of the invention, a POSITA would have been motivated to do so because storing Riddle's trees and lists in a database would allow multiple network operators to access simultaneously the classification information. As illustrated by Ferdinand, a POSITA would have appreciated the increased functionality of storing Riddle's data in a database—including searching, analyzing, and modifying the flow-entries.²⁹⁸ Such motivation would further Riddle's desired goal of determining whether the examined packet belongs to a service aggregate, such as an FTP session.²⁹⁹

²⁹⁵ E.g., Ferdinand, 19:8-12, 26:2-7, 28:14-17, Figs. 5, 7A-7C.

²⁹⁶ Ferdinand, 19:8-12, 22:18-23:23, 28:14-17, Figs. 5, 7A-7C.

²⁹⁷ Ferdinand, 20:22-22:12, 26:2-7, Fig. 5.

²⁹⁸ E.g., Ferdinand 41:32-42:3, 44:8-14, 47:3-48:11.

²⁹⁹ E.g., Riddle, 11:9-24, 13:36-62, Fig. 4B.

260. And as I describe below regarding each relevant claim element, a POSITA would have been motivated and found it obvious to have separate memory portions for Riddle's buffering, parsing/extraction operations, and state patterns/operations. Before the time of the invention, a POSITA would have been motivated to do so because Riddle's memory including a packet-buffer would provide a mechanism to store packets that may otherwise be dropped. A POSITA would have appreciated this provides the added benefit of improving performance by limiting packet drops. And as illustrated by Ferdinand, a POSITA would have been further motivated to include a cache coupled to Riddle's flow-entry database memory to reduce look-up times.³⁰⁰

261. Finally, to the extent Patentee argues that the Challenged Claims require distinct hardware components for the claimed parsing, lookups, protocol/state identification, and state processing/operations, a POSITA would have been motivated and found it obvious for Riddle's monitor to have distinct hardware components. Before the time of the invention, a POSITA would have been motivated to do so because using separate components allows for increased performance of Riddle's monitor. And the Challenged Patents acknowledge that a POSITA would have been appreciated the benefits and drawbacks to using separate hardware components versus software running on fast processors:

³⁰⁰ E.g., Ferdinand, 28:14-24, 54:18-22.

Note that while we are describing a particular hardware implementation of the invention embodiment of FIG. 3, it would be clear to one skilled in the art that the flow of FIG. 3 may alternatively be implemented in software running on one or more general-purpose processors, or only partly implemented in hardware... To one skilled in the art it would be clear that more and more of the system may be implemented in software as processors become faster.³⁰¹

2. Independent '099 Claim 1

262. It is my opinion that independent claim 1 of the '099 Patent is obvious in light of Riddle in view of Ferdinand.

- a. *'099 Claim 1's Preamble: "A packet monitor for examining packets passing through a connection point on a computer network in real-time, the packets provided to the packet monitor via a packet acquisition device connected to the connection point, the packet monitor comprising"*

263. Riddle discloses all elements of this preamble. As discussed above in Section IV.A.3, Riddle teaches automatically examining and classifying packet flows passing through a network using traffic classifier 304.³⁰² Riddle's Figure 3 illustrates the classifier 304, which parses packets and examines traffic flows for classification (traffic flows a, b, and c), as annotated below.

³⁰¹ '099 Patent, 21:25-38.

³⁰² Riddle, Abstract, 4:6-17, 12:27-41, 14:22-40, Fig. 3.

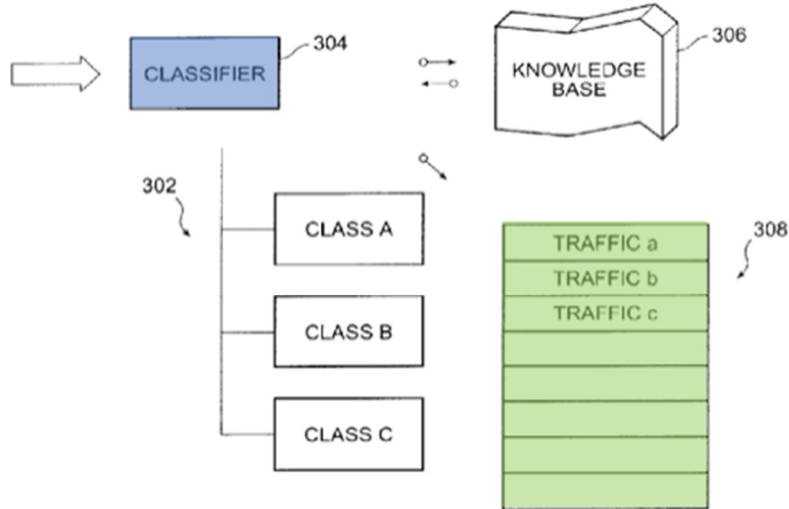
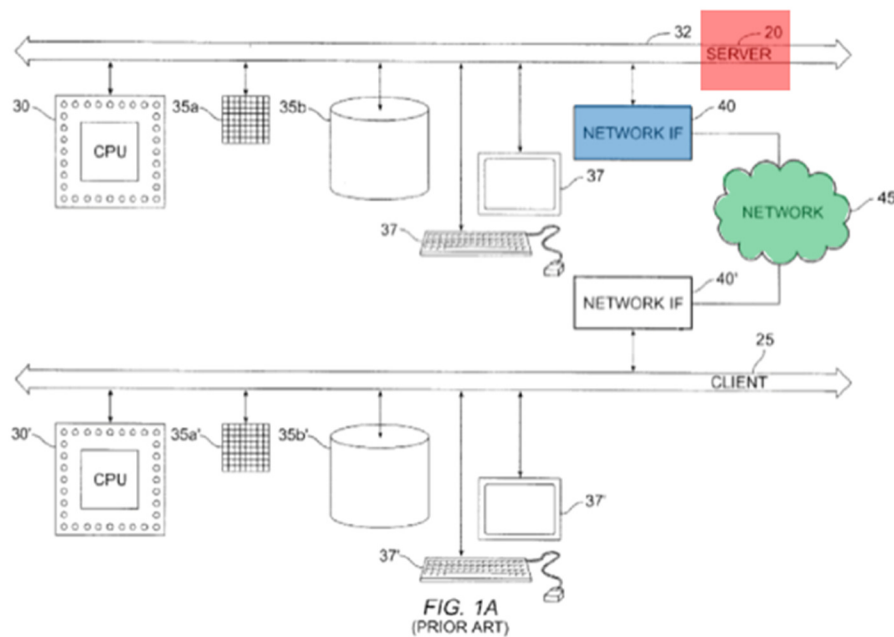


FIG. 3

264. Riddle details that its traffic classification system may be implemented in a classifier in a network-connected computer system, such as, in server 20, which acts as a packet monitor, and network interface 40, which acts as a packet acquisition device, shown below in annotated Figures 1A and 1B.



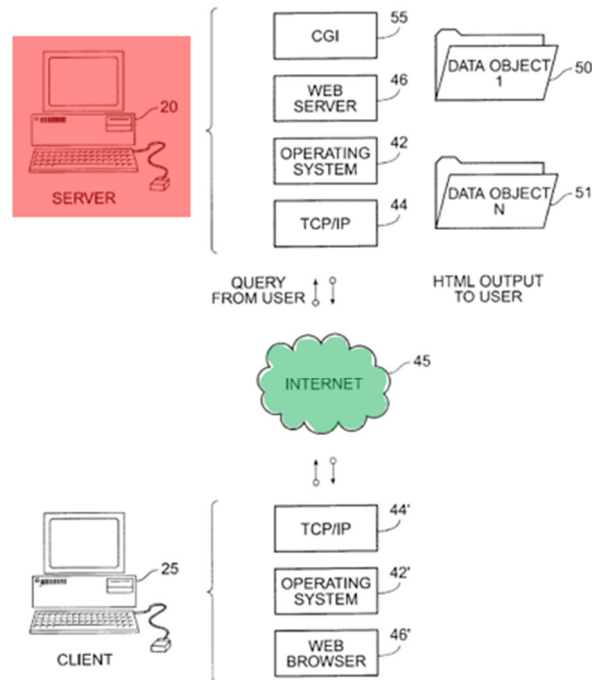


FIG. 1B
(PRIOR ART)

265. Riddle describes that its packet monitor examines packets as the packets pass through connection points of network connection 45 (shown above in green) via a system gateway.³⁰³ Riddle specifies that its packet examining method may cover any type of computer networking environment and is not limited to a client-server environment.³⁰⁴ And as the '099 Patent teaches, a connection point is simply where the packet monitor is connected to the network.³⁰⁵

266. Riddle details that its packet acquisition device, such as interface 40, is connected to a network connection point and provides packets to the packet monitor,

³⁰³ Riddle, 5:53-67, 6:9-15, Figs. 1A-1B.

³⁰⁴ Riddle, 5:53-67.

³⁰⁵ '099 Patent, 8:55-9:11, Fig. 1 showing connection points 121, 123, 125.

such a traffic classifier 304 in server 20.³⁰⁶ And for networks connecting multiple clients and servers of Figure 1A, Riddle teaches examining packets via a network routing means, which may include router 75, shown below in annotated Figure 1C, or traffic classifier 304.³⁰⁷ Riddle details that router 75 acts as a “system gateway ... which may also be a gateway having a firewall or a network bridge.”³⁰⁸ Based on these disclosures, a POSITA would have understood that Riddle’s system gateway is a connection point on the computer network.

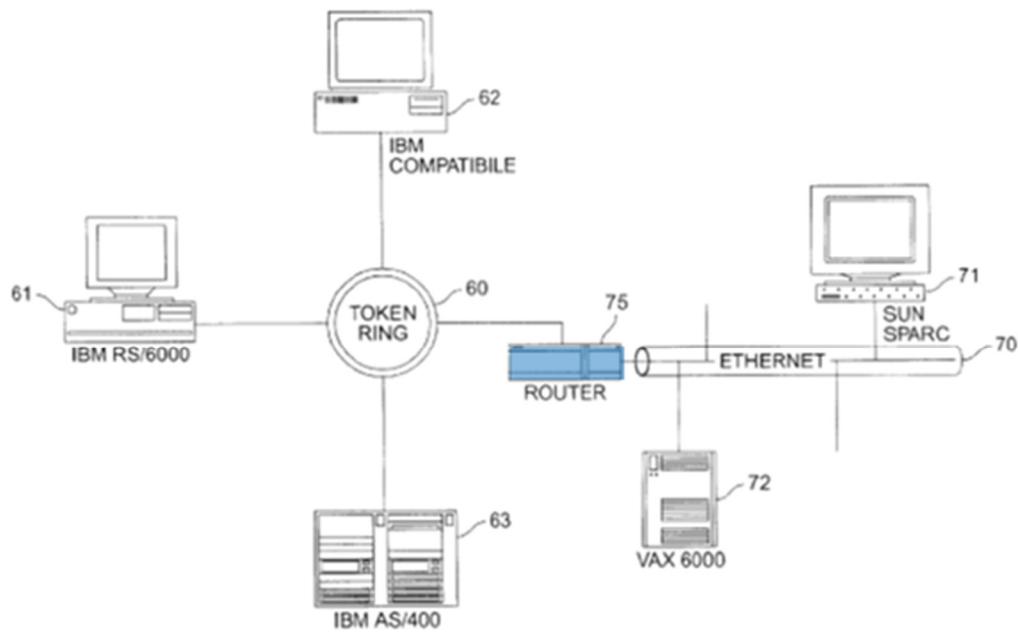


FIG. 1C
(PRIOR ART)

267. Further, Riddle discloses examining packets in real-time to manage network bandwidth based on flow classification. Riddle emphasizes the need to allocate

³⁰⁶ Riddle, 6:5-15, Figs. 1A-1B.

³⁰⁷ Riddle, 7:10-34, claim 8, Figs. 1C, 3.

³⁰⁸ Riddle, 7:21-24.

bandwidth to flows based on information ascertainable from multiple OSI layers of each flow.³⁰⁹ As Riddle explains, bandwidth management improves data transmission efficiency and avoids unwanted data transfer stoppage:

Certain pathological loading conditions can result in instability, overloading and data transfer stoppage. Therefore, it is desirable to provide some mechanism to optimize efficiency of data transfer while minimizing the risk of data loss. Early indication of the rate of data flow which can or must be supported is imperative. In fact, data flow rate capacity information is a key factor for use in resource allocation decisions. For example, if a particular path is inadequate to accommodate a high rate of data flow, an alternative route can be sought out.³¹⁰

To improve bandwidth management, Riddle describes “policies to assign available bandwidth from a single logical link to network flows.”³¹¹ And Riddle teaches examining and classifying traffic flows to allocate bandwidth consistent with those policies.³¹² From these disclosures, a POSITA would have understood that Riddle teaches traffic-examination and classification in real-time to assign available bandwidth.

³⁰⁹ Riddle, 1:54-61.

³¹⁰ Riddle, 2:4-13, 1:54-61.

³¹¹ Riddle, 2:66-67.

³¹² Riddle, Abstract, 4:6-23, 10:19-51.

268. Indeed, Riddle seeks to provide “a method for analyzing real traffic in a customer’s network and *automatically* producing a list of the ‘found traffic’”³¹³ Riddle specifies that its monitor can determine a packet’s traffic class, such as Real Time Protocol used for point-to-point telephony.³¹⁴ Based on these disclosures, a POSITA would have understood that Riddle teaches examining packets in real-time to optimally assign bandwidth based on flow rates.

269. As discussed above in Section IV.A.1, Riddle incorporates-by-reference the teachings of U.S. Patent 6,046,980 (“Packer”) as though fully set forth in Riddle. Like Riddle, Packer describes a traffic classification system and specifies that traffic flows are examined “continuously and automatically.”³¹⁵ This teaching further shows that Riddle discloses examining packets in real-time.

b. '099 Claim Element 1.1: “(a) a packet-buffer memory configured to accept a packet from the packet acquisition device”

270. Riddle discloses this claim element alone or renders it obvious in view of Ferdinand. As I detail below, Riddle teaches buffering packets in a router’s queue (including storage subsystem 35 of interface 40), which is a packet-buffer memory.

271. Specifically, Riddle describes its packet monitor as being implemented in a

³¹³ Riddle, 3:67-4:2, Abstract, 3:32-39, 4:6-9, 10:57-59.

³¹⁴ Riddle, 12:3-12.

³¹⁵ Ex. 1031 (U.S. Patent No. 6,046,980 (“Packer”)), 4:12-16.

router.³¹⁶ Riddle also discloses its router includes queues of packets.³¹⁷ A POSITA would have understood that Riddle's queue functions as a buffer within the router because Riddle's buffering of packets in its router's queue is a packet-buffer memory.

272. Regarding the structural components of the packet monitor, Riddle discloses storing data in storage subsystem 35:

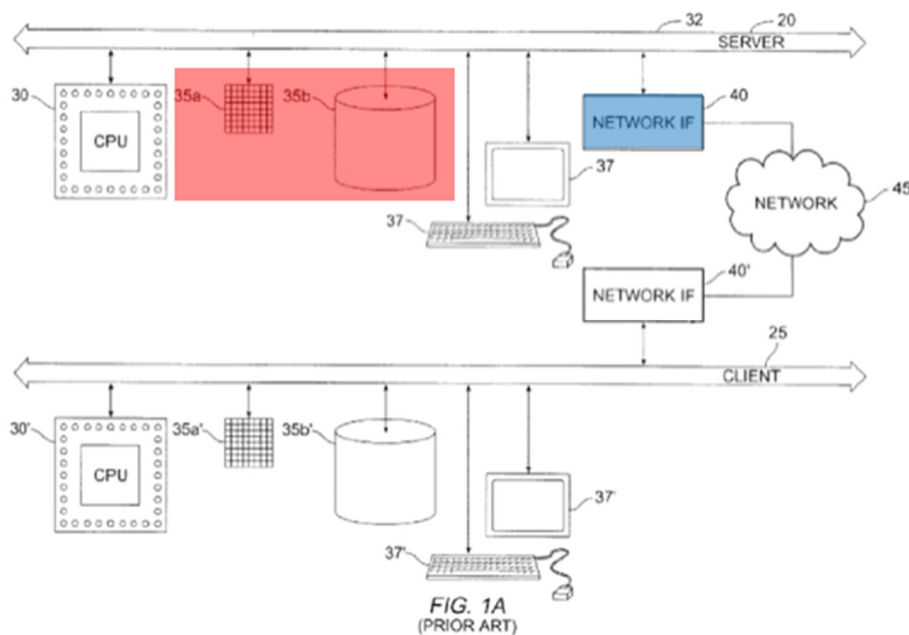
The hardware configurations are in general standard and will be described only briefly. In accordance with known practice, server 20 includes one or more processors 30 which communicate with a number of peripheral devices via a bus subsystem 32. These peripheral devices typically include *a storage subsystem 35, comprised of a memory subsystem 35a and a file storage subsystem 35b holding computer programs (e.g., code or instructions) and data*, a set of user interface input and output devices 37, and an interface to outside networks, which may employ Ethernet, Token Ring, ATM, IEEE 802.3, ITU X.25, Serial Link Internet Protocol (SLIP) or the public switched telephone network. ... Client 25 has the same general configuration, although typically with less storage and processing capability. Thus, while the client computer could be a terminal or a low-end personal computer, the server computer is generally a high-end workstation or mainframe, such as a SUN SPARC server. Corresponding elements and subsystems in the client computer are shown with corresponding, but

³¹⁶ Riddle, 7:21-24, claim 8.

³¹⁷ Riddle, 2:51-54.

primed, reference numerals.³¹⁸

Riddle's storage subsystem 35 includes memory subsystem 35a and file storage subsystem 35b for storing programs and data as shown below in annotated Figure 1A. As discussed above, Riddle's network interface 40 is a packet acquisition device. Riddle's Figure 1A shows network interface 40 being coupled to storage subsystem 35.



273. Moreover, Riddle discloses that its packet monitor may be implemented in a router.³¹⁹ And Riddle's router includes queues of packets.³²⁰ Based on these disclosures, a POSITA would have understood that Riddle's buffering of packets in its router's queue is a packet-buffer memory.

³¹⁸ Riddle, 6:1-23.

³¹⁹ Riddle, 16:54-60.

³²⁰ Riddle, 2:51-54.

274. As discussed above in Section IV.A.1, Riddle incorporates-by-reference the teachings of Packer as though fully set forth in Riddle. Packer incorporates-by-reference a set of appendices with exemplary source code illustrating the functionalities of the traffic classification system.³²¹ Packer's Appendix B specifies that the traffic classification system receives TCP flows in both directions for temporary storage in packet-buffer memory:

```
TCB – transport control block – TCP State information for both directions
/*
* BCB – Buffer Control Block. Contains packet info, including parsed
* flow spec, as well as pointers to various layers of the actual
* packet buffer
*/322
```

This teaching further shows that Riddle discloses a packet-buffer memory configured to accept packets.

275. To the extent Riddle does not disclose a packet-buffer memory, a POSITA would have been motivated and found it obvious to include a packet-buffer memory in Riddle's routing device based upon a POSITA's own knowledge of network devices or the disclosures in Ferdinand. Before the priority date of the

³²¹ Ex. 1031 (Packer), 1:54-2:3.

³²² Ex. 1027 (U.S. Patent Application No. 08/977,642 (“Packer Application”), Appendices), 71-72.

Challenged Patents, a POSITA would have known that packet-buffer memories, such as queues, were found in every routing device.³²³ As the '099 Patent and Ferdinand acknowledge, a POSITA would have understood that a packet buffer temporarily stores incoming packets until the device is ready to process the packets.³²⁴ In doing so, the packet buffer avoids packet loss because it provides a mechanism to store packets that may otherwise be dropped. Ferdinand discloses an exemplary packet-buffer memory, such as a frame buffer, which is used to accept packets in network monitors:

The available memory is divided into four blocks during system initialization. *One block includes receive frame buffers. They are used for receiving LAN traffic and for receiving secondary link traffic.* These are organized as linked lists of fixed sized buffers.³²⁵

276. Based on Ferdinand's teachings, a POSITA would have been motivated to include a packet-buffer memory in Riddle's monitor to temporarily store received packets and to improve performance by limiting packet drops. Including a packet-buffer memory in a packet acquisition device in accordance with the teachings of Riddle and Ferdinand amounts to nothing more than combining known prior-art technologies used in their ordinary and predictable manner to queue packet traffic.

³²³ E.g., Riddle, 2:51-55.

³²⁴ '099 Patent, 22:60-23:3; Ferdinand, 49:2-12.

³²⁵ Ferdinand, 26:2-7, 41:17-31, 49:11-12.

- c. *'099 Claim Element 1.2: “(b) a parsing/extraction operations memory configured to store a database of parsing/extraction operations that includes information describing how to determine at least one of the protocols used in a packet from data in the packet”*

277. Riddle discloses this claim element. Regarding the claimed “parsing/extraction operation memory,” Riddle discloses that its packet monitor includes storage subsystem 35 for storing “computer programs (e.g., code or instructions) and data” as discussed above regarding '099 claim element 1.1.³²⁶ As I detail below, these programs and data include a database of parsing/extraction operations to determine the protocols used in a packet from the data in the packet.

- (1) Riddle teaches the claimed memory being “configured to store a database of parsing/extraction operations”

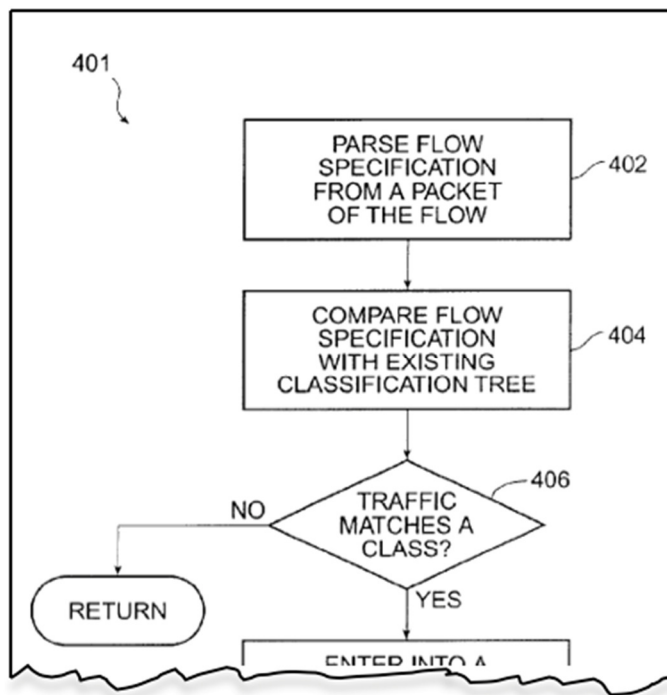
278. Riddle teaches that traffic flow classification includes parsing and extracting information from flow packets. For example, Riddle illustrates the steps for automatically classifying traffic in a flowchart beginning with Figure 4A, as annotated below. “In a step 402, a flow specification is parsed from the flow being classified.”³²⁷ Following the parsing, steps 404 and 406 compare the parsed specification information and determine whether the examined flow matches a preexisting traffic class.³²⁸ And at step 408, an entry is made into a saved list with the extracted

³²⁶ Riddle, 6:1-23, 6:43-50, Figs. 1A-1B.

³²⁷ Riddle, 12:42-44, Fig. 4A.

³²⁸ Riddle, 12:44-50, Fig. 4A.

identifying information such as “protocol type (SAP), IP protocol number, server port, traffic type if known, MIME type, a time of occurrence of the traffic.”³²⁹ Thus, Riddle discloses parsing/extraction operations to determine the protocols used in a packet.



279. Moreover, Riddle describes parsing/extraction operations as:

A method for automatically classifying traffic in a packet communications network ... comprising the steps of: parsing a packet into a first flow specification, wherein said first flow specification contains at least one instance of any one of the following: a protocol family designation, a direction of packet flow designation, a *protocol type designation*, a pair of hosts, a pair of ports, in HTTP protocol packets, a pointer to a MIME type.³³⁰

³²⁹ Riddle, 12:50-53, Fig. 4A.

³³⁰ Riddle, claims 1, 8, 11.

Riddle also describes each “particular host” as having an IP Internet Address, where a pair of hosts (*e.g.*, a client and server) communicating over different ports.³³¹ Thus, a POSITA would have understood that Riddle’s “a pair of hosts” refers to the network-layer source and destination addresses (*e.g.*, IP addresses) and that “a pair of ports” refers to the transport-layer source and destination port numbers. This further details how Riddle’s parsing/extraction operations determine the protocols used in a packet.

280. As discussed above in Section IV.A.1, Riddle incorporates-by-reference the teachings of U.S. Patent Application No. 09/198,051 (“the ’051 Application”) as though fully set forth in Riddle. The ’051 Application describes a traffic classification system and specifies parsing/extraction operations to examine packets:

A method for automatically determining a policy for allocating bandwidth resources in a packet telecommunication environment, wherein traffic flow for a plurality of service types are grouped into at least one class according to a hierarchical classification paradigm, said method comprising the steps: *extracting a flow specification from a traffic flow* having a traffic direction, a host side from the traffic class and at least one of any of the following: A global service, a port number[,], *an IP Address/Domain name; a MAC Address, and an IP precedence*³³²

³³¹ Riddle, 7:62-63, 11:13-15.

³³² Ex. 1028 (U.S. Patent Application No. 09/198,051 (“’051 Application”)), 23.

281. Riddle details that traffic classifier 304, which performs the parsing and extraction operations, communicates with knowledge base 306 (shown below in Figure 3). And Riddle states that knowledge base 306 stores the operations for determining a packet's traffic class.³³³

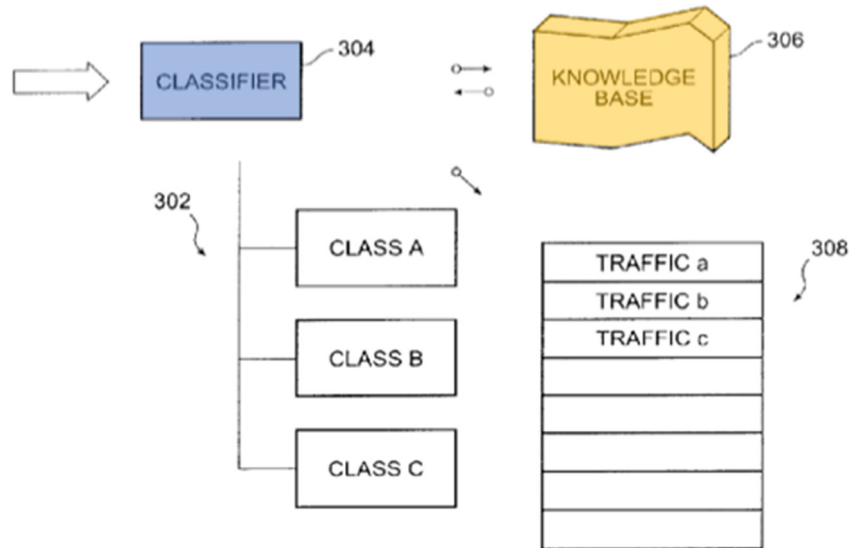


FIG. 3

282. Riddle specifies storing knowledge base 306 in a database and that classifier 304 uses knowledge base 306 to determine a packet's classification information:

FIG. 3 depicts components of a system for automatically classifying traffic according to the invention. A traffic tree 302 in which new traffic will be classified under a particular member class node. A traffic classifier 304 detects services for incoming traffic. Alternatively, the classifier may start with a service and determine the hosts using it. *A knowledge base 306 contains heuristics for determining traffic classes. The knowledge base*

³³³ Riddle, 12:30-36, Fig. 3.

may be embodied in a file or a relational database. In a particular embodiment, the knowledge is contained within a data structure resident in memory. A plurality of saved lists 308 stores classified traffic pending incorporation into traffic tree 302. In select embodiments, entries for each instance of traffic may be kept. In alternate embodiments, a copy of an entry and a count of duplicate copies for the entry is maintained.³³⁴

- (2) Riddle further teaches the claimed “includ[ing] information describing how to determine at least one of the protocols used in a packet from data in the packet”

283. Riddle details that its stored knowledge base 306 includes information describing how to determine the protocols used by a packet in a flow from data contained in the packet.³³⁵ Riddle seeks to address the purported lack of prior art “methods for automatically classifying packet traffic based upon information gathered from ... multiple layers in a multi-layer protocol network.”³³⁶ In doing so, Riddle teaches that its packet monitor examines packets to manage bandwidth “based on information ascertainable from multiple layers of OSI network model.”³³⁷

284. As shown in Riddle’s Figure 1D provided below, it was well known in the art before the priority date of the Challenged Patents that the OSI network model

³³⁴ Riddle, 12:26-41, Fig. 3.

³³⁵ Riddle, 12:30-36 (referring to “the heuristics for determining traffic classes”).

³³⁶ Riddle, 3:36-39.

³³⁷ Riddle, 1:54-57.

diagrams the relationship between the layers of the TCP/IP protocol suite. These layers include the application layer 88, the transport layer 86, the network layer 84, the data link layer 82, and the physical layer 80.³³⁸

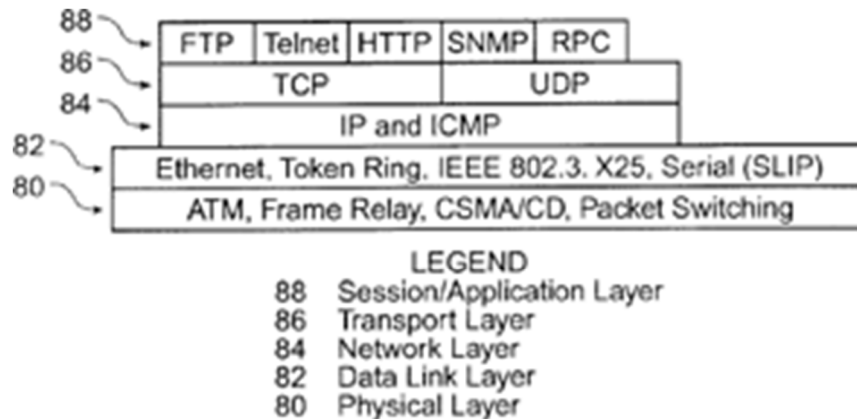


FIG. 1D
(PRIOR ART)

285. Riddle details how its parsing/extraction operations determine the protocols used in the packet from the packet’s data. Riddle states that its operations include defining traffic class characteristics based on packet protocol information:

“The method comprises applying individual instances of traffic classification paradigms to packet network flows based on selectable information obtained from a plurality of layers of a multi-layered communication protocol in order to define a characteristic class, then mapping the flow to the defined traffic class.”³³⁹ And Rid-

³³⁸ Riddle, 7:35-8:46, Fig. 1D.

³³⁹ Riddle, 4:10-15.

dle specifies its operations are “sufficiently robust to classify a complete enumeration of the possible traffic.”³⁴⁰ Further, Riddle explains that its operations “automatically classify a plurality of heterogeneous packets in a packet telecommunications system for management of network bandwidth in systems such as a private area network, a wide area network or an internetwork.”³⁴¹

286. By parsing packets, Riddle seeks to classify traffic flows by “traffic class.” Riddle defines a traffic class as “[a]ll traffic between a client and a server endpoints.... Traffic classes have properties or class attributes such as, directionality, which is the property of traffic to be flowing inbound or outbound.”³⁴² And Riddle defines a “flow” as “a single instance of a traffic class. For example, all packets in a TCP connection belong to the same flow. As do all packets in a UDP session.”³⁴³

287. In determining a flow’s traffic class, Riddle also determines the flow’s one or more protocols:

Traffic classes may be defined at any level of the IP protocol as well as for other non-IP protocols. For example, at the IP level, traffic may be defined as only those flows between a specified [sic] set of inside and outside IP addresses or domain names. An example of such a low level traffic class definition would be all traffic between my network and other corporate offices throughout the Internet. *At the application level, traffic*

³⁴⁰ Riddle, 4:15-17.

³⁴¹ Riddle, 4:55-60.

³⁴² Riddle, 5:42-45.

³⁴³ Riddle, 5:17-20.

classes may be defined for specific URIs within a web server. Traffic classes may be defined having “Web aware” class attributes. For example, a traffic class could be created such as all URIs matching “*.html” for all servers, or all URI patterns matching “*.gif” for server X, or for access to server Y with URI pattern “/sales/*” from client Z, wherein ‘*’ is a wildcard character, i.e., a character which matches all other character combinations. Traffic class attributes left unspecified will simply match any value for that attribute. For example, a traffic class that accesses data objects within a certain directory path of a web server is specified by a URI pattern of the directory path to be managed, e.g. “/sales/*”

The present invention provides a method for classifying traffic according to a definable set of classification attributes selectable by the manager, including selecting a subset of traffic of interest to be classified. The invention provides the ability to classify and search traffic based upon multiple orthogonal classification attributes.

Traffic class membership may be hierarchical. Thus, a flow may be classified by a series of steps through a traffic class tree, with the last step (i.e., at the leaves on the classification tree) mapping the flow to a policy. The policy is a rule of assignment for flows. *Web traffic may also be classified by HTTP header types such as Content-Type (MIME type) or User-Agent.*³⁴⁴

288. Riddle’s Table 2 provides examples of information from which traffic classes may be built.³⁴⁵ This table illustrates that Riddle’s monitor may use packet data

³⁴⁴ Riddle, 8:47-9:27.

³⁴⁵ Riddle, 9:64-65.

to determine the protocols used in the packet. Based on Table 2, an exemplary traffic class may be defined for an FTP application using a client IP address inside a network and a server IP address outside a network.

TABLE 2

<u>Components of a Traffic Class Specifier</u>		
<u>Inside (Client or Server)</u>	<u>Global</u>	<u>Outside (Server or Client)</u>
IP Address/Domain Name	TCP or UDP Service	IP Address/Domain Name
Port Number	e.g., WWW,	Port Number
MAC Address	FTP, RealAudio, etc. URI pattern for Web Service, MIME type for Web Service IPX Service SNA Service LAT Service IP precedence	MAC Address

289. Similarly, the '051 Application, which Riddle incorporates-by-reference as though fully set forth in Riddle, includes a Table 2 providing examples of information from which traffic classes may be built.³⁴⁶ The '051 Application's Table 2 shows that exemplary global components for web service applications may include "*.html," "*.gif," and "/sales/*":

³⁴⁶ Ex. 1028 ('051 Application), 17.

Components of a Traffic Class Specifier

<u>Inside (Client or Server)</u>	<u>Global</u>	<u>Outside (Server or Client)</u>
IP Address/Domain Name		IP Address/Domain Name
Port Number	TCP or UDP Service,	Port Number
MAC Address	e.g. WWW, FTP, RealAudio, etc. URI pattern for Web Service, e.g. "*.html", "*.gif", "/sales/*", etc. IPX Service SNA Service LAT Service IP precedence	MAC Address

Table 2

290. Riddle further teaches classifying traffic based on “protocol layer independent categories,” such as application-layer information:

Network traffic is automatically classified under existing classes, beginning with the broadest classes, an inbound traffic class and an outbound traffic class, *in protocol layer independent categories. For example, a particular instance of traffic may be classified according to its transport layer characteristics, e.g., Internet Protocol port number, as well as its application layer information, e.g., SMTP.* Characteristics such as MIME types may also be automatically identified. Standard protocols, such as, IPX, SNA, and services, such as, SMTP and FTP are recognized for automatic classification. *Classification is performed to the most specific level determinable. For example, in select embodiments, non-IP traffic, such as SNA, may be classified only by protocol, whereas Internet Protocol traffic may be classified to the /etc/services level.* Classification beyond a terminal classification level is detected and prevented. For example, in a

select embodiment, a class matching “ipx” or “nntp” will not be further automatically classified.³⁴⁷

291. Riddle further details how the packets are examined to determine the protocols used in the packet:

In a preferable embodiment, classification can extend to examination of the data contained in a flow’s packets. Certain traffic may be distinguished by a signature even if it originates with a server run on a non-standard port, for example, an HTTP conversation on port 8080 would not be otherwise determinable as HTTP from the port number. Further analysis of the data is conducted in order to determine classification in instances where: *1) FTP commands are used to define server ports, 2) HTTP protocol is used for non-web purposes*. The data is examined for indication of push traffic, such as pointcast, which uses HTTP as a transport mechanism. These uses may be isolated and classified into a separate class. Marimba and pointcast can be distinguished by looking into the data for a signature content header in the get request. Pointcast has URLs that begin with “/FIDO-1/.” *Other applications in which protocol can be inferred from data include Telnet traffic*. Both tn3270 and tn3270E (emulation) may be detected by looking into data and given a different class. Telnet traffic has option negotiations which may indicate an appropriate class.³⁴⁸

292. Riddle also provides additional detail on determining the packet’s protocols.

For example, Riddle discusses detecting Real Time Protocol (RTP) and Real Time

³⁴⁷ Riddle, 11:57-12:9.

³⁴⁸ Riddle, 11:47-67.

Streaming Protocol (RTSP):

A traffic class may be inferred from determining the identity of the creator of a resource used by the traffic class. For example, the identity of traffic using a certain connection can be determined by finding the identity of the creator of the connection. *This method is used to detect Real Time Protocol (RTP)* for point-to-point telephony, RTP for broadcast streaming, CCITT/ITU H320-telephony over ISDN, H323-internet telephony over the internet (bidirectional) and *RTSP real time streaming protocol* for movies (unidirectional).³⁴⁹

293. Based on these disclosures regarding databases and parsing/extraction operations, a POSITA would have understood that Riddle's storage subsystem 35 stores instructions (i.e., code) in the form of knowledge base 306 embodied in a relational database. And that these instructions include parsing/extraction operations with information from flow packets for classifying packets and flows into traffic classes by determining one or more protocols used in the flow.

d. '099 Claim Element 1.3: "(c) a parser subsystem coupled to the packet buffer and to the pattern/extraction³⁵⁰ operations memory"

294. Riddle discloses a parser subsystem. Specifically, Riddle teaches a processor programmed to perform parsing/extraction operations:

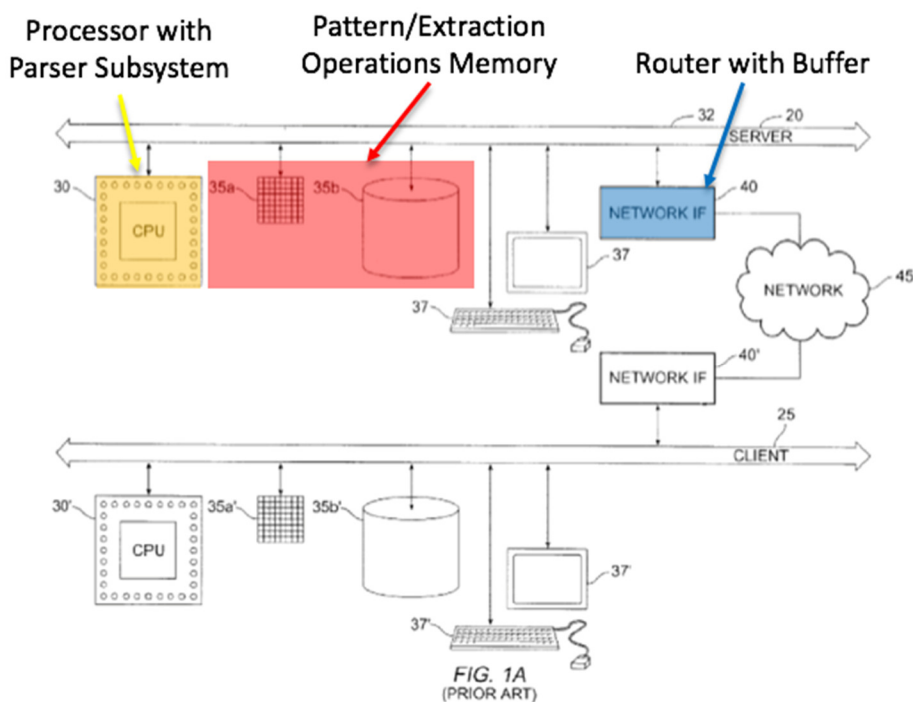
A system for automatically classifying traffic in a packet communications

³⁴⁹ Riddle, 12:1-12.

³⁵⁰ As best understood, it appears that the claimed "pattern/extraction" should read "parsing/extraction."

network ... comprising ... a network routing means; and, a processor means operative to: parse a packet into a first flow specification, wherein said first flow specification contains at least one instance of any one of the following: a protocol family designation, a direction of packet flow designation, a protocol type designation, a pair of hosts, a pair of ports, in HTTP protocol packets, a pointer to a MIME type.³⁵¹

295. As shown below in annotated Figure 1A, Riddle's system includes parser subsystem's processor 30 that communicates with parsing/extraction operations memory 35 and the router including the buffer.³⁵²



296. As discussed above with respect to '099 claim element 1.2, Riddle discloses a parsing/extraction operations memory. And as discussed above with respect to

³⁵¹ Riddle, claim 8.

³⁵² Riddle, 6:1-15, Fig. 1A.

'099 claim element 1.1, the Riddle discloses a packet buffer or the combination of Riddle and Ferdinand renders that claim element obvious. Thus, Riddle discloses its parser subsystem being coupled to the parsing/extraction operations memory and the packet buffer (or renders obvious the packet buffer in view of Ferdinand).

- e. *'099 Claim Element 1.4: “the parser subsystem configured to examine the packet accepted by the buffer, extract selected portions of the accepted packet, and form a function of the selected portions sufficient to identify that the accepted packet is part of a conversational flow-sequence”*

297. Riddle discloses this claim element. As I detail below, Riddle describes extracting portions of packet using its processor and memory subsystem, which is the claimed “parser subsystem.”

- (1) Riddle teaches the claimed parser subsystem is “configured to examine the packet accepted by the buffer, extract selected portions of the accepted packet”

298. Riddle teaches that its parser subsystem operates to apply “individual instances of traffic classification paradigms to packet network flows based on selectable information obtained from a plurality of layers of a multi-layered communication protocol in order to define a characteristic class, then mapping the flow to the defined traffic class.”³⁵³ As part of this examination and extraction process,

³⁵³ Riddle, Abstract, 4:10-15.

Riddle's system employs a relational database (knowledge base 306) to store heuristics (i.e., operations) for determining traffic classes based on parsing/extracting portions of a packet and matching those portions' attributes to a traffic class.³⁵⁴

299. As shown in Figure 4A's flowchart, Riddle's traffic classifier examines packets accepted by the monitor and extracts packet portions: "parse flow specification from a packet of the flow" (step 402).³⁵⁵ And Riddle specifies examining a number of packet flows and extracting flow specification portions from the flows:

A method for automatically classifying traffic in a packet communications network, said network having any number of flows, including zero, comprising the steps of: parsing a packet into a first flow specification, wherein said first flow specification contains at least one instance of any one of the following: a protocol family designation, a direction of packet flow designation, a protocol type designation, a pair of hosts, a pair of ports, in HTTP protocol packets, a pointer to a MIME type.³⁵⁶

300. Based on these teachings, a POSITA would have understood that the processor and memory subsystem portions in Riddle's router work together to examine packets and extract packet portions. And as described above regarding '099 claim elements 1.1 and 1.3, to the extent Riddle does not disclose the claimed "buffer,"

³⁵⁴ Riddle, 12:26-41, 9:28-42, 9:48-49.

³⁵⁵ Riddle, 12:42-53, Fig. 4A.

³⁵⁶ Riddle, claims 1, 11.

the combination of Riddle and Ferdinand renders obvious a buffer accepting packets.

- (2) Riddle teaches the claimed parser subsystem is “configured to ... form a function of the selected portions sufficient to identify that the accepted packet is part of a conversational flow-sequence”

301. Riddle discloses its parser forms a function (i.e., a flow specification) of packet portions to identify the packet is part of a conversational flow-sequence in at least two ways: (a) classifying based on service aggregates and (b) classifying based on PointCast.

(a) Riddle’s Service Aggregates Are The Claimed “Conversational Flow-Sequence”

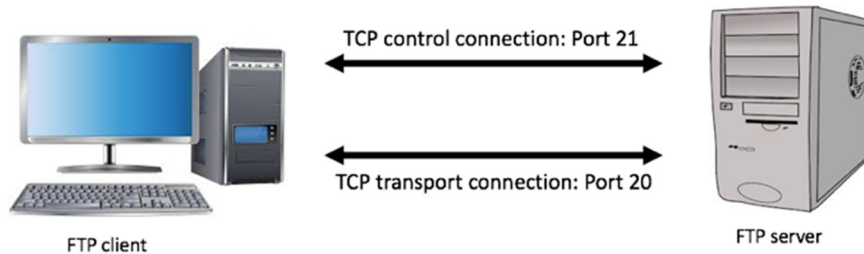
302. Riddle’s parser subsystem is configured to form a function of the selected packet portions “sufficient to identify that the accepted packet is part of a conversational flow-sequence,” as recited in this claim element. For example, as I detail below, Riddle discloses identifying whether a packet is part of “service aggregates.” Riddle’s service aggregate is a traffic class that links separate connection flows based on the application associated with the flows.

303. Riddle’s service aggregates meet the claimed “conversational flow-sequences” and “conversational flows” recited throughout the Challenged Claims, under either proposed construction for those terms. And Riddle teaches using these

service aggregates to form a function of selected packet portions sufficient to identify that the accepted packet is part of a conversational flow-sequence.

304. As discussed above in Section III.A.4, the FTP protocol became standardized in 1980.³⁵⁷ In the FTP protocol, a client establishes at least two connections to a server: (i) a first connection for communicating commands to control the file transfer and to send data about the state of the transfer, and (ii) a second connection for transferring the file.³⁵⁸

305. Below, I illustrate an example of two separate TCP connections used for FTP with one TCP connection on port 21 for control and another TCP connection on port 20 for data transport.



FTP Connections Between Server & Client

306. Taking into account this well-known background information, Riddle details classifying separate packet flows by a common “service aggregates” traffic class for applications using multiple flows between a client and a server:

A service aggregate is provided for certain applications that use more

³⁵⁷ Ex. 1037 (RFC765 – File Transfer Protocol).

³⁵⁸ Ex. 1037 (RFC765), 6-7.

than one connection in a particular conversation between a client and a server. For example, an FTP client in *conversation* with an FTP server employs a command channel and a transfer channel, which are distinct TCP sessions on two different ports. In cases where two or three TCP or UDP sessions exist *for each conversation* between one client and one server, it is useful to provide a common traffic class i.e., the service aggregate, containing *the separate conversations*. In practice, these types of *conversations* are between the same two hosts, but use different ports. According to the invention, a class is created with a plurality of traffic specifications, each matching various component *conversations*.³⁵⁹

As this passage illustrates, Riddle teaches that its monitor employs service aggregates for applications that may use more than one connection for a particular conversation.

307. Riddle specifies that its parser subsystem checks whether the parser packet portion is part of a service aggregate (i.e., a conversational flow):

FIG. 4B depicts a flowchart 403 of the processing steps for integrating traffic classes into a classification tree in an alternative embodiment.... In a step 420, an instance of saved traffic is retrieved from the saved traffic list 308. Next in a decisional step 422, the instance of saved traffic is examined to determine whether it is well-known (e.g. registered SAP, protocol type, assigned port number) and a name representing its type exists. *If this is so then processing continues with a test of whether the saved traffic belongs to a service aggregate in step 426.* Otherwise, in a step 423 the

³⁵⁹ Riddle, 11:10-23.

instance of saved traffic is examined to determine whether it appears to be a server connection port of an unregistered IP port (or a port that has not been configured). If this is not so then, processing continues with the next traffic class in the saved list in step 420. *In decisional step 426, the instance of saved traffic is examined to determine whether it belongs to a service aggregate. For example, an FTP session has one flow that is used to exchange commands and responses and a second flow that is used to transport data files. If the traffic does belong to a service aggregate, then in a step 428, a traffic class is created which will match all components of the service aggregate.* In a further step 425, a new traffic class is created to match the instance of saved traffic. The class may be flat or hierarchical.³⁶⁰

As shown below, Riddle illustrates this check for “service aggregate” conversational flows in Figure 4A and 4B’s flowchart. For example Riddle’s step 406 checks whether “traffic matches a class?” and step 426 checks whether “saved traffic belongs to a service aggregate?” And when describing command options for controller 304, Riddle details in Table 3 specifies the controller may “detect services in both directions.”³⁶¹

³⁶⁰ Riddle, 13:36-62, Fig. 4B.

³⁶¹ Riddle, 14:28-40.

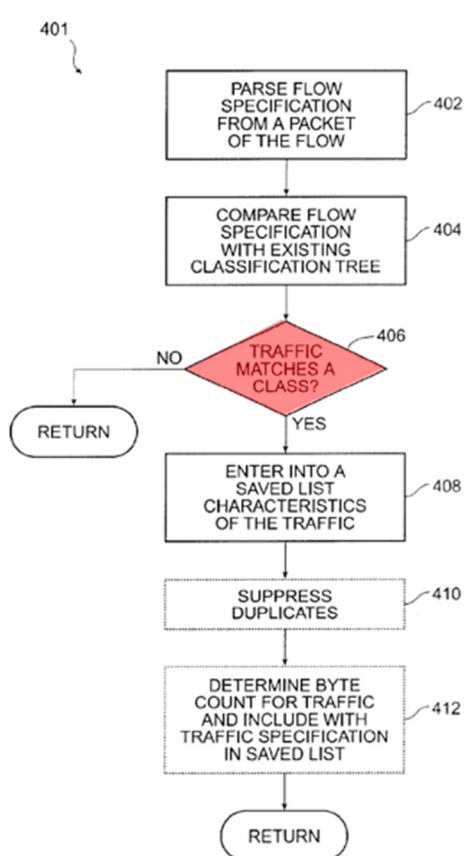


FIG. 4A

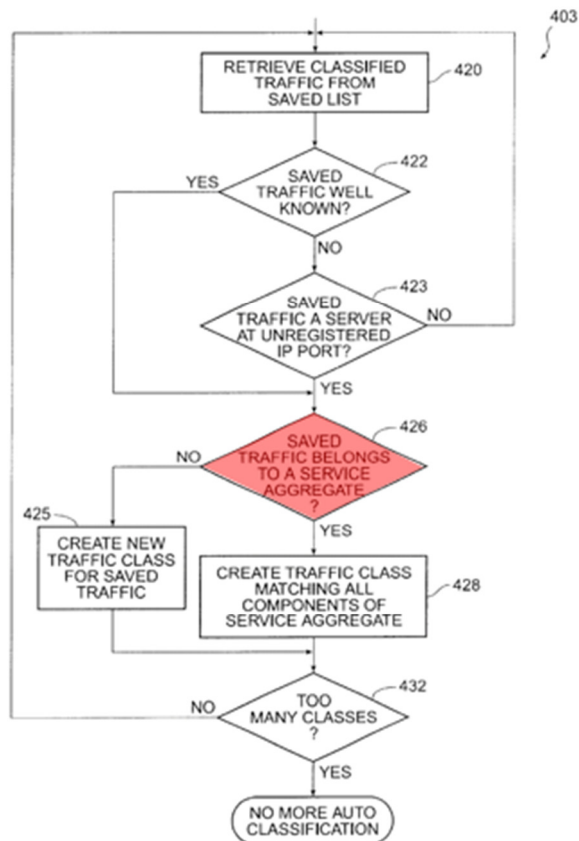


FIG. 4B

308. The '864 Provisional further illustrates how a POSITA would have understood Riddle's service aggregates relate multiple connection flows based on an FTP application's specific software program activity:

[T]he concept of “service aggregates” (service groups) [is] different traffic types that are associated together (ex. FTP has one stream that it uses to exchange commands and responses, and a second that the data files are actually sent over). Whenever we recognize the signature of one of these types of traffic, we create a traffic class (or class hierarchy) that can match all the components of the aggregate. This bundling is mainly a convenience to the user, makes it clearer what's going on, but also permits you to get group counts of all the parts that make up what

the user thinks as the service.³⁶²

309. Further, as I detail above in Section IV.A.6, Riddle describes creating traffic classes based on data relating to RTP and RTSP.³⁶³ A POSITA would have understood RTP and RTSP are analogous to FTP, and that those protocols use a separate control flow with one or more linked dataflows.³⁶⁴ As such, a POSITA would have appreciated this is another example of a common “service aggregate” traffic class for applications using multiple flows between a client and a server.

310. When addressing FTP applications, Riddle teaches “subclassifying” components of conversational flows:

Subclassification of traffic into a tree is performed by matching the hosts and then searching for particular services. Traffic *specifications are aggregate kinds of traffic for a traffic class, e.g., different components of FTP may reside under class FTP. Subclassification is performed by first locating a class that matches, and then performing finer grade matchings.* Processing commences with a decision on what traffic is to be subclassified. A marker is placed in the match_all default node so that when match processing reaches the marker, the autotclassification processing depicted in flowchart 403, determines that it has not found an existing class for the traffic being classified.³⁶⁵

³⁶² '864 Provisional, 69.

³⁶³ Riddle, 12:1-12.

³⁶⁴ Ex. 1045, 4-5 (RFC1889 illustrating well-known RTP information); Ex. 1046, 9-10 (RFC2326 illustrating well-known RTSP information).

³⁶⁵ Riddle, 11:24-36.

311. Riddle teaches displaying subclassifications to the user, which further illustrates these subclassifications are for disjointed packet flows:

In an optional step 413 (not show), after the processing of flowchart 401 completes or at periodic intervals or on demand, a list of traffic classes produced in steps 402 through 412 are displayed to a network manager. ... The display can be hierarchical, as depicted in lines (3) below:

```
FTP    (3)
  FTP-cmd
  FTP-data
to host1
  tcp
  FTP
    FTP-cmd
    FTP-data
  HTTP
    images
    java
    text
  port 9999 ....366
```

312. The above hierarchical classification tree illustrates “conversational flows” in three ways. First, under the FTP application, the tree shows linking disjointed FTP

³⁶⁶ Riddle, 12:64-13:23, 11:13-23. As shown in '864 Provisional (see page 22), the first line “FTP” of Riddle’s exemplary classification tree should be directly above the subclassifications “FTP-cmd” and “FTP-data.”

connections FTP-cmd and FTP-data. Second, under “tcp” below “to host1,” the tree shows linking disjointed FTP connections FTP-cmd and FTP-data under the FTP subclass. Third, under “tcp” below “to host1,” the tree shows linking images, java, and text under the HTTP subclass.

313. Riddle’s claims 1 and 2 further teach that service aggregates are conversational flows. These claims show that service aggregates are a sequence of packets exchanged in any direction as a result of activity, such as running an application, forming multiple connection flows linked by that activity. In pertinent part, claim 1 recites “said network having any number of flows” and “parsing a packet into a first flow specification.”³⁶⁷ Claim 2 depends from claim 1 and includes the further step of:

[F]or at least a second flow having a second flow specification, recognizing said second flow specification and said first flow specification to ***comprise together a service aggregate***; thereupon,

[A]ssociating said first flow specification and said second flow specification with a newly-created classification tree node, said ***newly-created classification tree type node having a first traffic specification corresponding to said first flow specification and a second traffic specification corresponding to said second flow specification***.³⁶⁸

314. As discussed above in Section III.J, I agree with the German court’s finding

³⁶⁷ Riddle, claim 1.

³⁶⁸ Riddle, claim 2.

that an FTP communication, which comprises two TCP connections, taught a “conversational flow.” And similarly, Riddle teaches identifying multiple flows as part of an FTP communication. As the German court recognized, the identification of flows as part of an FTP communication (as Riddle discusses) teaches identifying packets as being part of a conversational flow-sequence.

315. For these reasons, it is my opinion that Riddle teaches forming a function of selected packet portions sufficient to identify that the accepted packet is part of a conversational flow-sequence.

(b) Riddle’s PointCast Traffic Is The Claimed “Conversational Flow-Sequence”

316. As another example of a “conversational flow-sequence,” Riddle teaches that its parser subsystem parses packet to identify whether packet portions are PointCast traffic. And as discussed above, Patentee’s U.S. Provisional Patent Application No. 60/141,903 (“’903 Provisional”) discloses that identifying PointCast traffic creates a conversational flow.³⁶⁹

317. PointCast was known in the relevant art before the priority date of the Challenged Patents. It is my understanding that PointCast Inc. formed in 1992.³⁷⁰ From

³⁶⁹ Ex. 1016 (’903 Provisional), 7:16-25.

³⁷⁰ Ex. 1035 (PointCast Inc. 1998 SEC Filings), 89.

1992 to 1995, PointCast Inc. primarily sold software allowing users to automatically acquire, format, and present news and other content from certain online services. And in 1995, PointCast redirected its focus towards the development of the PointCast Network, which launched in 1996.³⁷¹ The company described the PointCast Network as follows:

The PointCast Network automatically appears whenever the computer is idle, replacing a screensaver with a constant stream of useful, personalized news and information. Viewers can effortlessly absorb headlines, stock quotes and other personalized news on screen or in the scrolling ticker, and can click on any headline to obtain in-depth information.³⁷²

318. PointCast's software used "push-technology" to retrieve information from the Internet and push it (via download) to the user's computer.³⁷³ The PointCast Network would run in the background of a user's computer and collect information free of charge.³⁷⁴ When the computer was idle, the PointCast Network acts as a "screen saver that draws a continuous stream of ... information from a server on the 'net."³⁷⁵ In 1996, the PointCast Network displayed information "channels" on a

³⁷¹ Ex. 1035 (PointCast Inc. 1998 SEC Filings), 28.

³⁷² Ex. 1035 (PointCast Inc. 1998 SEC Filings), 7; Ex. 1032 (Wall Street Journal PointCast article), 1.

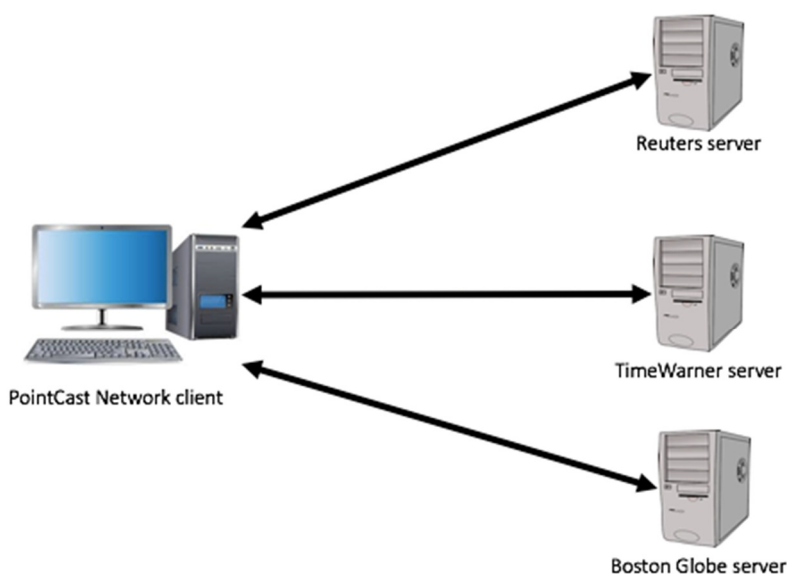
³⁷³ Ex. 1032 (Wall Street Journal PointCast article), 1.

³⁷⁴ Ex. 1033 (Computer World PointCast article), 2.

³⁷⁵ Ex. 1033 (Computer World PointCast article), 2; Ex.1034 (Christian Science Monitor PointCast article), 1.

screen saver such as news, weather, and stocks.³⁷⁶ These channels provided information derived from different sources including Reuters, TimeWarner, the Los Angeles Times, and the Boston Globe.³⁷⁷

319. As shown in the below exemplary figure, the PointCast Network application software would run on a client computer and connect to a Reuters server, a TimeWarner server, and a Boston Globe server.



PointCast Network Connections Between Client & Servers

320. Patentee's '903 Provisional, which is incorporated-by-reference in each of the Challenged Patents, acknowledges that it was known in the art that each clients' computer receives multiple, separate traffic flows associated with the PointCast Network application:

³⁷⁶ Ex. 1034 (Christian Science Monitor PointCast article), 2.

³⁷⁷ Ex. 1034 (Christian Science Monitor PointCast article), 2.

The state processor processes single and multi packet protocol recognition. It may have to search through a series of possible states to determine the flow's actual state. The result of this processing is a consolidated flow entry. This enables the monitor to correctly determine disjointed flows. For example, *a PointCast session* (PointCast, Inc., Cupertino, CA) *will open multiple conversations packet-by-packet that might look like separate flows to prior art monitors. However, each of these connections is merely a sub-flow under the PointCast master flow, so a single flow that consolidates all of the information for the flow is desired. The analyzer is able to so consolidate individual connections* since the state of the overall flow is maintained by the monitor.³⁷⁸

321. Patentee's '903 Provisional also acknowledges that PointCast traffic flows include a PointCast-specific identification signature: "During the initial connection between a PointCast server and client, specific key tokens exist in the data exchange that will result in a signature for PointCast."³⁷⁹

322. As I detail below, Riddle teaches creating a single flow to describe disjointed PointCast Network flows. During the district court trial between Patentee and NetScout, inventor Russell Dietz testified that creating a single flow to describe disjointed flows is a type of conversational flow (while referring to the below demonstrative):

³⁷⁸ Ex. 1016 ('903 Provisional), 7:16-25, 74:5-8.

³⁷⁹ Ex. 1016 ('903 Provisional), 28:22-24.

Q. Using this example that you have here, what was the problem that you and your group noticed in the mid-'90s time frame?

A. Well, the problem that we noticed was that more and more connection flows were -- were -- were related to what it is that you were doing in the application and how that was happening was missing. So -- so, in other words, you couldn't really see it. So let me show you another slide that gives you a view of what was going on. So basically, what I've done here is think of that app or that web page that you're using made up of lots of these different connection flows. And the problem is, is that each of them are different or -- or as becoming a very big problem because wireless network providers and other kinds of networks that were coming up in front of and around the Internet were having a very difficult time being able to make sure that the services that were provided to present those apps or web pages to you were being delivered and that they could figure out that that app that you're running is actually related to all of those connection flows.

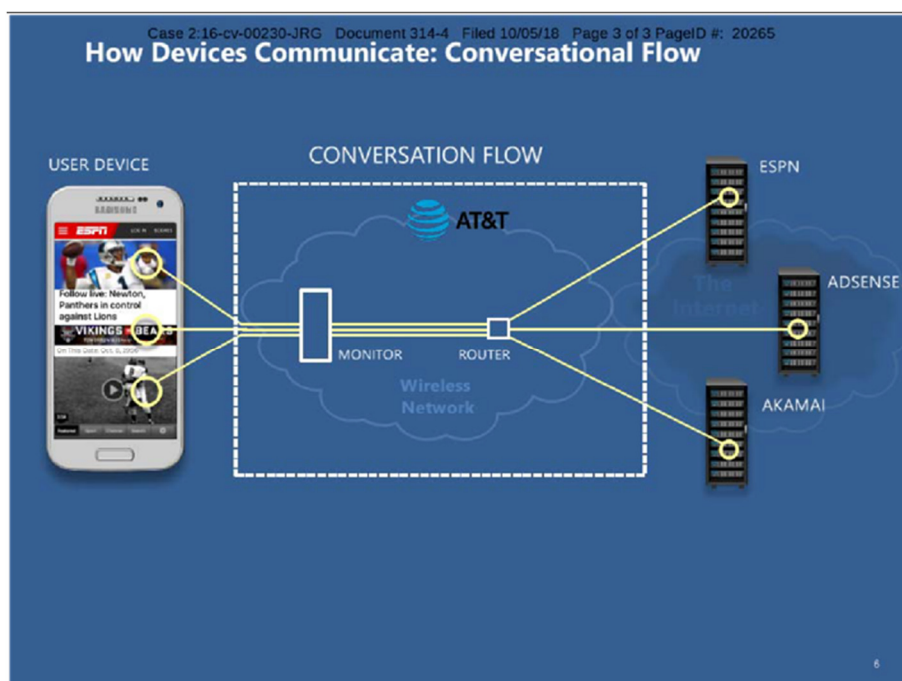
Q. And how did you and your team of other inventors solve that problem of not being able to group those different connections?

A. Well, you know, what we did was we came up with a way -- a new way of associating all of those packets that I -- that I described, pulling information out of all those packets that I described earlier, and -- and -- and associating that information back with all of these different -- these different connection flows. Basically, we created a new view into that app. And on -- on the -- on the next slide, I can show you what that looked like. So what we -- what we came up with was a way to take

information from all of those different packets in each of those connection flows and create a conversational flow. ***And the conversational flow, as we see in this picture, can be 3 or 300 or 30 different connection flows, but they're all associated now to that one application, the app on your phone and that web page.***

Q. And is conversational flow the term that's used in your team's patents?

A. Yes, it is.³⁸⁰



³⁸⁰ Ex. 1068 (10/10/17 Trial Transcript in *NetScout* district court case), 55:11-57:15); Ex. 1072 (R. Dietz Demonstrative Slide in *NetScout* district court case), 3; Ex. 1071 (M. Lyons Declaration in *NetScout* district court case), ¶4.

323. Riddle discloses creating a single flow class for multiple, separate PointCast flows.³⁸¹ Riddle describes examining packet flows for an indication of push traffic, such as PointCast traffic using HTTP to transport data.³⁸² Riddle teaches isolating and classifying PointCast's push traffic into a separate class.³⁸³ Riddle identifies PointCast flows by examining packet headers for signature content in the get request. By searching the packet headers for URLs that begin with "/FIDO-1/," Riddle teaches classifying these separate flows as PointCast flows.³⁸⁴ As was known in the art before the priority date of the Challenged Patents, tag FIDO-1 was used to fetch concatenated connection flows.³⁸⁵

324. Riddle discloses creating a single flow class for these disjointed PointCast flows. By searching packet headers for URLs that begin with "/FIDO-1/," Riddle teaches classifying these separate flows as PointCast flows:

In a preferable embodiment, classification can extend to examination of the data contained in a flow's packets. Certain traffic may be distinguished by a signature even if it originates with a server run on a non-standard port, for example, an HTTP conversation on port 8080 would not be otherwise determinable as HTTP from the port number. Further analysis of the data is conducted in order to determine classification in instances where: 1) FTP

³⁸¹ Riddle, 11:57-12:9.

³⁸² Riddle, 11:47-67.

³⁸³ Riddle, 11:47-67.

³⁸⁴ Riddle, 11:57-12:9.

³⁸⁵ Ex. 1036 (U.S. Patent No. 6,807,558), 30:62-31:17, 33:28-44, 39:14-40:21.

commands are used to define server ports, 2) HTTP protocol is used for non-web purposes. *The data is examined for indication of push traffic, such as pointcast, which uses HTTP as a transport mechanism. These uses may be isolated and classified into a separate class. Marimba and pointcast can be distinguished by looking into the data for a signature content header in the get request. Pointcast has URLs that begin with "/FIDO-1/."* Other applications in which protocol can be inferred from data include Telnet traffic. Both tn3270 and tn3270E (emulation) may be detected by looking into data and given a different class. Telnet traffic has option negotiations which may indicate an appropriate class.³⁸⁶

As was known in the art before the priority date of the Challenged Patents, tag FIDO-1 was used to fetch concatenated connection flows.³⁸⁷

325. Moreover, Riddle discloses that one of its autotclassification processes includes PointCast traffic and that a flow specification's outside service field may identify PointCast traffic.³⁸⁸ And Riddle specifies: "Network managers need not be aware of services which are known to be derivative of others, e.g., pointcast and marimba are special cases of HTTP and tn3270 is a special case of Telnet, in order to work with the system."³⁸⁹

326. As discussed above, the PointCast Network application software would run

³⁸⁶ Riddle, 11:47-67.

³⁸⁷ Ex. 1036 (U.S. Pat. No. 6,807,558), 30:62-31:17, 33:28-44, 39:14-40:21.

³⁸⁸ Riddle, 14:54-63, 15:16-20.

³⁸⁹ Riddle, 15:28-31.

on a client computer and connect to a series of servers. Riddle teaches creating a single class for such disjointed PointCast connections by searching packet headers for URLs that begin with /FIDO-1/.³⁹⁰ In doing so, Riddle classifies these connections as PointCast flows.

327. In sum, Patentee's '903 Provisional describes multiple flows being associated with the PointCast Network application. And Riddle teaches identifying and associating packet portions as PointCast traffic. Thus, by examining packet portions for URLs that begin with /FIDO-1/ to classify PointCast traffic, a POSITA would have understood that Riddle's parser subsystem forms a function of selected packet portions sufficient to identify that an accepted packet is part of a conversational flow-sequence.

f. '099 Claim Element 1.5: “(d) a memory storing a flow-entry database including a plurality of flow-entries for conversational flows encountered by the monitor”

328. Riddle renders obvious this claim element alone or in view of Ferdinand. As discussed with respect to '099 claim elements 1.1 and 1.2, Riddle discloses that its monitor includes storage subsystem 35 having memory subsystem 35a and file storage subsystem 35b for storing computer programs (e.g., code or instructions) and data.³⁹¹

³⁹⁰ Riddle, 11:11-23, 11:47-12:9, 15:16-20.

³⁹¹ Riddle, 6:1-23, 6:43-50, Figs. 1A-1B.

329. Riddle’s system stores flow-entries in a series of lists 308 that include a plurality of flow-entries encountered by the monitor. As illustrated below in annotated Figure 3, Riddle discloses a plurality of saved lists 308 that store classified traffic pending incorporation into traffic tree 302.³⁹² Riddle details that “entries for each instance of traffic may be kept” or “a copy of an entry and a count of duplicate copies for the entry is maintained.”³⁹³ Riddle retrieves flow-entries stored in lists 308, and incorporates the saved lists 308 into the classification tree, such as traffic tree 302, where each node of the tree represents a traffic class.³⁹⁴

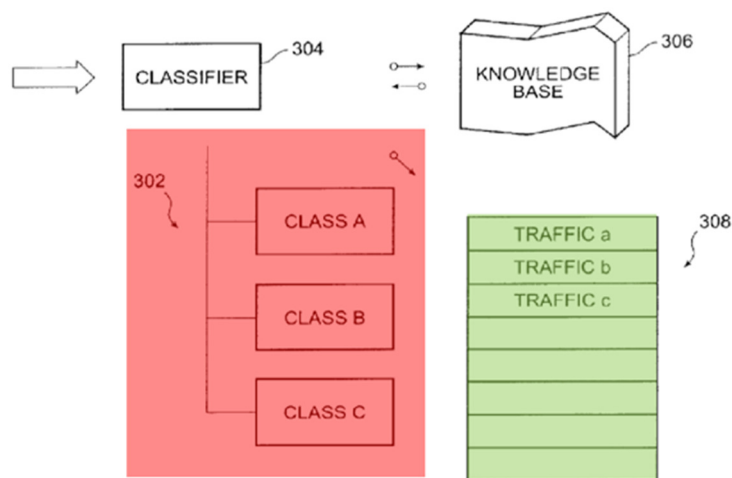


FIG. 3

330. Riddle’s Figure 4A, provided below, shows the process of parsing a flow specification from a packet, then storing the flow specifications in the saved list:

FIG. 3 depicts components of a system for automatically classifying traffic

³⁹² Riddle, 12:37-38, Fig. 3.

³⁹³ Riddle, 12:39-41.

³⁹⁴ Riddle, 9:28-33, 8:47-50, 13:35-62, Fig. 4B.

according to the invention. A traffic tree 302 in which new traffic will be classified under a particular member class node.... *A plurality of saved lists 308 stores classified traffic pending incorporation into traffic tree 302.* In select embodiments, entries for each instance of traffic may be kept. In alternate embodiments, a copy of an entry and a count of duplicate copies for the entry is maintained.

FIG. 4A depicts a flowchart 401 of processing steps for automatically classifying traffic. *In a step 402, a flow specification is parsed from the flow being classified. Then in a step 404, the flow specification parsed from the flow in step 402 is compared with the traffic specifications in each node of the classification tree.* Rules are checked starting from most specific to least specific. *In a decisional step 406, a determination is made if traffic matches one of the classes being classified. If this is so, then in a step 408, an entry is made in a list of identifying characteristics, such as protocol type (SAP), IP protocol number, server port, traffic type if known, MIME type, a time of occurrence of the traffic.* In an optional step 410, duplicate instances having the same identifying characteristics are suppressed, in favor of keeping a count of the duplicates and a most recent time traffic with these identifying characteristics was encountered. In an optional step 412, a byte count of traffic of this type has been detected is included....

In an optional step 413 (not show), after the processing of flowchart 401 completes or at periodic intervals or on demand, *a list of traffic classes produced in steps 402 through 412 are displayed to a network manager. The list may be sorted by any well-known criteria* such as: 1) most “hits” during a recent interval, 2) most recently-seen (most recent time first), 3)

most data transferred (bytes/second) during some interval, or a moving average.³⁹⁵

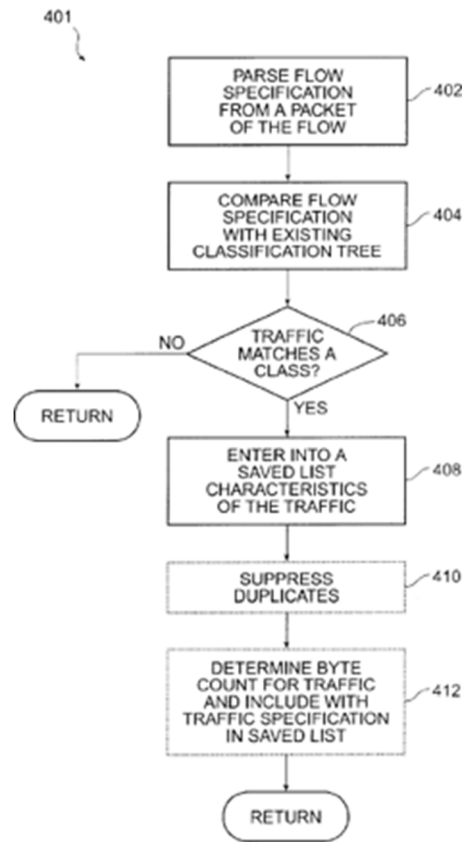


FIG. 4A

331. As I discuss above with respect to '099 claim element 1.3, Riddle teaches identifying a service aggregate flow as one type of traffic.³⁹⁶ Riddle explains that a service aggregate links together into a “conversation” multiple connection flows based on specific software program activity (e.g., Pointcast traffic).³⁹⁷ As such, Riddle discloses storing separate flow-entries for these previously encountered

³⁹⁵ Riddle, 12:27-13:5.

³⁹⁶ Riddle, 11:10-22, 13:53-59.

³⁹⁷ Riddle, 11:11-23, 11:60-63.

conversational flows, such as PointCast traffic.

332. Riddle's Figure 4B is a flowchart illustrating further processing of the packets parsed in Figure 4A to determine whether a traffic class, such as a service aggregate, needs to be created for the flow:

FIG. 4B depicts a flowchart 403 of the processing steps for integrating traffic classes into a classification tree in an alternative embodiment. Processing steps of flowchart 403 periodically at a defined interval of seconds, having a value of 30 in the preferable embodiment, incorporate newly classified traffic into the classification tree. ***In a step 420, an instance of saved traffic is retrieved from the saved traffic list 308.*** Next in a decisional step 422, the instance of saved traffic is examined to determine whether it is well-known (e.g. registered SAP, protocol type, assigned port number) and a name representing its type exists. If this is so then processing continues with a test of whether the saved traffic belongs to a service aggregate in step 426. Otherwise, in a step 423 the instance of saved traffic is examined to determine whether it appears to be a server connection port of an unregistered IP port (or a port that has not been configured). If this is not so then, processing continues with the next traffic class in the saved list in step 420. In decisional step 426, the instance of saved traffic is examined to determine whether it belongs to a service aggregate. For example, an FTP session has one flow that is used to exchange commands and responses and a second flow that is used to transport data files. If the traffic does belong to a service aggregate, then in a step 428, a traffic class is created which will match all components of the service aggregate. In a further step 425, a new traffic class is created to match the instance of saved traffic. The class may

be flat or hierarchical.³⁹⁸

333. As the above passage explains, Riddle’s monitor retrieves previously stored data from the saved lists (step 420, shown below in green).³⁹⁹ If the saved traffic is well known, Riddle’s monitor tests whether the retrieved traffic belongs to a service aggregate “conversational flow” (step 426, shown below in red).⁴⁰⁰ If it does, Riddle teaches creating a traffic class that “will match all components of the service aggregate” (step 428, shown below in red).⁴⁰¹

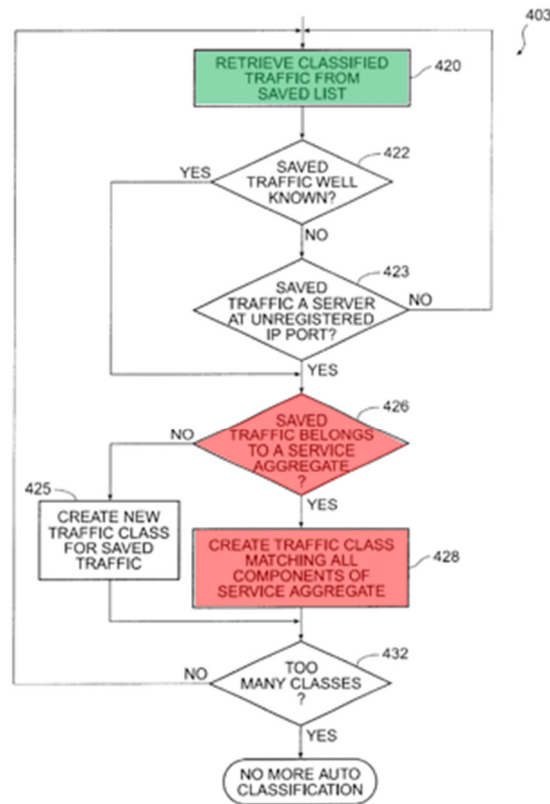


FIG. 4B

³⁹⁸ Riddle, 13:35-62, Fig. 4B.

³⁹⁹ Riddle, 13:40-52, 4:49-51, Fig. 4B.

⁴⁰⁰ Riddle, 13:52-56, Fig. 4B.

⁴⁰¹ Riddle, 13:56-59, Fig. 4B.

334. As such, based on Riddle's teachings regarding classification trees illustrated in Figures 4A-4B's flowcharts, a POSITA would have understood that Riddle's saved lists 308 store in memory a plurality of flow-entries.

335. A POSITA would have been motivated and found it obvious to store Riddle's lists 308, and related tree 302, in a flow-entry database based upon a POSITA's own knowledge of network devices. As discussed with respect to '099 claim element 1.2, Riddle describes databases, such as relational database 306, for storing the heuristics for determining traffic classes.⁴⁰² Further, it was well known to a POSITA to store data related to network traffic in a database in a network device. Routers, such as Riddle's "network routing means," were well known to use databases for storing routing information learned through protocols.⁴⁰³ Such databases allow for faster lookup of routing information. At the time of the prior art, such databases were often a set of hardware tables for even faster lookup.⁴⁰⁴ For routers routing information, Riddle discloses storing packet classification information in multiple lists.⁴⁰⁵ Thus, storing Riddle's flow-entries in a database amounts to nothing more than a simple implementation leading to the predictable

⁴⁰² Riddle, 12:32-35.

⁴⁰³ Riddle, 6:1-15; '099 Prosecution History, 213-214 (06/25/2003 Office Action, p.3).

⁴⁰⁴ Riddle, 6:1-15, 15:1-15; Ferdinand, 23:19-23, 28:16-24; '099 Prosecution History, 213-214 (06/25/2003 Office Action, p.3).

⁴⁰⁵ Riddle, 12:37-38, 12:61-63.

result of saving Riddle's classification information in a flow-entry database as a set of tables.

336. To the extent Riddle alone does not render obvious memory storing a flow-entry database, a POSITA would have been motivated and found it obvious to store Riddle's lists 308 in a flow-entry database based on Ferdinand's teachings. For example, Ferdinand describes a statistics database (STATS) 36 for storing classification information:

STATS 36 is responsible for the maintenance and initial analysis of the database.... STATS 36 is also responsible for tracking events of interest in the form of various statistical reductions.... STATS performs lookup on all addressing fields. It assigns new data structures to address field values not currently present. It performs any hashing for fast access to the database.⁴⁰⁶

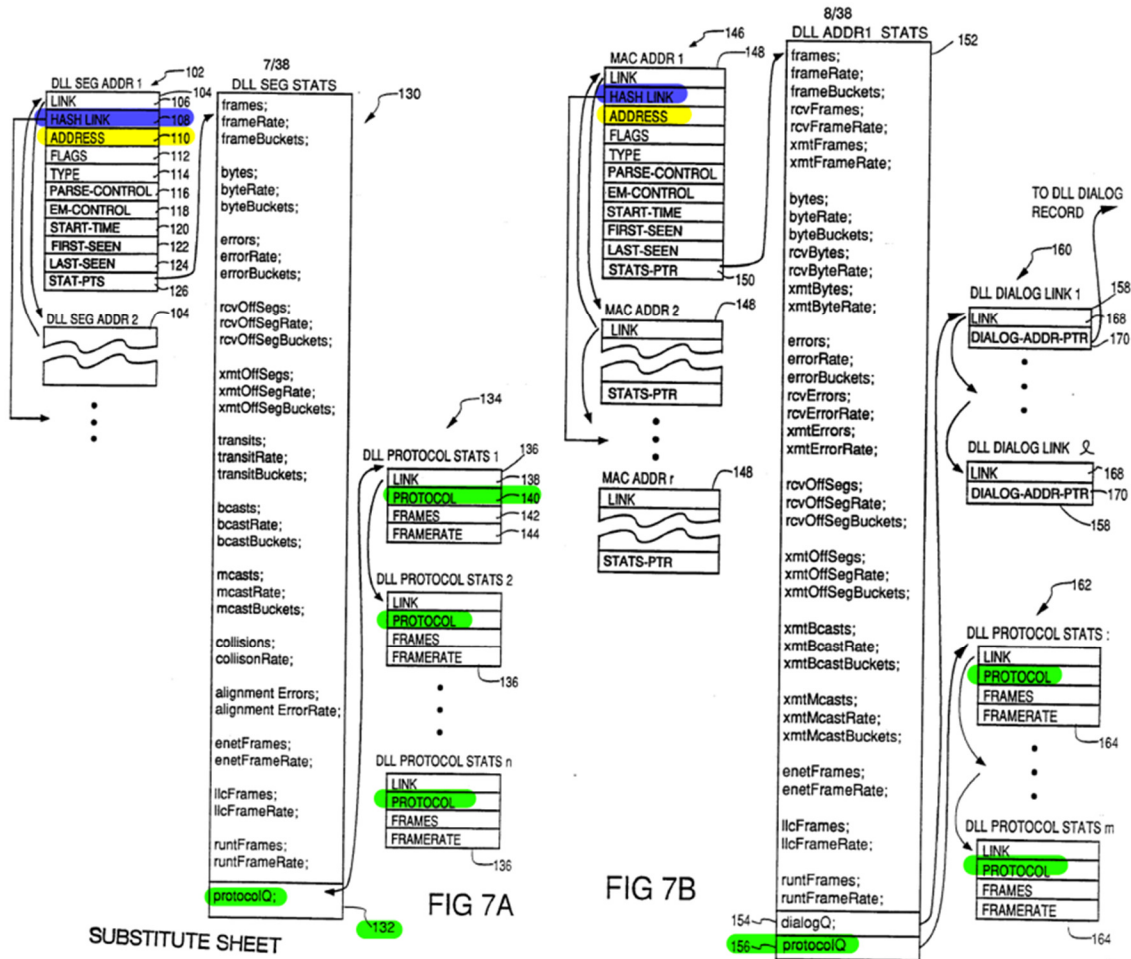
STATS defines the database and it contains subroutines for updating the statistics which it keeps. STATS contains the type definitions for all statistics records (e.g. DLL, IP, TCP statistics).⁴⁰⁷

337. Ferdinand illustrates the flow-entry database structure in Figures 7A-7C, provided below. For example, Ferdinand's flow-entry information includes protocol identifiers (shown in green), source and destination addresses (shown in yellow), and hashes (shown in blue), among others. Ferdinand also discloses that its

⁴⁰⁶ Ferdinand, 23:3-22.

⁴⁰⁷ Ferdinand, 28:14-17.

database's access routines allow for the hiding of the database's internal structure from other modules in the system: "Database accesses are generally performed using access routines. This hides the internal structure of the database from other modules and also ensures that appropriate interlocks are applied to shared data."⁴⁰⁸



⁴⁰⁸ Ferdinand, 27:16-19.

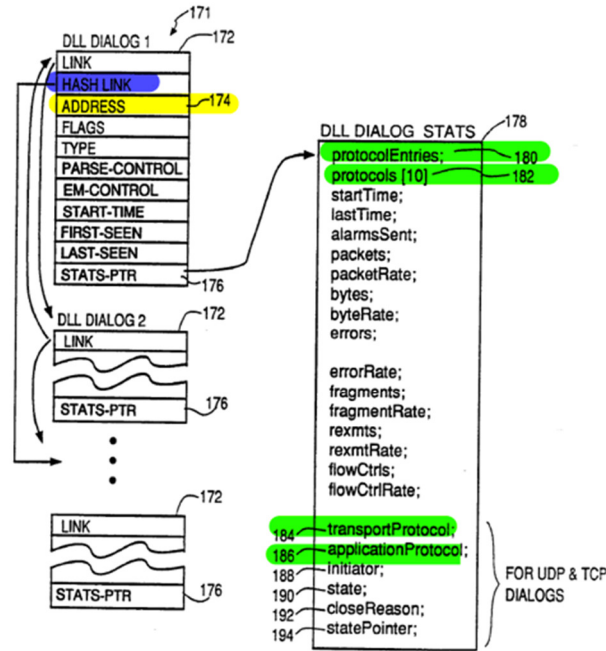


FIG 7C

338. As explained above, both Riddle and Ferdinand relate to devices that classify traffic. Riddle and Ferdinand both describe classifying the same types of traffic, such as FTP and other protocol types like TCP and UDP.⁴⁰⁹ And like Riddle, Ferdinand discloses displaying the results of its analysis to a user.⁴¹⁰ As such, a POSITA would have found it desirable to implement Riddle's saved lists in a flow-entry database, based on Ferdinand's teachings. A POSITA would have been motivated to do so because of the increased functionality of storing data in a database including searching, analyzing, and modifying the flow-entries. Such motivation

⁴⁰⁹ Riddle, 10:1-18 (Table 2); Ferdinand, 29:4-30:10, 39:23-40:16.

⁴¹⁰ Riddle, 12:64-13:9, 14:1-5; Ferdinand, 60:10-15, Fig. 22.

would further Riddle's desired goal of determining whether the packet monitor has received duplicate flow-entries.⁴¹¹ Moreover, this implementation would have allowed multiple network operators to access simultaneously Riddle's classification information consistent with Riddle's preference for storing multiple saved lists.⁴¹² A POSITA would have appreciated accessing Riddle's multiple list is similar to accessing a set of database tables because Riddle discloses its traffic classification information and heuristics may be stored in a file or relational database.⁴¹³ And Ferdinand specifies that using such a database allows for hiding the database's internal structure from other system modules, which would be another motivation to implement Riddle's saved lists in a flow-entry database.⁴¹⁴

339. Modifying Riddle's monitor would have led to predictable results given that Riddle's saved lists include similar information as that saved in Ferdinand's database. As such, including Riddle's classification information in a flow-entry database is nothing more than an obvious implementation to a POSITA based on Ferdinand's teachings.

⁴¹¹ Riddle, 12:53-57, Fig. 4A (step 410).

⁴¹² Riddle, 12:37-38, 12:61-63.

⁴¹³ Riddle, 12:34-35.

⁴¹⁴ Ferdinand, 27:16-19, 54:13-17.

- g. *'099 Claim Element 1.6: “(e) a lookup engine connected to the parser subsystem and to the flow-entry database, and configured to determine using at least some of the selected portions of the accepted packet if there is an entry in the flow-entry database for the conversational flow sequence of the accepted packet”*

340. Riddle discloses this claim element. As I detail below, Riddle describes using portions of packets to determine if there is a stored flow-entry. To make this determination, Riddle uses its processor and memory subsystem, which is the claimed “lookup engine.”

341. As discussed with respect to '099 claim elements 1.5, Riddle teaches a storage subsystem having a flow-entry database that includes a plurality of flow-entries for conversational flows encountered by the monitor or that such a flow-entry database is obvious. Riddle, for example, discloses a plurality of flow-entries in the form of a plurality of saved lists 308 storing classified traffic pending incorporation into traffic tree 302 and measurement data such as byte counts (step 412).⁴¹⁵ Riddle further states that “entries for each instance of traffic may be kept” or “a copy of an entry and a count of duplicate copies for the entry is maintained.”⁴¹⁶ And Riddle describes retrieving previously stored data from the saved lists (Figure 4B's step 420).

⁴¹⁵ Riddle, 12:37-38, Fig. 3.

⁴¹⁶ Riddle, 12:39-41, Fig. 4A.

342. As shown in Figure 4B, Riddle teaches looking up whether a flow matches a traffic class in relation to classifying a service aggregate based on a plurality of indicators.⁴¹⁷ Further, as shown in Figure 4A, Riddle describes comparing parsed packet information to store flow-entry information (step 404) to ultimately suppress duplicate entries when classifying network traffic (step 410). When suppressing duplicates for service aggregates (i.e., “conversational flows”), Riddle details that the suppression step 410 uses packet portions to determine if the packet belongs to a previously-encountered conversational flow sequence, such as service aggregate traffic.⁴¹⁸ Based on Riddle’s discussion of the related flowcharts in Figures 4A-4B, a POSITA would have understood that Riddle’s monitor uses packet portions to determine if there is a flow-entry for any previously-encountered conversational flow sequence and looks up whether a received packet belongs to a flow-entry (for example, class A) in traffic tree 302 corresponding to a conversational-flow sequence.

343. Further, when determining whether a flow-entry for a conversational flow already exists, Riddle discloses a device working with the parser subsystem to examine entries in the flow-entry database. As such, a POSITA would have understood that such a device is a lookup engine running on Riddle’s processor having

⁴¹⁷ Riddle, 13:42-47, claim 5, Fig. 4B.

⁴¹⁸ Riddle, 12:42-49.

programming code performing the above-described lookup functions: “The method for automatically classifying heterogeneous packets in a packet telecommunications environment of the present invention is implemented in the C programming language and is operational on a computer system such as shown in FIG. 1A.”⁴¹⁹ Moreover, Riddle describes a “processor means” that performs the function of matching a parsed flow specification to a traffic class.⁴²⁰

344. Further, a POSITA would have understood that Riddle’s elements for determining if the flow-entry database has a conversational flow-entry achieve Riddle’s goals of examining and classifying traffic.⁴²¹ As such, a POSITA would have understood that these lookup engine elements are connected to Riddle’s parser subsystem and flow-entry database.

- h. ’099 Claim Element 1.7: “(f) a state patterns/operations memory configured to store a set of predefined state transition patterns and state operations such that traversing a particular transition pattern as a result of a particular conversational flow-sequence of packets indicates that the particular conversational flow-sequence is associated with the operation of a particular application program, visiting each state in a traversal including carrying out none or more predefined state operations”*

345. Riddle discloses this claim element. As discussed with respect to ’099 claim

⁴¹⁹ Riddle, 5:53-57. As I detail above with respect to ’099 claims elements 1.3-1.4, Riddle’s operations run on a processor having programming code.

⁴²⁰ Riddle, claim 8.

⁴²¹ Riddle, 4:6-15.

elements 1.1, 1.2, and 1.5, Riddle discloses that its monitor includes storage subsystem 35 having memory subsystem 35a and file storage subsystem 35b storing “computer programs (e.g., code or instructions) and data.”⁴²² Riddle’s memory and file storage contain a set of predefined state transition patterns and state operations, such as those stored in knowledge base database 308.⁴²³

(1) Riddle teaches the claimed storing of “state transition patterns and state operations”

346. As discussed above in Section VI.D, Patentee has previously proposed to construe a “state operation” as “an operation to be performed while the state processor is in a particular state.” And as discussed in that section, the Challenged Patents provide examples of state operations, such as updating flow-entries, creating new flow-entries, reporting metrics, identifying flow’s application program, and searching records. Riddle teaches this claim element under Patentee’s proposed construction for “state operations” or that terms ordinary and customary meaning.

347. Further, the ’099 Patent states that “a state transition rule is a rule typically containing a test followed by the next-state to proceed to if the test result is true.”⁴²⁴ This is consistent with how a POSITA would have understood rules for traversing state transition patterns.

⁴²² Riddle, 6:1-23, 6:43-50, Figs. 1A-1B.

⁴²³ Riddle, 12:26-41, Fig. 3.

⁴²⁴ ’099 Patent, 15:2-4.

348. Riddle details that its monitor classifies flows by “a series of steps through a traffic class tree, with the last step (i.e., at the leaves on the classification tree) mapping the flow to a policy.”⁴²⁵ And such class trees are common data classification structures in which each of the tree’s leaf nodes correspond to last encountered flow states. At the time of the purported invention, a POSITA would have understood that Riddle’s steps walk through a set of state transition patterns and state operations. For example, Riddle details checks nodes in the classification tree to match a packet’s flow identification information to a given traffic class:

A classification tree is a data structure representing the hierarchical aspect of traffic class relationships. Each node of the classification tree represents a class, and has a traffic specification, i.e., a set of attributes or characteristics describing the traffic associated with it. Leaf nodes of the classification tree may contain policies. According to a particular embodiment, *the classification process checks at each level if the flow being classified matches the attributes of a given traffic class. If it does, processing continues down to the links associated with that node in the tree. If it does not, the class at the level that matches determines the policy for the flow being classified.* If no policy specific match is found, the flow is assigned the default policy.

In a preferred embodiment, *the classification tree is an N-ary tree with its nodes ordered by specificity.* For example, in classifying a particular flow

⁴²⁵ Riddle, 9:20-25.

in a classification tree ordered first by organizational departments, the attributes of the flow are compared with the traffic specification in each successive department node and if no match is found, then processing proceeds to the next subsequent department node. If no match is found, then the final compare is a default “match all” category. If, however, a match is found, then classification moves to the children of this department node. The child nodes may be ordered by an orthogonal paradigm such as, for example, “service type.” Matching proceeds according to the order of specificity in the child nodes. Processing proceeds in this manner, traversing downward and from left to right in FIGS. 2A and 2B, which describe a classification tree, searching the plurality of orthogonal paradigms. *Key to implementing this a hierarchy is that the nodes are arranged in decreasing order of specificity. This permits search to find the most specific class for the traffic before more general.*⁴²⁶

349. To illustrate Riddle’s state transition patterns and state operations, it is worth outlining Riddle’s teachings regarding its exemplary classification trees shown in Figures 2A, 2B, and 3. Figures 2A and 2B depict bandwidth allocations between Departments A and B.⁴²⁷ As provided below, Figure 2A shows Riddle’s packet classifier traverses a traffic tree to check traffic classes for Departments A and B. At resource node 202, Riddle’s classifier tests if the instant flow is intended for Department A inside host subnet A (in yellow) by comparing, for instance, the

⁴²⁶ Riddle, 9:28-63.

⁴²⁷ Riddle, 10:19-24.

packet's source (client) IP to the range of IP address defined for subnet A.⁴²⁸ If so, then the state is updated and the classifier proceeds to the next state operation (in blue) to test if the flow is for an FTP outside port 2.0 (206) or the web (208).⁴²⁹ If the flow was not for Department A, then the classifier proceeds to the next state operation (in yellow) to test if the flow is intended for Department B inside host subnet B (204) by comparing the packet's source (client) IP to the range of IP addresses defined for subnet B.⁴³⁰ If so, then the state is updated and the classifier proceeds to the next state operation (in blue) to test if the flow is for an FTP server (210) or the web (212). If the flow was not for Departments A or B, then the classifier gives the flow a default classification (205).⁴³¹ These tests are state transitions indicating whether a particular conversational flow is associated with a particular application program, e.g., an FTP program or web browser.⁴³² As such, Riddle's Figure 2A illustrates a set of rules with each node containing a test followed by the next state if the test result is true.

⁴²⁸ Riddle: 10:28-39.

⁴²⁹ Riddle: 10:28-39.

⁴³⁰ Riddle: 10:19-39, Table 2.

⁴³¹ Riddle: 10:19-56, Table 2.

⁴³² Riddle, 10:19-39.

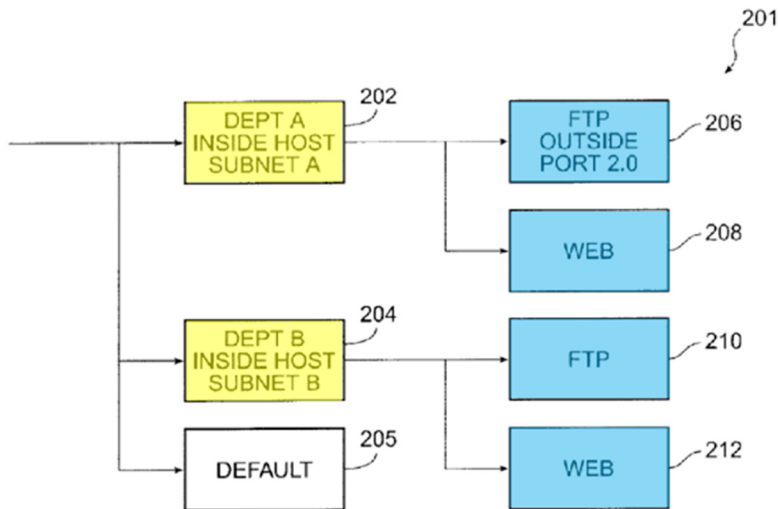


FIG. 2A

350. As provided below, Figure 2B shows another classification tree in which Riddle's packet classifier tests if the instant flow is web traffic (220 in yellow).⁴³³ If so, then the state is updated and classifier proceeds to the next state operation (in blue) to test if the flow is for Department A (226) or Department B (228).⁴³⁴ If the flow is not web traffic (220), then the classifier proceeds to the next state operation (in yellow) to test if the flow is a TCP flow (224).⁴³⁵ If so, then the state is updated and the classifier proceeds to the next state operation (in blue) to test if the flow is for Department A (230) or Department B (232).⁴³⁶ If the flow was not a web or TC flow, then the classifier gives the flow a default classification (225).⁴³⁷ These tests

⁴³³ Riddle, 10:40-51.

⁴³⁴ Riddle, 10:40-51.

⁴³⁵ Riddle, 10:40-51.

⁴³⁶ Riddle, 10:40-51.

⁴³⁷ Riddle, 10:40-56.

are state transitions indicating whether a particular conversational flow is associated with a particular application program, e.g., an FTP program or web browser.⁴³⁸ Again, Riddle's Figure 2B illustrates a set of rules with each node containing a test followed by the next state if the test result is true.

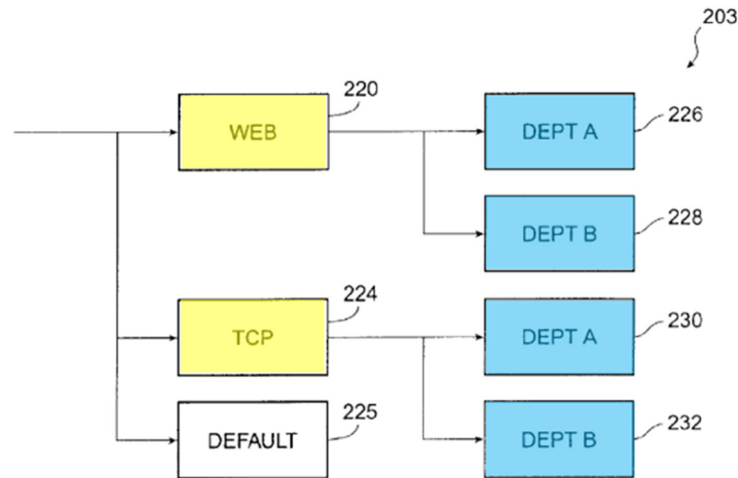


FIG. 2B

351. Further, Riddle teaches displaying a hierarchical classification tree that lists a stored set of pre-defined state transition patterns and state operations related to FTP. For example, Riddle describes classes “to host 1,” “tcp,” and “FTP.”⁴³⁹ As each packet is received, Riddle's classification proceeds along this hierarchy leading to matching of a flow with the operation of a particular FTP application program. Based on these teachings, a POSITA would have understood that transition-

⁴³⁸ Riddle, 10:19-39.

⁴³⁹ Riddle, 13:11-22.

ing from one class to the next in Riddle's hierarchy involves traversing set of pre-defined state transition patterns and state operations (e.g., saving TCP-session information from classified TCP packets exchanged with the host; and aggregating saved TCP-sessions belonging to the instance of FTP application being classified).

352. As discussed above in Section IV.A.1, Riddle incorporates-by-reference the teachings of Packer as though fully set forth in Riddle. Packer further describes a traffic classification system illustrated by the flowchart of Figure 5F, provided below. In describing this flowchart, Packer specifies traversing a set of predefined state-transition patterns and state operations by recursively processing through matching child-class definitions:

FIG. 5F depicts a flowchart 511 showing the component steps of traffic classification step 504 of FIG. 5A. The processing steps of flowchart 511 determine a class and a policy for a flow, such as new flow 300, by traversing a classification tree such as the classification tree 201 in FIG. 2A. Processing begins with a first node in the classification tree 201, such as department A node 202. In a step 570, a traffic specification of a child node, such as FTP node 206 is compared with a flow specification of the new flow 300. In a decisional step 572, if a match is discovered, then in a step 574, the processing of flowchart 511 is applied to the child node 206 recursively. Otherwise, if child node 206 does not match the new flow, then in a decisional step 576, processing determines whether any sibling nodes exist for the child node 206. If processing detects the existence of a sibling of child node 206, such as

child node 208, then processing continues on node 208 with step 570. Otherwise, if there are no further child nodes for node 202, then in a decisional step 578, a determination is made whether the node is a leaf node having a policy. If no policy exists, then in a step 580, processing backtracks to a parent node and looks for a policy associated with the parent node to apply to the new flow 300. Otherwise, if a policy is associated with the node 202, then in a step 582, the policy is associated with the new flow 300.⁴⁴⁰

⁴⁴⁰ Ex. 1031 (Packer), 18:1-26.

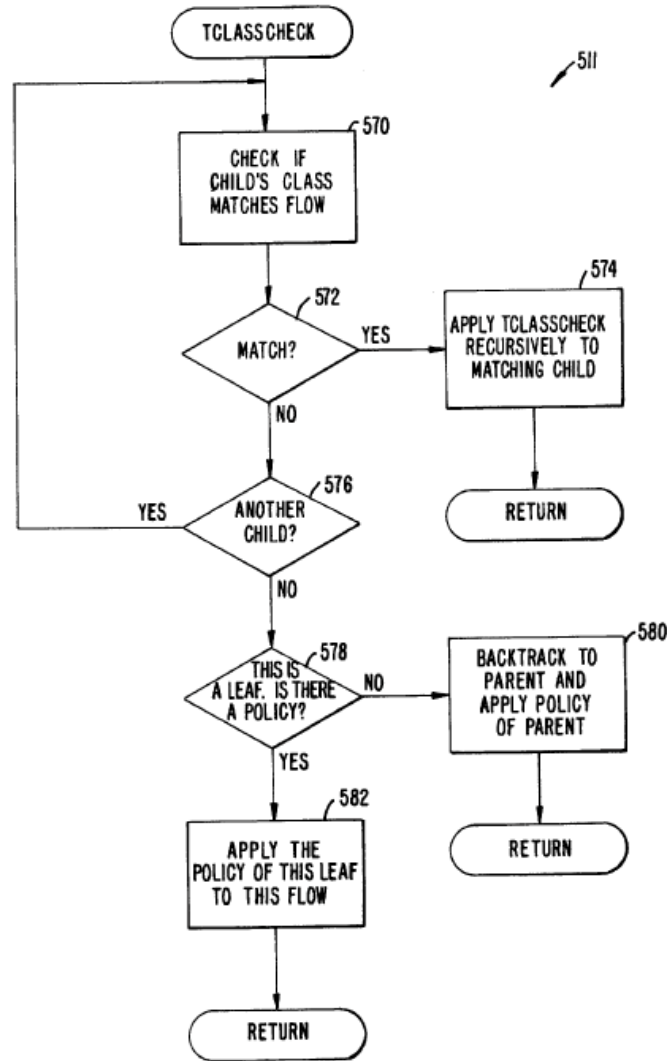


FIG. 5F.

353. Below, I have annotated Riddle’s Figure 4B to illustrate traversing a state transition pattern for a particular conversational flow-sequence. Illustrated by steps 426-428 (in red), Riddle’s classification tree can include a state operation of determining if the traffic belongs to a “service aggregate” application (i.e., “conversational flow-sequence) with multiple connections between computers.⁴⁴¹

⁴⁴¹ Riddle, 11:10-23.

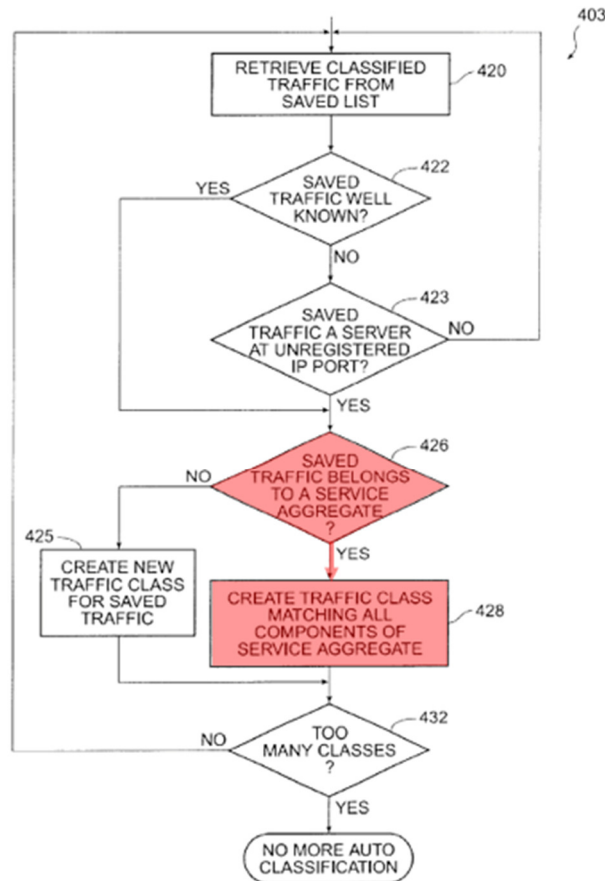


FIG. 4B

- (2) Riddle teaches the claimed “traversing a particular transition pattern as a result of a particular conversational flow-sequence of packets indicates that the particular conversational flow-sequence is associated with the operation of a particular application program, visiting each state in a traversal including carrying out none or more predefined state operations”

354. For the flowchart steps shown in Figure 4A, Riddle describes analyzing information that identifies the characteristics of the traffic as the classifier parses the flow’s packets and matching the parsed packets to a class:

FIG. 4A depicts a flowchart 401 of processing steps for automatically classifying traffic. In a step 402, a flow specification is parsed from the

flow being classified. Then in a step 404, the flow specification parsed from the flow in step 402 is compared with the traffic specifications in each node of the classification tree. Rules are checked starting from most specific to least specific.⁴⁴²

355. Further, Riddle describes that, after an initial classification, its classifier performs “subclassification” in a sequential manner through finer-grade matchings as the classifier identifies characteristics, such as identifying the hosts and services, leading to matching of a flow with the operation of a particular application program.⁴⁴³ In accordance with the transition pattern of the classification tree, Riddle further teaches its classifier advances through the sequence of packets of a particular flow to parse and classify those packets. This results in Riddle’s classifier performing a corresponding predefined state operation at each node to update the identifying characteristics of the flow.

356. Returning to the example of FTP operations above in Riddle’s Figure 2A, Riddle describes traversing a particular transition pattern that includes (1) comparing the packets’ source (client) IP to the range of IP addresses defined for subnet B and, if so, (2) determining if the sequence of packets involves the FTP protocol.⁴⁴⁴ In doing so, Riddle teaches testing for subclassifications when encountering packet

⁴⁴² Riddle, 12:42-48.

⁴⁴³ Riddle, 11:25-31, 13:11-22.

⁴⁴⁴ Riddle, 10:19-39.

information relating to FTP applications such as Figure 2A's classes 206, 210:

Subclassification of traffic into a tree is performed by matching the hosts and then searching for particular services. Traffic *specifications are aggregate kinds of traffic for a traffic class, e.g., different components of FTP may reside under class FTP. Subclassification is performed by first locating a class that matches, and then performing finer grade matchings.* Processing commences with a decision on what traffic is to be subclassified. A marker is placed in the match_all default node so that when match processing reaches the marker, the autotclassification processing depicted in flowchart 403, determines that it has not found an existing class for the traffic being classified.⁴⁴⁵

When testing for such subclassifications, Riddle's classifier performs an operation to determine if the flow is an FTP command flow or an FTP data flow.⁴⁴⁶

357. Moreover, with flows involving FTP applications, Riddle teaches performing state operations to determine if the flow belongs to a service aggregate (i.e., a conversational flow-sequence).⁴⁴⁷ As Riddle details:

A service aggregate is provided for certain applications that use more than one connection in a particular conversation between a client and a server. For example, an FTP client in conversation with an FTP server employs a command channel and a transfer channel, which are distinct TCP sessions on two different ports. In cases where two or three TCP

⁴⁴⁵ Riddle, 11:24-36.

⁴⁴⁶ Riddle, 11:12-15.

⁴⁴⁷ Riddle, 11:10-23; 13:52-57; Fig. 4B.

or UDP sessions exist for each conversation between one client and one server, it is useful to provide a common traffic class i.e., the service aggregate, containing the separate conversations.⁴⁴⁸

As shown in Figure 2A, a POSITA would have understood that traversing this example transition pattern (Client IP/Subnet B/FTP/FTP-cmd/FTP service aggregate) as a result of a particular conversational flow-sequence of packets, indicates an association with the FTP-application program initiated on the user's (client's) computer.⁴⁴⁹

358. Further, Riddle describes additional examples of traversing a transition pattern resulting from a particular conversational flow-sequence of packets. One example is Riddle's examined packet relating PointCast traffic. As discussed above regarding '099 claim element 1.4, Riddle's conversational flow-sequences are associated with the operation of a particular application program, such as PointCast Network application software.

359. The Challenged Patents explain that searching for one or more patterns in the parsed packet information is an example of a state operation.⁴⁵⁰ Riddle discloses searching for patterns/strings in HTTP headers and data: "Web traffic may

⁴⁴⁸ Riddle, 11:10-23.

⁴⁴⁹ As explained regarding '099 claim element 1.4, Riddle's "conversational flow-sequence[s]" are associated with the operation of a particular application program.

⁴⁵⁰ '646 Patent, claim 13; '789 Patent, claims 17, 27, 46.

also be classified by HTTP header types such as Content-Type (MIME type) or User-Agent.”⁴⁵¹ Thus, when checking whether a flow is for web applications at Figure 2A’s boxes 208 or 212, a POSITA would have understood that Riddle’s classifier performs a state operation that searches for one or more patterns, such as searching for URI patterns matching “/FIDO-1/” indicating PointCast traffic.⁴⁵² To a POSITA, this classification pattern identifies a single PointCast client accessing multiple news-type servers.⁴⁵³ As shown in Figure 2A, Riddle’s sequence of packets traversing this example transition pattern (Client IP/Subnet B/Web/HTTP/Pointcast) indicates an association with the PointCast application program initiated on the user’s computer.⁴⁵⁴

360. Another example of traversing a transition pattern resulting from a particular conversational flow-sequence of packets is Riddle creating a new flow-entry for previously unencountered traffic flows. The Challenged Patents explain that state operations may include creating a new flow-entry for future packets to be identified with the flow.⁴⁵⁵

⁴⁵¹ Riddle, 9:24-26; 8:67-9:11, 11:48-59.

⁴⁵² Riddle, 11:57-63, 14:53-64, 15:16-31.

⁴⁵³ Ex. 1034, 2.

⁴⁵⁴ As I explain above with respect to ’099 element 1.4, each of Riddle “conversational flow-sequence” is associated with the operation of a particular application program.

⁴⁵⁵ ’789 Patent, claim 47.

361. When classifying flows, Riddle teaches creating a new flow-entry for previously unencountered traffic flows and suppressing duplicate entries at Figure 4A's steps 408 and 410:

In a decisional step 406, a determination is made if traffic matches one of the classes being classified. If this is so, then in a step 408, an entry is made in a list of identifying characteristics, such as protocol type (SAP), IP protocol number, server port, traffic type if known, MIME type, a time of occurrence of the traffic. In an optional step 410, duplicate instances having the same identifying characteristics are suppressed, in favor of keeping a count of the duplicates and a most recent time traffic with these identifying characteristics was encountered.⁴⁵⁶

362. Thus, when checking whether a flow is for web or FTP applications at Figure 2A's boxes 206, 208, 210, or 212, a POSITA would have understood that Riddle's classifier performs a state operation every time it creates a new flow-entry per step 408.

363. Yet another example of traversing a transition pattern resulting from a particular conversational flow-sequence of packets is Riddle determining metrics relating to parsed packet information. The Challenged Patents explain that state operations may include determining metrics that relate to the examined flow.⁴⁵⁷

364. Riddle discloses determining and reporting metrics related to flows, such as

⁴⁵⁶ Riddle, 12:48-57.

⁴⁵⁷ '751 Patent, claims 11-13, 16.

byte count, most hits, time most recently seen, most data transferred, moving average, bytes per second, most recently used, most hits, and number of bytes received.⁴⁵⁸ For example, Riddle states “[i]n an optional step 412, a byte count of traffic of this type has been detected is included.”⁴⁵⁹ Riddle also teaches sorting packet information based on determined metrics:

The list may be sorted by any well-known criteria such as: 1) most “hits” during a recent interval, 2) most recently-seen (most recent time first), 3) most data transferred (bytes/second) during some interval, or a moving average. The user may choose an interval length or display cutoff point (how many items, how recent, at least B bytes per second, or other thresholds).⁴⁶⁰

365. Further, Riddle describes using time stamp metrics to organize traffic classes: “In a further step 425, a new traffic class is created to match the instances of saved traffic.... In a related embodiment in place of step 425, a display of traffic classes, sorted by most recently used, most hits, number of bytes received during any interval, which is determined by a plurality of time stamps, is available on demand to a network manager.”⁴⁶¹

⁴⁵⁸ Riddle, 12:53-13:8, 14:1-5.

⁴⁵⁹ Riddle, 12:57-59.

⁴⁶⁰ Riddle, 13:1-8.

⁴⁶¹ Riddle, 13:59-14:5.

366. A further example of traversing a transition pattern resulting from a particular conversational flow-sequence of packets is Riddle updating flow-entries. The Challenged Patents explain that state operations may include updating a flow-entry.⁴⁶²

367. Riddle describes updating flow-entries when encountering pertinent parsed packet information:

In an optional step 410, duplicate instances having the same identifying characteristics are suppressed, in favor of keeping a count of the duplicates and *a most recent time traffic with these identifying characteristics was encountered*. In an optional step 412, *a byte count of traffic of this type has been detected is included*.⁴⁶³

And Riddle's flowchart details the monitor will "parse flow specification from a packet of the flow" (step 402), "compare flow specification with existing classification tree" (step 404), determine if "traffic matches a class?" (step 406), "enter into a saved list characteristics of the traffic" (step 408), "suppress duplicates" (step 410), and "determine byte count for traffic and include with traffic specification in saved list" (step 412).⁴⁶⁴

368. For existing flows, Riddle describes storing "a count of the duplicates and a

⁴⁶² '646 Patent, claim 15; '751 Patent, claim 13; '789 Patent, claims 15, 30, 45.

⁴⁶³ Riddle, 12:53-59.

⁴⁶⁴ Riddle, Figure 4A.

most recent time traffic with these identifying characteristics was encountered.”⁴⁶⁵

The time stored shows the last encountered state of the flow. When suppressing duplicates, a POSITA would have understood that Riddle teaches performing state operations relating to flow-entry updating: the count of the duplicates, the most recent time traffic with the same identifying characteristics was encountered, and the byte count of the detected traffic.

369. Accordingly, a POSITA would have understood Riddle describes a stored set of predefined state transition patterns and operations where each state is visited during transversal. And, for the reasons described above, a POSITA would have understood that these states relate to an indication of all previous events in the flow that lead to recognition of the content of all of the protocol levels.

- (3) Riddle in view of Ferdinand renders obvious claimed “state patterns/operations memory” being separate from other claimed memories

370. To the extent Patentee argues ’099 claim elements 1.1-1.4, and 1.6 require separate physical memories, i.e., one packet-buffer memory, one parsing/extraction operations memory, and one flow-entry database memory, the combination of Riddle and Ferdinand renders these claim elements obvious. Regarding the packet-buffer memory recited in ’099 claim element 1.1, a POSITA would have found it a simple implementation to have a distinct packet-buffer memory based on the

⁴⁶⁵ Riddle, 12:56-57.

teachings of Riddle and Ferdinand.

371. Regarding the parsing/extraction operation memory and the flow-entry database memory, using different memories for different functionalities was well understood by a POSITA.⁴⁶⁶ Indeed, Riddle already discloses two distinct memories, a memory subsystem 35a and a file storage subsystem 35b.⁴⁶⁷ Thus, a POSITA would have found it a simple implementation to have a distinct operation memory and flow-entry database memory based on the teachings of Riddle and Ferdinand.

372. Using different memories for different functionalities was well understood by a POSITA.⁴⁶⁸ Ferdinand confirms this by disclosing dividing memory into four separate blocks with each block devoted to a different functionality.⁴⁶⁹ The advantages of using different memories for different functionalities was well known to a POSITA.⁴⁷⁰ And a POSITA would have been motivated to provide Riddle's high speed, real-time monitor with dedicated memories because, for example, dedicating memories to processors or processing functions ensures memory access times are reduced and system performance increased due to reduced contention for

⁴⁶⁶ '099 Prosecution History, 213-214 (06/25/2003 Office Action, p.3).

⁴⁶⁷ Riddle, 6:5-8, Fig. 1A.

⁴⁶⁸ '099 Prosecution History, 213-214 (06/25/2003 Office Action, p.3); Riddle, 6:5-8, Fig. 1A.

⁴⁶⁹ Ferdinand, 26:2-18.

⁴⁷⁰ '099 Prosecution History, 213-214 (06/25/2003 Office Action, p.3); Riddle, 6:5-8, Fig. 1A.

memory.⁴⁷¹

- i. *'099 Claim Element 1.8: “(g) a protocol/state identification mechanism coupled to the state patterns/operations memory and to the lookup engine, the protocol/state identification engine⁴⁷² configured to determine the protocol and state of the conversational flow of the packet; and*

373. Riddle discloses this claim element. As I discuss above, Riddle teaches the following claim elements:

- Lookup engine (element 1.6);
- Examining packets and determining the protocols used in the packets (element 1.2);
- Identifying packets as part of a conversational-flow sequence (element 1.4); and
- State patterns/operations memory and determining the state of the conversational flow of a packet (element 1.7).

To collectively execute these operations, Riddle describes a processor and code, which is the claimed “protocol/state identification mechanism.” My above discussions regarding claim elements 1.2, 1.4, and 1.6-1.7 show how Riddle discloses a protocol/state identification mechanism coupled to the state patterns/operations

⁴⁷¹ '099 Prosecution History, 213-214 (06/25/2003 Office Action, p.3); Riddle, 6:5-8, Fig. 1A.

⁴⁷² As best understood, it appears that the claimed “protocol/state identification engine” should read “protocol/state identification mechanism.”

memory and to the lookup engine. Moreover, these discussions show Riddle’s protocol/state identification engine determines the protocol and state of the conversation flow of the packet.

374. Riddle specifies that its “[t]raffic classes may be defined at any level of the IP protocol as well as for other non-IP protocols.”⁴⁷³ As I discuss above with respect to claim element 1.7, Riddle discloses determining protocol (e.g., IP, TCP, etc.) and the state of the conversational flow via traffic classifier 304, and processing through Riddle’s classification trees 302 stored in knowledge base 306.⁴⁷⁴

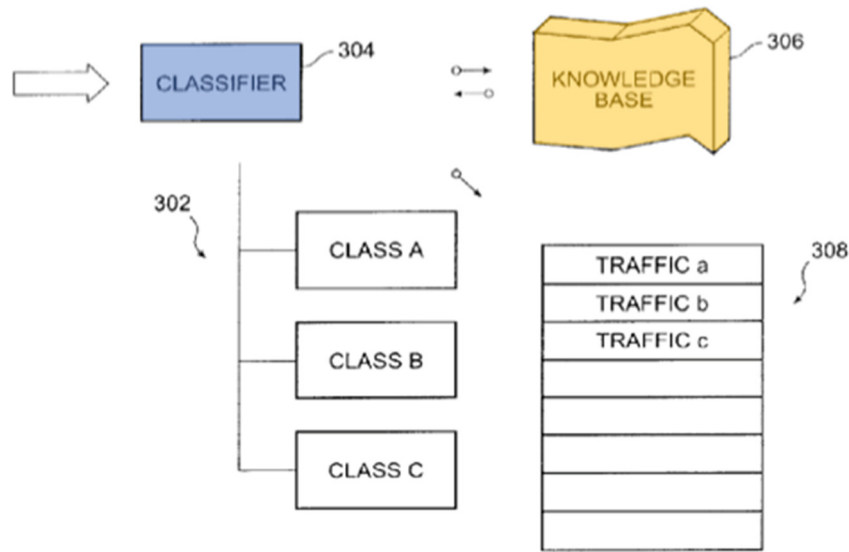


FIG. 3

375. As I discuss above, by searching for patterns/referencing strings in headers and data, Riddle determines the packet’s protocols and states.⁴⁷⁵ For example, a

⁴⁷³ Riddle, 8:58-59.

⁴⁷⁴ Riddle, 12:27-36.

⁴⁷⁵ Riddle, 9:24-26, 11:48-49.

POSITA would have understood that an IP's header's IP-protocol field indicates TCP or UDP for Riddle's monitor.⁴⁷⁶ Another examples are well-known ports indicate application-layer protocols (e.g., SMTP or HTTP), and HTTP data containing "/FIDO-1/" indicates PointCast traffic.⁴⁷⁷ As the conversation-flow sequence of packets traverse Riddle's classification tree, a POSITA would have understood that the state is updated upon making each classification node match, and determining the next branch of classification.⁴⁷⁸

376. As discussed with respect to claim element 1.2 and 1.7, Riddle discloses a processor and corresponding code for determining the protocol and state operations. For example, Riddle teaches: "The method for automatically classifying heterogeneous packets in a packet telecommunications environment of the present invention is implemented in the C programming language and is operational on a computer system such as shown in FIG. 1A."⁴⁷⁹ Based on these teachings, a POSITA would have recognized Riddle's processor and corresponding code is the claimed "protocol/state identification [mechanism/engine]." And a POSITA would have understood that Riddle's protocol/state identification mechanism, state patterns/operations memory, and lookup engine work together to achieve Riddle's

⁴⁷⁶ Riddle, 3:51-54.

⁴⁷⁷ Riddle, 11:57-62.

⁴⁷⁸ Riddle, 10:59-12:63; Figs. 2A, 2B, 3, 4A, 4B.

⁴⁷⁹ Riddle, 5:53-57.

goal of classifying traffic.⁴⁸⁰

377. As discussed above in Section IV.A.1, Riddle incorporates-by-reference the teachings of Packer as though fully set forth in Riddle. As shown below in annotated Figure 3, Packer describes packet acquisition taking place and a “protocol finite state machine” examining the packet to determine whether the flow is new (300), the flow is ending (298), or belongs to an existing flow (299). This further illustrates Riddle teaches the monitor storing a set of predefined state transition patterns and operations.

⁴⁸⁰ Riddle, 4:6-15.

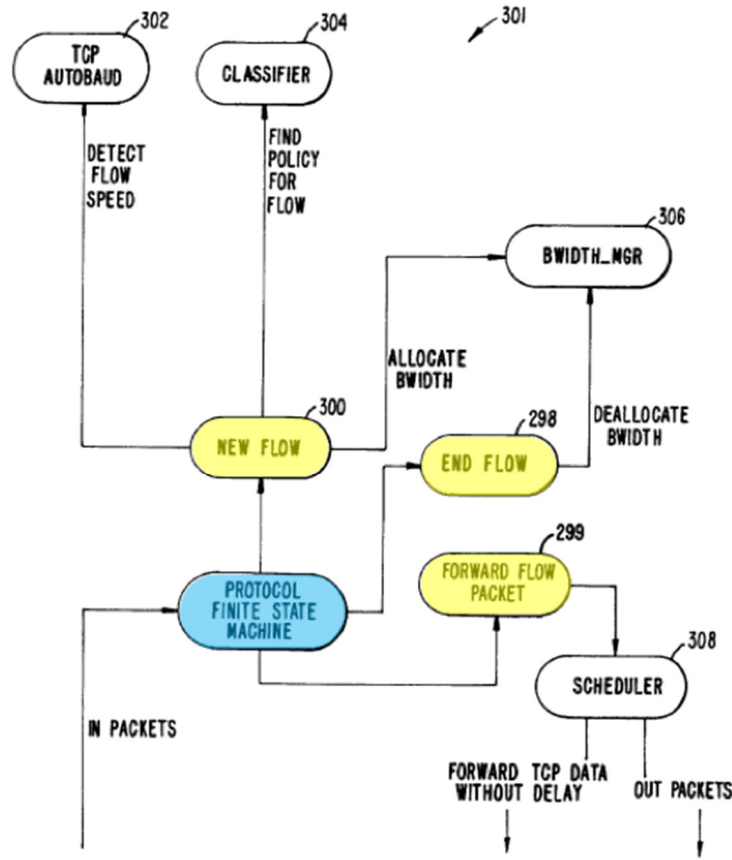


FIG. 3.

- j. '099 Claim Element 1.9: “(h) a state processor coupled to the flow-entry database, the protocol/state identification engine, and to the state patterns/operations memory, the state processor, configured to carry out any state operations specified in the state patterns/operations memory for the protocol and state of the flow of the packet”

378. Riddle discloses this claim element. As discussed with respect to claim element 1.5, Riddle alone or in view of Ferdinand renders obvious storing a flow-entry database. And as discussed with respect to claim elements 1.7-1.8, Riddle describes a protocol/state identification mechanism, state patterns/operations memory, and how the state operations are carried out for the protocol and state of

the flow. For executing these operations, Riddle teaches a processor and code, which is the claimed “state processor.”⁴⁸¹

379. As I discuss above with respect to claim element 1.8, Riddle discloses that processor 30 performs specified state operations for the protocol and flow state during the packet classification process discussed: “The method for automatically classifying heterogeneous packets in a packet telecommunications environment of the present invention is implemented in the C programming language and is operational on a computer system such as shown in FIG. 1A.”⁴⁸² As illustrated in Figure 1A, a POSITA would have understood that Riddle’s processor is coupled to components like protocol/state identification engine, state patterns/operations memory, and the memory storing flow-entry lists.⁴⁸³

380. Further, Riddle describes the processor parses the packet to identify the flow specification and then carries out state operations to match the packet’s specification to traffic flow classes:

A system for automatically classifying traffic in a packet telecommunications network, said network having any number of flows, including zero, comprising:

⁴⁸¹ Regarding processors, the ’099 Patent states that a POSITA could implement the individual hardware elements of the Challenged Patents with software running on a single processor. ’099 Patent, 21:25-38.

⁴⁸² Riddle, 5:53-57.

⁴⁸³ Riddle, 5:53-6:23, claim 8, Figs. 1A-1C, 3.

a plurality of network links upon which said traffic is carried; a network routing means; and,

a processor means operative to:

parse a packet into a first flow specification, wherein said first flow specification contains at least one instance of any one of the following:

a protocol family designation,

a direction of packet flow designation, a protocol type designation,

a pair of hosts, a pair of ports,

in HTTP protocol packets, a pointer to a MIME type; thereupon,

match the first flow specification of the parsing step to a plurality of classes represented by a plurality of said classification tree type nodes, each said classification tree type node having a traffic specification and a mask, according to the mask; thereupon,

if a matching classification tree type node was not found in the matching step, associating said first flow specification with one or more newly-created classification tree type nodes; thereupon, incorporating said newly-created classification tree type nodes into said plurality of said classification tree type nodes.⁴⁸⁴

381. Based on these teachings, a POSITA would have recognized that Riddle's processor is configured to carry out state operations specified in the state patterns/operations memory to achieve Riddle's goal of classifying traffic. Further, a

⁴⁸⁴ Riddle, claim 8.

POSITA would have understood that Riddle's processor is coupled to components like protocol/state identification engine, state patterns/operations memory, and the memory storing flow-entry lists.⁴⁸⁵ As I discuss above regarding '099 claim element 1.5, Riddle alone or in view of Ferdinand renders obvious storing flow-entry lists in a database.

382. Further, Riddle discloses applying policies associated with "leaf" nodes of the classification tree.⁴⁸⁶ As discussed above in Section IV.A.1, Riddle incorporates-by-reference the teachings of Packer as though fully set forth in Riddle. As shown below in annotated Figure 5F, Packer similarly applies policies associated with "leaf" nodes of the classification tree: "The processing steps of flowchart 511 determine a class and a policy for a flow[.]"⁴⁸⁷ As such, a POSITA would have understood that each of Riddle's traffic classes, e.g., FTP-server or World-Wide-Web traffic classes, can have its own respective policies.⁴⁸⁸ And a POSITA would have understood that Riddle's processor ultimately applies each relevant policy based on the state of the flow, as illustrated in Packer's Figure 5B⁴⁸⁹

⁴⁸⁵ Riddle, 5:53-6:23, claim 8, Figs. 1A-1C.

⁴⁸⁶ Riddle, 9:29-42, Packer, 18:3-5 ("The processing steps of flowchart 511 determine a class and a policy for a flow[.]", Fig. 5F.

⁴⁸⁷ Packer, 18:3-5, Fig. 5F.

⁴⁸⁸ Riddle, 10:36-39.

⁴⁸⁹ Riddle, 9:29-42, claim 3, Packer, Fig. 5F.

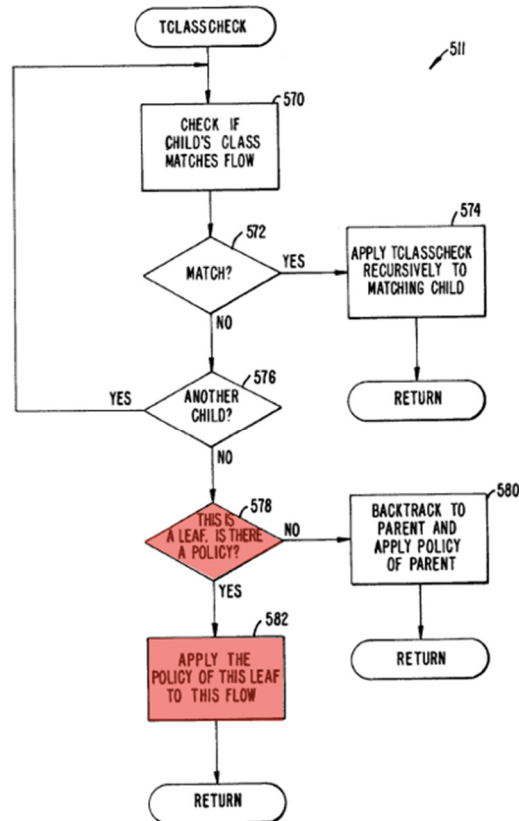


FIG. 5F.

383. To the extent Patentee asserts that the “parser subsystem,” “lookup engine,” “protocol/state identification mechanism” and “state processor” of claim elements 1.3, 1.6, 1.8 and 1.9 require separate pieces of hardware, it would have been obvious to a POSITA to implement Riddle’s processor and programming code to be separate hardware components. This is because using dedicated hardware for various functions, especially functions as common as parsing, data lookup, and protocol/state identification, would have been obvious to a POSITA because dedicated hardware allows for faster operation for faster networks, as described in the ’099 patent itself and/or Ferdinand:

Each of the individual hardware elements through which the data flows in the system are now described with reference to FIGS. 10 and 11. Note that while we are describing a particular hardware implementation of the invention embodiment of FIG. 3, *it would be clear to one skilled in the art that the flow of FIG. 3 may alternatively be implemented in software running on one or more general-purpose processors, or only partly implemented in hardware.* An implementation of the invention that can operate in software is shown in FIG. 14. The hardware embodiment (FIGS. 10 and 11) can operate at over a million packets per second, while the software system of FIG. 14 may be suitable for slower networks. *To one skilled in the art it would be clear that more and more of the system may be implemented in software as processors become faster.*⁴⁹⁰

384. Further, Ferdinand discloses its monitor can include separate hardware components for performing various functions, such real time parser (RTP) 32, database 36, boot/load 22, and memory transport module 34, event manager 38, and control module 42.⁴⁹¹ As provided below, Ferdinand's Figure 5 illustrates its monitor having separate hardware components.

⁴⁹⁰ '099 Patent, 21:25-38.

⁴⁹¹ Ferdinand, 19:5-13.

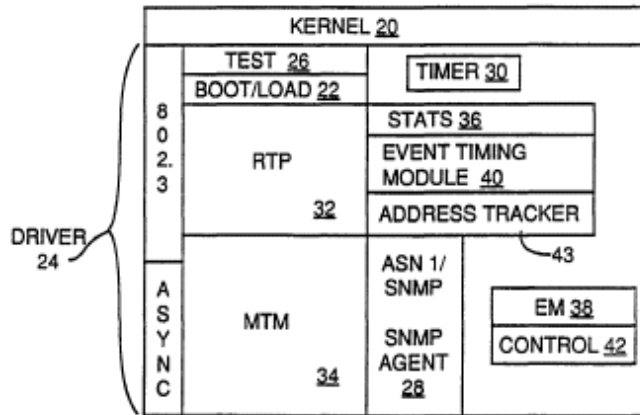


FIG 5

385. Desiring increased performance, a POSITA would have been motivated to utilize dedicated hardware components for parsing, lookups, protocol/state identification, and state processing/operations. On the other hand, a POSITA would have understood that Riddle’s use of a processor for these functions is less expensive and a more extensible solution than using dedicated hardware components.

- k. *’099 Claim Element 1.10: “the carrying out of the state operations furthering the process of identifying which application program is associated with the conversational flow-sequence of the packet, the state processor progressing through a series of states and state operations until there are no more state operations to perform for the accepted packet, in which case the state processor updates the flow-entry, or until a final state is reached that indicates that no more analysis of the flow is required, in which case the result of the analysis is announced”*

386. Riddle discloses this claim element. As discussed with respect to claim element 1.7, Riddle describes parsing a packet and carrying out state operations to

identify which application program is associated with the conversational flow-sequence of the packet. Riddle describes this process when identifying whether the packet belongs to a service aggregate class.⁴⁹²

387. Further, as discussed with respect to claim elements 1.7 and 1.9, Riddle teaches the state processor progresses through a series of states and state operations until there are no more state operations to perform for the accepted packet. For example, Riddle describes progressing through a traffic class tree which corresponds to progressing through a series of states and state operations. As incorporated-by-reference into Riddle, Packer's Figure 5F illustrates progress through a series of states and state operations.

388. Riddle also discloses that it will update the flow-entry when traffic matches a class. Riddle states:

FIG. 4A depicts a flowchart 401 of processing steps for automatically classifying traffic. In a step 402, a flow specification is parsed from the flow being classified. Then in a step 404, the flow specification parsed from the flow in step 402 is compared with the traffic specifications in each node of the classification tree. ***Rules are checked starting from most specific to least specific. In a decisional step 406, a determination is made if traffic matches one of the classes being classified. If this is so, then in a step 408, an entry is made in a list of identifying***

⁴⁹² Riddle, 4:18-23, 11:10-36, 13:45-62, Fig. 4B.

*characteristics, such as protocol type (SAP), IP protocol number, server port, traffic type if known, MIME type, a time of occurrence of the traffic.*⁴⁹³

And Riddle's flowchart details that the monitor will "parse flow specification from a packet of the flow" (step 402), "compare flow specification with existing classification tree" (step 404), determine if "traffic matches a class?" (step 406), and "enter into a saved list characteristics of the traffic" (step 408).⁴⁹⁴

389. If the monitor determines the parsed packet information does not match a traffic class, Riddle states that the packet will be given a default policy:

All traffic which does not match any user specified traffic class falls into an automatically created default traffic class which has a default policy. In FIG. 2A, the default category is depicted by a default node 205, and in FIG. 2B, the default category is depicted by a default node 225.⁴⁹⁵

390. As shown in Figures 2A and 2B, Riddle's "default" state indicates that no more analysis of the flow is required.

391. Upon completing the packet's examination, classification, and processing, Riddle reaches reaching a final state and then reports the results of this analysis and related metrics:

⁴⁹³ Riddle, 12:42-53.

⁴⁹⁴ Riddle, Fig. 4A.

⁴⁹⁵ Riddle, 10:52-56, Figs. 2A-2B.

In an optional step 413 (not show), after the processing of flowchart 401 completes or at periodic intervals or on demand, a list of traffic classes produced in steps 402 through 412 are displayed to a network manager. The list may be sorted by any well-known criteria such as: 1) most “hits” during a recent interval, 2) most recently-seen (most recent time first), 3) most data transferred (bytes/second) during some interval, or a moving average. The user may choose an interval length or display cutoff point (how many items, how recent, at least B bytes per second, or other thresholds). The Network manager may then take some action (e.g. pushing a button) to select the traffic types she wishes to add to the classification tree. The display can be hierarchical, as depicted in lines (3) below:

```
FTP496      (3)
  FTP-cmd
  FTP-data
to host1
  tcp
  FTP
    FTP-cmd
    FTP-data
  HTTP
    images
    java
    text
```

⁴⁹⁶ As shown in the '864 Provisional (Ex. 1024, 24), the first line “FTP” of Riddle’s exemplary classification tree should be directly above the subclassifications “FTP-cmd” and “FTP-data.”

port 9999

(3)

wherein the “port 9999” entry is an inference corresponding to an application checking for repeated or simultaneous connections made to a specific port.

In a related embodiment, a threshold for display or class creation of well-known traffic types is provided.⁴⁹⁷

392. As taught in the above passage, Riddle discloses displaying the received traffic sorted by metrics, such as the most “hits” (i.e., the count of duplicates), most recently seen, most data transferred, how many items, how recent, bytes per second, most recently used, number of bytes received during any interval, or other thresholds.

393. For the above reasons, it is my opinion that Riddle in view of Ferdinand renders obvious claim 1 of the '099 Patent.

3. Dependent '099 Claim 2

394. Riddle discloses all the limitations of this claim. Claim 2 depends from independent claim 1 and recites: “A packet monitor according to claim 1, wherein the flow-entry includes the state of the flow, such that the protocol/state identification mechanism determines the state of the packet from the flow-entry in the case that the lookup engine finds a flow-entry for the flow of the accepted packet.”

⁴⁹⁷ Riddle, 12:64-13:23, 14:1-5.

395. As discussed with respect to '099 claim element 1.5, Riddle describes a storage subsystem for storing a flow-entry database having a plurality of flow-entries for conversational flow encountered by the monitor. For example, and as shown below in Figure 3, Riddle discloses a plurality of saved lists 308 storing classified traffic packets pending incorporation into traffic tree 302.⁴⁹⁸

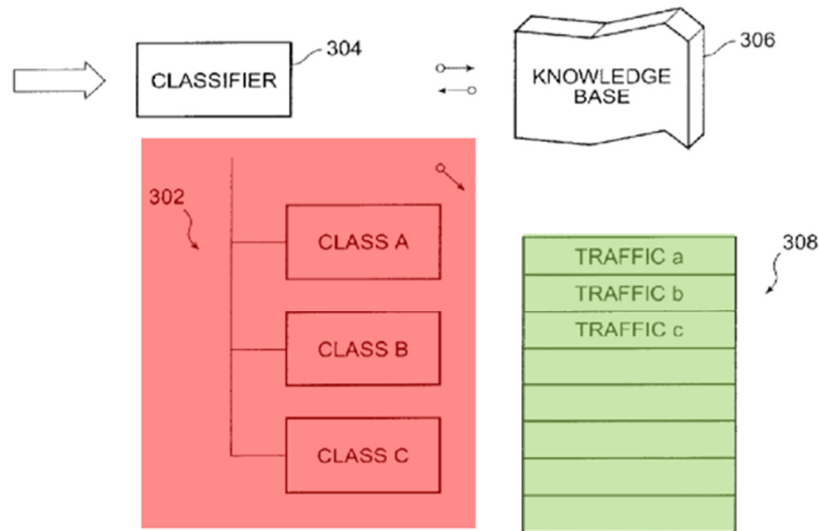


FIG. 3

396. As another example, Riddle teaches that its monitor parses flow packets to identify the packet's flow specification and stores the specification in the database's flow-entries which contain the state of the flow:

FIG. 4A depicts a flowchart 401 of processing steps for automatically classifying traffic. In a step 402, a flow specification is parsed from the flow being classified. Then in a step 404, the flow specification parsed from the flow in step 402 is compared with the traffic specifications in each node of

⁴⁹⁸ Riddle, 12:27-38, Fig. 3.

the classification tree. Rules are checked starting from most specific to least specific. In a decisional step 406, a determination is made if traffic matches one of the classes being classified. If this is so, then in a step 408, *an entry is made in a list of identifying characteristics, such as protocol type (SAP), IP protocol number, server port, traffic type if known, MIME type, a time of occurrence of the traffic.* In an optional step 410, duplicate instances having the same identifying characteristics are suppressed, in favor of keeping a count of the duplicates and a most recent time traffic with these identifying characteristics was encountered. In an optional step 412, a byte count of traffic of this type has been detected is included.⁴⁹⁹

Riddle illustrates the updating of flow-entries containing the state of the flow in Figure 4A's flowchart that details the monitor will "enter into a saved list characteristics of the traffic" (step 408) and "suppress duplicates" of packet characteristics (step 410). And as I discuss with respect to '099 claim element 1.7, Riddle teaches that its flow-entries include the flow's traffic-type, which are sequentially refined.

397. As detailed with respect to '099 claim element 1.8, the '099 Patent discloses that one determines the state of the flow by the relationship of packets and the entire conversational flow.⁵⁰⁰ And the Challenged Patents specify that the state of the flow includes "parameters such as the time, length of the conversational flow, data

⁴⁹⁹ Riddle, 12:42-59.

⁵⁰⁰ '099 Patent, 5:27-34.

rate, etc.”⁵⁰¹ Thus, a POSITA would have understood that Riddle’s monitor determines the state of the flow, at least, by determining metrics like the count of duplicates, the most recent time the monitor encountered a flow with the same identifying characteristics, and a byte count of the detected flow.⁵⁰² Riddle details that at least the “time of occurrence of the traffic” metrics are stored in the flow-entry.⁵⁰³ Consistent with that detail, a POSITA would have understood that Riddle’s other metrics also identify characteristics for storage in the flow-entry.

398. For the above reasons, it is my opinion that Riddle in view of Ferdinand renders obvious claim 2 of the ’099 Patent. And for all the reasons set forth above, it is my opinion that Riddle in view of Ferdinand renders obvious claims 1 and 2 of the ’099 Patent.

B. For the ’099 Patent, Riddle in View of Ferdinand and Further in View of Baker Renders Obvious Dependent Claims 4 and 5.

399. It is my opinion that a POSITA would have recognized that each and every limitation of the ’099 claims 4 and 5 is disclosed or rendered obvious in light of Riddle in view of Ferdinand and further in view of Baker. ’099 claims 4 and 5 depend from independent claim 1, which is obvious over Riddle in view of Ferdinand, as shown above in Section VII.A.

⁵⁰¹ E.g., ’099 Patent, 5:27-34.

⁵⁰² Riddle, 12:53-13:8, 14:1-5, Fig. 4A (steps 410, 412).

⁵⁰³ Riddle, 12:42-59.

1. Reasons to Modify the Combination of Riddle and Ferdinand and Further in View of Baker

400. Riddle and Baker are in the same field of endeavor and contain overlapping disclosures with similar purposes.

401. Riddle discloses classifying packets in a data communications network into traffic classes based upon protocols such as FTP, HTTP, TCP, UDP, and/or Ethernet.⁵⁰⁴ In doing so, Riddle already contains a description of each protocol. One of the expressly stated advantages of Riddle's monitor "is that network managers need not know the technical aspects of each kind of traffic in order to configure traffic classes."⁵⁰⁵ Given that network managers using Riddle's monitor do not need to know the technical aspects of each kind of traffic, a POSITA would have appreciated that it would be desirable to include a mechanism to update the protocols that Riddle's monitor can classify.

402. Baker teaches parsing, filtering, generating and analyzing data (or frames of data) transmitted over a data communications network based on one or more programmably configurable protocol descriptions which may be stored and retrieved from an associated memory.⁵⁰⁶ Baker discloses storing protocol descriptions as

⁵⁰⁴ Riddle, Fig. 1D, Table 2.

⁵⁰⁵ Riddle, 15:37-40.

⁵⁰⁶ Baker, 3:32-4:6.

“protocol description files (PDF).”⁵⁰⁷

403. As such, before the time of the purported invention, a POSITA would have been motivated and found it obvious to make the packet flow classifier of Riddle more configurable based on a POSITA’s general knowledge and Baker’s teachings.

404. In particular, it would have been obvious to a POSITA to modify Riddle’s teaching of a traffic classification system with Baker’s teaching of PDFs so that Riddle’s processor receives commands in a high-level protocol description language describing protocols and translates those commands into parsing/extraction operations because a POSITA understood the advantageousness for Riddle’s classification system to allow for updating protocol descriptions.⁵⁰⁸ This would be highly desirable in Riddle’s monitor because Riddle envisions that the network managers using the monitor do not know the technical details of the protocols.⁵⁰⁹

405. Moreover, Baker provides an explicit motivation to combine: “[I]t would be highly desirable to be able to parse all protocols with a single configurable software (or hardware) module so that support for additional protocols could be added

⁵⁰⁷ Baker, 19:6-10 (“In the presently preferred embodiment, each of these protocol description records with its associated field, statistics, lookup, and filter record information is also written to a protocol specific protocol description file (PDF).”).

⁵⁰⁸ Baker, 19:6-10.

⁵⁰⁹ Riddle, 15:28-31.

to a system without requiring substantial modification to the system or its control logic.”⁵¹⁰ A POSITA would have realized that combining Baker with Riddle provides the advantage of making Riddle’s monitor more configurable without requiring substantial modification to the monitor itself.

406. Additionally, Baker states that its invention may be used with network monitors such as those disclosed in Riddle. Baker’s invention “may be employed in any system where it is useful to be able to examine and perform various operations on contiguous bit-fields in data structures, wherein each data structure is composed of predefined fields of one or more contiguous bits.”⁵¹¹ Baker also discloses that the invention “may be incorporated in a network device, such as a network analyzer, bridge, router, or traffic generator, including a CPU and a plurality of input devices, storage devices, and output devices.”⁵¹² These are the same types of devices as Riddle’s monitor.⁵¹³ Given Baker’s disclosures, this combination is nothing more than a simple substitution of one known element for another used in their ordinary and predictable manner to update Riddle’s protocols.

⁵¹⁰ Baker, 3:3-8.

⁵¹¹ Baker, 6:18-22.

⁵¹² Baker, 4:27-30

⁵¹³ Riddle, 5:55-57, claim 8, Figs. 1A, 1B, 1C.

407. Further, Riddle discloses traffic classification based upon the Ethernet protocol and Baker discloses PDFs for the Ethernet protocol.⁵¹⁴ And Baker describes PDFs for Generic Protocols which could be other protocols used in Riddle.⁵¹⁵ Thus, a POSITA would have been motivated to modify the teachings of Riddle to increase the ability to handle updated PDFs as taught by Baker.⁵¹⁶ This analysis applies to each of the '099 claim elements described below that rely on Riddle in view of Baker.⁵¹⁷

2. Dependent '099 Claim 4

408. Riddle in view of Ferdinand and further in view of Baker renders obvious this claim. Claim 4 depends from independent claim 1 and recites:

A packet monitor according to claim 1, further comprising: a compiler processor coupled to the parsing/extraction operations memory, the compiler processor configured to run a compilation process that includes:

receiving commands in a high-level protocol description language that describe the protocols that may be used in packets encountered by the monitor, and

translating the protocol description language commands into a plurality of parsing/extraction operations that are initialized into the parsing/extraction operations memory.

⁵¹⁴ Riddle, Fig 1D, 7:47-53; Baker, Table 12, 21:32-22:1.

⁵¹⁵ Baker, Table 13.

⁵¹⁶ Baker, Table 13.

⁵¹⁷ Baker, Table 13.

409. As I discuss with respect to '099 claim elements 1.2 and 1.3, Riddle teaches a processor (e.g., processor 30) coupled to the parsing/extraction operations memory that identifies information relating to protocols used in each packet.⁵¹⁸ Given that Riddle's monitor extracts packet portions to identify packet-protocol data, a POSITA would have understood that Riddle's processor uses a protocol description language to initialize parsing/extraction operations. Further, Riddle discloses that the processor operates to identify a parsed flow's "protocol family designation" and "protocol type designation."⁵¹⁹ And Riddle states that its packet classifier "is implemented in the C programming language," which is a high-level language.⁵²⁰ In doing so, Riddle discloses "apply individual instances of traffic classification paradigms to packet network flows based on selectable information obtained from a plurality of layers of a multi-layered communication protocol in order to define a characteristic class, then mapping the flow to the defined traffic class."⁵²¹ Based on Riddle's teachings, a POSITA would have understood that Riddle selects this protocol-specific information based on high-level protocol description language.

410. As I further discuss with respect to '099 claim elements 1.2 and 1.3, Riddle

⁵¹⁸ Riddle, 6:1-15, claim 8, Fig. 1A.

⁵¹⁹ Riddle, 12:50-53, claim 8.

⁵²⁰ Riddle, 5:53-57.

⁵²¹ Riddle, 4:10-15, 4:56-65, 5:53-57, 9:13-19.

discloses generating classification trees containing parsing/extraction operations that are stored in memory. For example, Riddle states that the “function of the classifier 304 is controlled by a command language interface.”⁵²² Moreover, Riddle discloses translating the high level “syntax of the traffic specifications” into a plurality of parsing/extraction operations that are initialized into the parsing/extraction operations memory.⁵²³

411. Further, Riddle teaches that a network manager, based on high-level traffic descriptions, configures the network flow classifier via a user interface.⁵²⁴ Riddle discloses that the manager takes actions on packets and flows encountered by the monitor based on protocol-type by performing high-level actions.⁵²⁵ Examples of these actions include pushing a button on the interface presented to the manager, or using a command language interface.⁵²⁶ Riddle explains that one advantage of its monitor “is that network managers need not know the technical aspects of each kind of traffic in order to configure traffic classes.”⁵²⁷ Given that managers using Riddle’s monitor do not need to know the technical aspects of each kind of traffic,

⁵²² Riddle, 14:28-29.

⁵²³ Riddle, 14:67-15:31.

⁵²⁴ Riddle, 12:65-13:10.

⁵²⁵ Riddle, 12:65-13:10, 14:27-66.

⁵²⁶ Riddle, 12:65-13:10, 14:27-66.

⁵²⁷ Riddle, 15:37-40.

a POSITA would have appreciated the desirability to include a mechanism to update the protocols that Riddle's monitor can classify.

412. Baker discloses running a compilation process for high-level, human-readable protocol description files (PDFs) and initializing state transition patterns/operations relating to the PDFs:

[T]he initialization of the system includes a determination of the presence of PDF files and the extraction of the protocol and associated control record information from all of the PDF files found. The number of PDF files is determined, and a ProtocolList is constructed consisting of a sorted vector of protocol records at least the size of the number of PDF files found. The name of each protocol record found is inserted in the ProtocolList. The PDF files are then read to memory in the sequence described above for the PDF file writing. The lookup records that indicate a next protocol are associated with the appropriate entries in the ProtocolList.⁵²⁸

413. Baker discloses examples of protocol descriptions, such as Tables 12 and 13 which are provided below. Table 12 provides protocol descriptions for the Ethernet, and Table 13 provides protocol descriptions for Baker's "Generic Protocol."⁵²⁹

⁵²⁸ Baker, 17:7-18, 2:63-3:7, 11:26-12:6.

⁵²⁹ Baker, 21:32-22:1, Tables 12-13.

TABLE 12
ETHERNET v2.0 PROTOCOL SPECIFICATION

0	15	23	47
Destination Hardware Address			
Source Hardware Address			
Ethernet Protocol Type			

Destination Hardware Address - destination hardware station address (48 bits)

Source Hardware Address - source hardware station address (48 bits)

Ethernet Protocol Type - upper layer protocol designator (16 bits)
0x8888=GP

TABLE 13
GENERIC PROTOCOL (GP) SPECIFICATION

0	7	15	23	31
Version No.	HeaderLen	Frame Type	Frame Length	
Checksum		Control	Hop Count	
Src Socket		Dst Socket		
Src Address				
Dst Address				

Version Number (4 bits) - software version number
HeaderLen (4 bits) - length of GP header in 32 bit words.
0- 4 = illegal
5 = No Optional fields,
6-15 = 32-320 bits of options

Frame Type (8 bits) - upper level protocol identifier
0 = Unknown
1 = GP1
2 = GP2
3-255 = Unknown

Frame Length (16 bits) - frame length in bytes including header
Checksum (16 bits) - Checksum of header including options
Control (8 bits) - reserved (must be zero)

414. Further, Baker discloses a software code example in the high-level programming language of C++. ⁵³⁰ This code example describes extracting the control information from the PDF files, creating a protocol definition in memory with required data structures, and inserting records into the ProtocolList, as shown in “PCOL.CPP.”⁵³¹

415. It is my understanding that the '099 patent incorporates-by-reference disclosures of the '725 patent. ⁵³² I note that the '725 patent describes a “protocol description language” as follows:

⁵³⁰ Baker, 128:20-132:24.

⁵³¹ Baker, 128:20-132:24.

⁵³² '099 Patent, 1:15-20.

Input to the compiler includes a set of files that describe each of the protocols that can occur. These files are in a convenient protocol description language (PDL) which is a high level language. PDL is used for specifying new protocols and new levels, including new applications. The PDL is independent of the different types of packets and protocols that may be used in the computer network. A set of PDL files is used to describe what information is relevant to packets and packets that need to be decoded. The PDL is further used to specify state analysis operations. Thus, the parser subsystem and the analyzer subsystems can adapt and be adapted to a variety of different kinds of headers, layers, and components and need to be extracted or evaluated, for example, in order to build up a unique signature.⁵³³

The '725 Patent additionally states:

The protocol description language (PDL) files 336 describes both patterns and states of all protocols that an occur at any layer, including how to interpret header information, how to determine from the packet header information the protocols at the next layer, and what information to extract for the purpose of identifying a flow, and ultimately, applications and services.... This information is input into compiler and optimizer 310.⁵³⁴

416. The '725 Patent discloses examples of PDL files that include commands for a particular protocol.⁵³⁵ Similarly, Baker discloses that each PDF file describes

⁵³³ '725 Patent, 41:24-37.

⁵³⁴ '725 Patent, 9:29-40; Fig. 4.

⁵³⁵ '725 Patent, 45:1-94:67 (including PDL files for several protocols).

commands for a particular protocol.⁵³⁶ As the following examples illustrate, Baker's disclosures regarding its PDF files teach the '099 claim 4's high-level protocol description language.

417. First, according to the '725 Patent, an exemplary command in a high-level protocol description language is a "HEADER" attribute that "describe[s] the length of the protocol header."⁵³⁷ Baker similarly discloses a "numBits" attribute that describes "the total bit length of the protocol header."⁵³⁸

418. Second, according to the '725 Patent, another exemplary command in a high-level protocol description language is a "PROTOCOL" definition used to "define the order of the FIELDS and GROUPs within the protocol header."⁵³⁹ Similarly, Baker discloses a "fields" attribute that references the associated "field records that describe the protocol header" where each field record includes, for example, a "fblen" attribute describing "the length of the field in bits" and a "fdwoff" attribute describing "the byte offset from the start of protocol header," among other

⁵³⁶ Baker, 11:22-25 ("[E]ach of these protocol description records with its associated field, statistics, lookup, and filter record information is also written to a protocol specific protocol description file (PDF).").

⁵³⁷ '725 Patent, 48:41-50, and col. 73 ("HEADER { LENGTH=14 }").

⁵³⁸ Baker, Table 1 ("numBits" attribute), 11:32, 57:1 (fread(&num_bits, sizeof(num_bits), 1, fp); // Read fixed header length in bits).

⁵³⁹ '725 Patent, 47:34-48:18, col. 79 (PROTOCOL section provides "Detailed packet layout for the IP datagram. This includes all fields and format. All offsets are relative to the beginning of the header.").

attributes.⁵⁴⁰

419. Third, according to the '725 Patent, a further exemplary command in a high-level protocol description language is a "CHILDREN" attribute "used to describe how children protocols are determined."⁵⁴¹ Similarly, Baker discloses a "ptr2np" attribute of each field record that includes a "pointer to lookup structure/class . . . next protocol definition to use (0 = none)" and the "next protocol lookup records" are described with reference to Table 4 as including a "Protocol" attribute describing the "pointer to protocol description record," among other attributes.⁵⁴²

420. As such, a POSITA would have understood that Baker's PDFs are protocol descriptions provided in a protocol description language. This is because, upon initialization, Baker teaches that the system is able to "extract[] the protocol and associated control record information" from the file, construct a ProtocolList, and read the PDF file into memory in the sequence described at col. 11:26-12:6. Thus, Baker's PDFs include commands in a high-level language that describe protocols that may be encountered by the monitor and, for example, how to interpret header information and how to determine from the packet header information the protocol

⁵⁴⁰ Baker, Table 1 ("fields" attribute), Table 2 ("fblen" and "fdwoff" attributes), 11:36-40, 11:49-50, 147:35-36.

⁵⁴¹ '725 Patent, 49:45-55, col. 79 ("CHILDREN { DESTINATION=Protocol }").

⁵⁴² Baker, Table 2 ("ptr2np" attribute), Table 4 ("Protocol" attribute), 8:13-15, 148:10-12, 165:37-166:5, 177:4-26.

at the next layer. Baker thus teaches protocol description files that are provided in a protocol description language, and are written in a protocol description language as described in by the '725 Patent at col. 41:24-37.

421. Thus, a POSITA would have found it obvious in view of Baker's teachings to modify Riddle's processor with compilation processes that (a) receive commands in a high-level protocol description language describing protocols and (b) translate those commands into parsing/extraction operations. A POSITA would have been motivated to do so because such compilation processes allow for easily updating protocol information. Further, Baker explains the benefit of using such PDFs as "it would be highly desirable to be able to parse all protocols with a single configurable software (or hardware) module so that support for additional protocols could be added to a system without requiring substantial modification to the system or its control logic."⁵⁴³ Based on Baker's teachings, a POSITA would have understood that modifying Riddle's monitor provides the advantage of making the monitor more configurable without requiring substantial modification to the monitor itself.

3. Dependent '099 Claim 5

422. Riddle in view of Ferdinand and further in view of Baker renders obvious this claim. Claim 5 depends from dependent claim 4 and recites:

⁵⁴³ Baker, 3:3-8.

A packet monitor according to claim 4, wherein the protocol description language commands also describe a correspondence between a set of one or more application programs and the state transition patterns/operations that occur as a result of particular conversational flow-sequences associated with an application program, wherein the compiler processor is also coupled to the state patterns/operations memory, and wherein the compilation process further includes translating the protocol description language commands into a plurality of state patterns and state operations that are initialized into the state patterns/operations memory.

423. As I describe with respect to '099 claim elements 1.7 to 1.10, Riddle teaches a correspondence between a set of one or more application programs, such as FTP and PointCast programs, and the state transition patterns/operations that occur as a result of processing a particular conversational flow-sequence through the process of generating classification tree (e.g., Figures 2A-2B) containing parsing/extraction operations that are used to classify a conversational flow.⁵⁴⁴

424. Further, as I detail with respect to '099 claim 4, Baker discloses receiving commands in a high-level protocol description language that describe the protocols that may be used in packets encountered by the monitor. And Baker teaches translating the protocol description language commands into a plurality of state patterns/operations. For example, and as shown below, Baker discloses that Figure 11

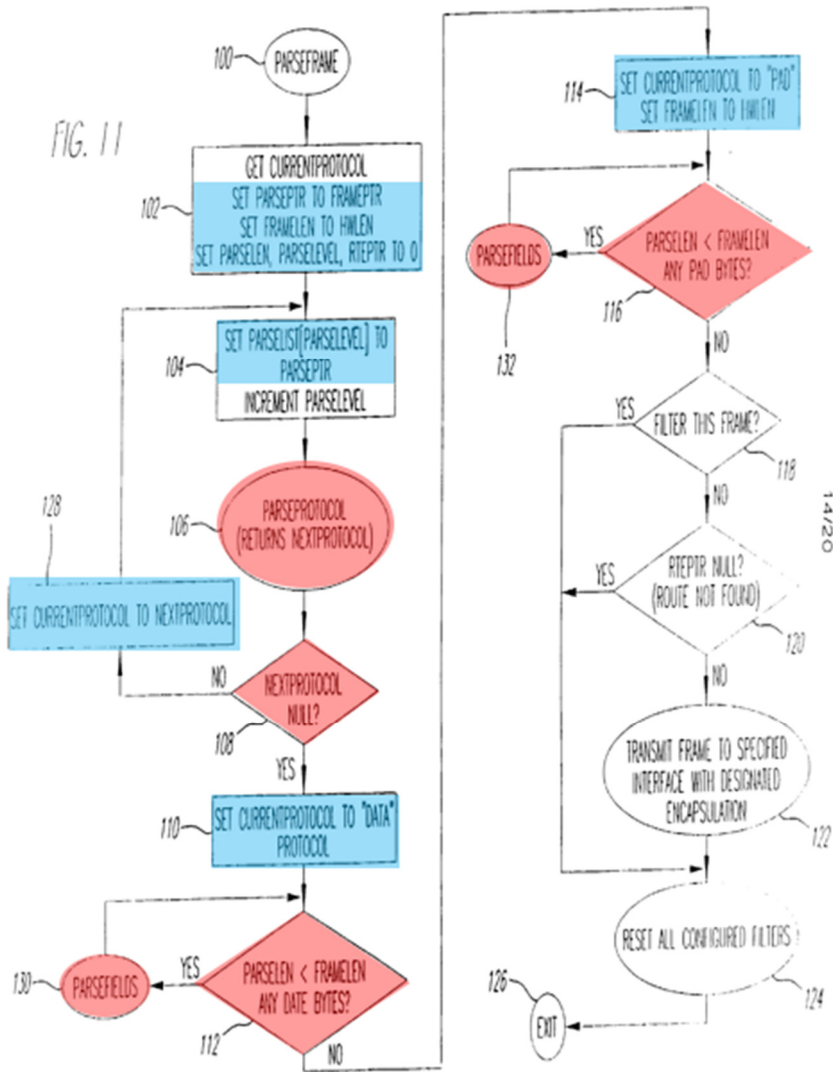
⁵⁴⁴ E.g., Riddle, 9:20-63, 10:19-56, 13:11-22, Figs. 2A, 2B, 3.

describes frame parsing control logic (i.e., how successive protocol headers are parsed), and parsing the remaining information as application data and frame pad.⁵⁴⁵ Figure 11 also shows setting states (blue boxes) followed by specific operations (red boxes):

The ParseFrame control logic systematically parses through each network frame (at 104 to 108 and 128) until all known protocol headers have been parsed. Any remaining frame bits are parsed as application data (at 110, 112 and 130) and/or pad data (at 114, 116 and 132).⁵⁴⁶

⁵⁴⁵ Baker, 26:26-32, Fig. 11.

⁵⁴⁶ Baker, 36:28-36, 37:17-38:24, Fig. 11;.



425. As shown below, Baker teaches that Figure 12 shows protocol-specific option operations.⁵⁴⁷ For example, Figure 12 includes step 160 inquiring whether the CurrentProtocol supports optional fields, and if so, those frames are parsed using the ParseFields control logic at step 164.⁵⁴⁸

⁵⁴⁷ Baker, 37:1-16, 38:25-39:19, Fig. 12.

⁵⁴⁸ Baker, 38:25-33.

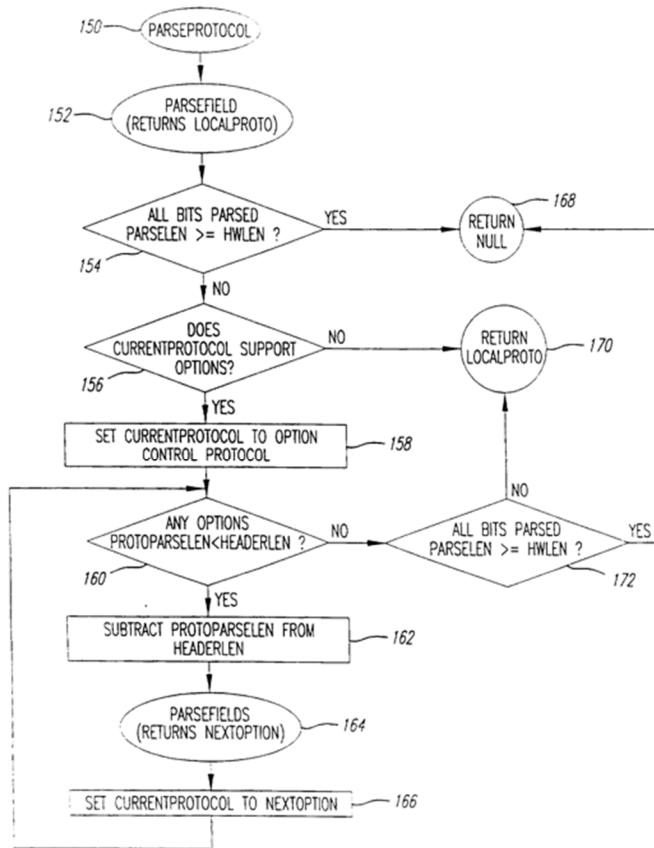


FIG. 12

426. Thus, a POSTIA would have found it obvious in view of Baker’s teachings to use flexible parsing operations with Riddle’s “service aggregates” and URI searches.⁵⁴⁹ Because it would allow Riddle’s classification system to easily update protocol descriptions, a POSITA would have been motivated to place Riddle’s aggregation methods into a configurable system as taught by Baker. A POSITA would have understood that such a modification of Riddle results in a compiler processor, as discussed in claim 4, coupled to the state patterns/operations

⁵⁴⁹ Baker, 3:3-8.

memory.

C. For the '099 Patent, Riddle in View of Ferdinand and Further in View of Yu Renders Obvious Claims 1 and 2.

427. It is my opinion that a POSITA would have recognized that each and every limitation of the '099 claims 1 and 2 is disclosed or rendered obvious in light of Riddle in view of Ferdinand and further in view of Yu. Specifically, my opinions regarding Riddle in view of Ferdinand are exactly the same as those above in Section VII.A, but further include the teachings of Yu. Thus, as if fully set forth here, I incorporate the discussion from Section VII.A regarding the obviousness of '099 Claims 1 and 2 over Riddle in view of Ferdinand.

428. As discussed above, all of the Challenged Claims require “conversational flows” or a “conversational-flow sequence.” For example, '099 claim element 1.7 recites “the particular conversational-flow sequence is associated with the operation of a particular application program.” While Riddle itself teaches identifying conversational flows, Yu further demonstrates identifying conversational flows through its “flow classification.”⁵⁵⁰ Further, Yu teaches flow classification logic that “keeps track of the flow’s state until matching criteria is met” when identifying whether a packet belongs to a conversational flow.⁵⁵¹

⁵⁵⁰ Yu, 1:56-60, 3:32-36, 3:47-49; 4:1-8.

⁵⁵¹ Yu, 4:62-64.

429. Yu defines a “stream” which is analogous to the ’099 Patent’s “connection flow.” Specifically, Yu defines a “stream” as “packets that have the same source and destination address, source and destination port, and protocol type.”⁵⁵² Further, Yu defines a “flow” as “all packets that match the same flow classification specification” and specifies that “a flow may include *one or more* streams.”⁵⁵³

430. Yu teaches that its flow classification specification provides the screening criteria for the flow classifier logic to sort network traffic into “flows” (which may include multiple streams, i.e., connection flows), such as defining a specific pair of hosts running a specific application.⁵⁵⁴ And Yu details that “the matching criteria used by a flow classifier to classify a flow may include a specific value, a range, or wildcard on interface port numbers, protocols, IP addresses, TCP ports, *applications, application data*, or any user specifiable criteria.”⁵⁵⁵ As such, Yu teaches identifying the ’099 Patent’s “conversational flow” by Yu’s flow classifier linking multiple “streams” into a “flow” based on application or application data.

431. Moreover, Yu describes state tracking:

[M]ultiple packets may be required for more sophisticated flow classification (stateful packet inspection), since the policy decisions (action

⁵⁵² Yu, 4:1-4.

⁵⁵³ Yu, 3:47-49; 4:7-8.

⁵⁵⁴ Yu, 3:32-36.

⁵⁵⁵ Yu, 1:56-60.

specifications) may come from *different applications* which may have implemented different flow classifiers. In those cases, *the application's flow classification logic keeps track of the flow's state until a matching criteria is met.*⁵⁵⁶

432. During this “learning” process, Yu teaches that the required policies may be bound to each stream of a “flow” so that actions can be taken on future packets without intervention from the “host” application.⁵⁵⁷ Upon creating a hash value from well-known fields, Yu specifies using this hash value to find corresponding policies to reduce further complicated pattern-matching.⁵⁵⁸

1. Reasons to Modify the Combination of Riddle and Ferdinand Further in View of Yu

433. Riddle, Ferdinand, and Yu are in the same field of endeavor and contain overlapping disclosures with similar purposes.

434. As I detail above, Riddle discloses a monitor that examines and classifies packets in a flow.⁵⁵⁹ Similarly, Yu teaches examining packets in a flow.⁵⁶⁰ Both

⁵⁵⁶ Yu, 4:57-64.

⁵⁵⁷ Yu, 4:67-5:13.

⁵⁵⁸ Yu, 4:23-29.

⁵⁵⁹ E.g., Riddle, Abstract, 12:26-13:62, claim 8.

⁵⁶⁰ Yu, 1:22-26 (“policy-based application examines every packet coming in from the network”); Riddle, 12:43-44 (“a flow specification is parsed from the flow being classified”).

references teach determining an application relating to the packet flow.⁵⁶¹ And both references disclose applying a policy to a flow of identically classified packets.⁵⁶² Based on these similarities and teachings, a POSITA would have looked to Yu for ways to implement Riddle's monitor.

435. Yu teaches that "multiple packets may be required for more sophisticated flow classification (stateful packet inspection)."⁵⁶³ And Yu discloses that "[f]low classification and analysis technique is more than just looking into the packet's address, port number and protocol type and or other header information. It often involves state tracking for newer applications."⁵⁶⁴ Thus, a POSITA would have been motivated to use multiple packets to classify a flow, as Yu teaches, to manage application policies to classify flows.⁵⁶⁵

2. Riddle in View of Ferdinand and Further in View of Yu Renders Obvious Identifying a "Conversational Flow-Sequence" and the Claimed State Tracking

436. To the extent Riddle does not disclose or render obvious identifying a "conversational flow sequence" or the claimed state tracking, the combination of Riddle

⁵⁶¹ Yu, 3:34-36 ("flow classification specification 203a can be...running a specific application"); Riddle, 8:64-66 (Traffic classes may also be defined "[a]t the application level").

⁵⁶² Yu, 4:1-9 ("[a]ll packets belonging to the same stream are to be regulated by the same policy"); Riddle, 13: 63-64 ("policy is determined for the traffic class").

⁵⁶³ Yu, 4:57-62.

⁵⁶⁴ Yu, 1:63-67.

⁵⁶⁵ Yu, 2:45-50, 5:1, 6:19-21.

and Ferdinand, and further in view of Yu renders obvious these claim elements.

437. As I discuss above, a POSITA would have understood that Riddle teaches storing its state-based logic on storage subsystem 35 along with the other computer programs, code, and instructions.⁵⁶⁶ Thus, in accordance with Yu's teachings, implementing Riddle's packet classifier to use hardware acceleration coupled with software logic for stateful packet inspection would not have changed Riddle's principle of operation. As Riddle already describes processing steps in Figures 4A-4B, using Yu's state-traffic logic in Riddle's monitor amounts to nothing more than combining known prior art technologies used in their ordinary and predictable manner to examine and classify flows in Riddle's monitor.

438. Further, a POSITA would have understood that the combination of Riddle and Ferdinand and further in view of Yu results in Yu's state-tracking logic examining flow specifications parsed by Riddle for incoming packets to thereby track the flow's state. And a POSITA would have appreciated that this straightforward combination would not adversely impact Riddle's existing processing logic.

439. For all the above reasons, a POSITA would have been motivated to combine the teachings of Riddle, Ferdinand, and Yu to render obvious identifying that an accepted packet is part of a "conversational flow-sequence" as well as carrying out state operations with the state processor progressing through a series of states and

⁵⁶⁶ Riddle, 6:5-8.

state operations.

440. As set forth in my analysis of the '099 Patent in Sections VII.A.2 and VII.A.3 above, Riddle and Ferdinand disclose or render obvious all the remaining elements of '099 claims 1 and 2. As such, it is also my opinion that Riddle in view of Ferdinand and further in view of Yu renders obvious claims 1 and 2 of the '099 Patent.

D. For the '099 Patent, Riddle in View of Ferdinand and Baker and Further in View of Yu Renders Obvious Dependent Claims 4 and 5.

441. As I discuss above in Section VII.C, Riddle in view of Ferdinand and further in view of Yu renders obvious independent '099 claim 1. Further, as I discuss above in Section VII.B, Riddle in view of Ferdinand and further in view of Baker render obvious '099 claims 4 and 5. Because claims 4 and 5 depend from independent claim 1, it is also my opinion that a POSITA would have recognized that each and every limitation of the '099 claims 4 and 5 is disclosed or rendered obvious in light of Riddle in view of Ferdinand and Baker and further in view of Yu. Specifically, my opinions regarding Riddle in view of Ferdinand and Baker are exactly the same as those above in Section VII.B, but further include the teachings of Yu. Thus, as if fully set forth here, I incorporate the discussion from Section VII.B, regarding the obviousness of '099 Claims 4 and 5 over Riddle in view of Ferdinand and further in view of Baker.

442. As discussed above, all Challenged Claims require “conversational flows” or a “conversational-flow sequence.” For example, ’099 claim element 1.7 recites “the particular conversational-flow sequence is associated with the operation of a particular application program.” While Riddle itself teaches identifying conversational flows, Yu further demonstrates identifying conversational flows through its “flow classification.”⁵⁶⁷ Further, Yu teaches flow classification logic that “keeps track of the flow’s state until matching criteria is met” when identifying whether a packet belongs to a conversational flow.⁵⁶⁸

443. For all the reasons set forth in VII.C, a POSITA would have been motivated to combine the teachings of Riddle, Ferdinand, Baker, and Yu to render obvious identifying that an accepted packet is part of a “conversational flow-sequence” as well as carrying out state operations with the state processor progressing through a series of states and state operations as recited in claim 1.

444. As set forth in my analysis in above Sections VII.B.2 and **Error! Reference source not found.** above, Riddle, Ferdinand, and Baker disclose or render obvious all the remaining elements of ’099 claims 4 and 5. As such, it is also my opinion that Riddle in view of Ferdinand and Baker and further in view of Yu renders obvious claims 4 and 5 of the ’099 Patent.

⁵⁶⁷ Yu, 1:56-60, 3:32-36, 3:47-49; 4:1-8.

⁵⁶⁸ Yu, 4:62-64.

E. For the '099 Patent, Riddle in View of Ferdinand and Further in View of RFC1945 Renders Obvious Claims 1 and 2.

445. It is my opinion that a POSITA would have recognized that each and every limitation of the '099 claims 1 and 2 is disclosed or rendered obvious in light of Riddle in view of Ferdinand and further in view of RFC1945. Specifically, my opinions regarding the combination of Riddle and Ferdinand are exactly the same as those above in Section VII.A, but further include the teachings of RFC1945. Thus, as if fully set forth here, I incorporate the discussion from Section VII.A regarding the obviousness of '099 Claims 1 and 2 over Riddle in view of Ferdinand. As discussed above, all of the Challenged Claims require “conversational flows” or a “conversational-flow sequence.” For example, '099 claim element 1.7 recites “the particular conversational-flow sequence is associated with the operation of a particular application program.” While Riddle itself teaches identifying conversational flows, RFC1945 further demonstrates identifying conversational flows through the additional example of the use of HTTP header fields.

446. As I detail below, RFC1945 shows that HTTP header fields (such as host, URL, User Agent, User Agent Profile) and HTTP Referrer⁵⁶⁹ fields were known in the art and being used to relate traffic flows. For example, RFC1945 describes the “refer[r]er” request header field as:⁵⁷⁰

⁵⁶⁹ It is common in the art for this term to be misspelled as “Referer.”

⁵⁷⁰ RFC1945, 44-45

10.13 Referer

The Referer request-header field allows the client to specify, for the server's benefit, the address (URI) of the resource from which the Request-URI was obtained. This allows a server to generate lists

of back-links to resources for interest, logging, optimized caching, etc. It also allows obsolete or mistyped links to be traced for maintenance. The Referer field must not be sent if the Request-URI was obtained from a source that does not have its own URI, such as input from the user keyboard.

```
Referer      = "Referer" ":" ( absoluteURI | relativeURI )
```

Example:

```
Referer: http://www.w3.org/hypertext/DataSources/Overview.html
```

If a partial URI is given, it should be interpreted relative to the Request-URI. The URI must not include a fragment.

Note: Because the source of a link may be private information or may reveal an otherwise private information source, it is strongly recommended that the user be able to select whether or not the Referer field is sent. For example, a browser client could have a toggle switch for browsing openly/anonymously, which would respectively enable/disable the sending of Referer and From information.

447. The HTTP Referrer field identifies the address of the webpage that is linked to a resource being requested. By checking the HTTP Referrer field, a POSITA would have been able to determine where the HTTP request originated.⁵⁷¹

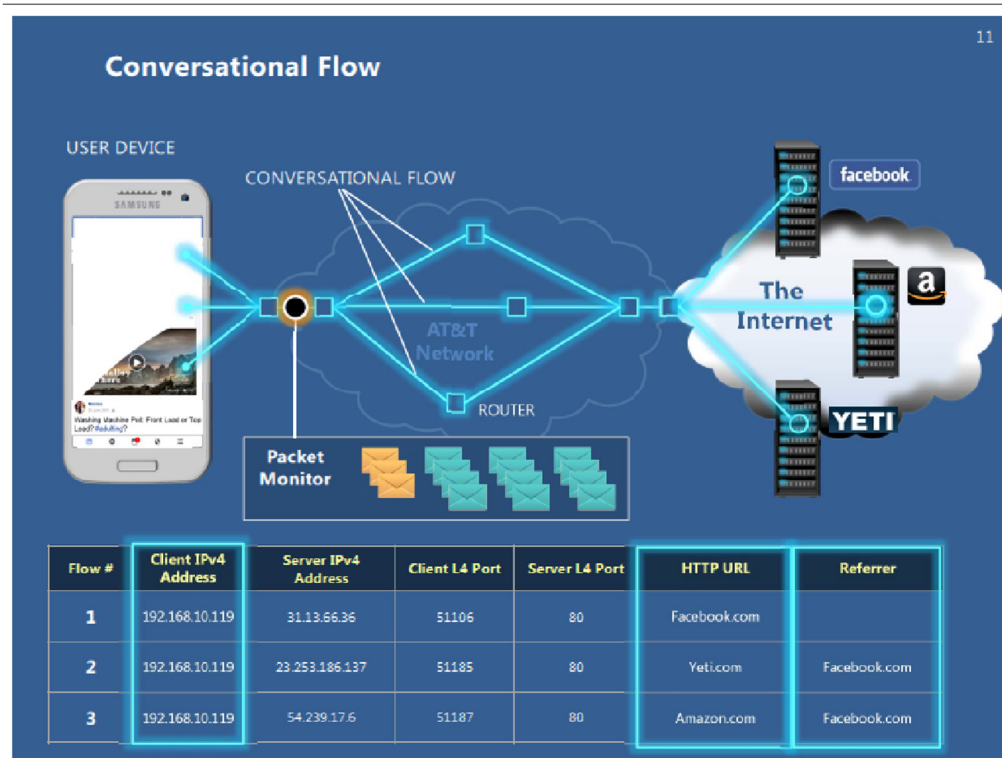
448. In prior district court litigations, Patentee has acknowledged that it was known in the art that HTTP fields can be used to identify a conversational flow.

For example, Patentee's technical expert, Dr. Almeroth, used the below slide to assert that an accused infringer created a "conversational flow" through the use of the HTTP Referrer field.⁵⁷² This slide shows the HTTP Referrer field as the only field

⁵⁷¹ RFC1945, 37-46.

⁵⁷² Ex. 1076 (Dr. Almeroth's Demonstrative Slide in *NetScout* district court case), 5; Ex. 1075 (S. Abdullah Declaration in *NetScout* district court case), ¶3.

that links together the separate flows shown in blue.



449. Moreover, Dr. Almeroth, testified:

So here's the -- the next step of the animation, please. Multiple requests. There's one response. That fills in part of it. And then the next request and response. And so now you have three separate flows.

They're separate flows. Each flow has information about that connection. *But one of the things that you see in these second and third connections -- or flows, is what's called a referrer. In this case, what's being described is information in a connection in a flow record that can be -- then be used to associate that flow record with another flow.*

So in this case, because the user device made a request to the Facebook server, that caused subsequent requests to be made, and those subsequent requests were at the behest of the first one, then the referrer is

Facebook.

So if you go to the next step, *what I've highlighted here is how the information in a particular flow record can be used to associate those flow records together to create what's called a conversational flow.*

So the idea is the flow-entries can be separate, they can contain information about that flow, but that information can then be correlated with information from other flows so that the analyzer understands that those are related flows.⁵⁷³

450. Dr. Almeroth further testified that the HTTP Referrer header field may satisfy the requirements of a “conversational flow” by correlating connection flows.

For example, Dr. Almeroth testified:

I had presented evidence about the HTTP referrer and this other information that's also part of the flow record. And that was what I used in my example animation to show how information in the NetScout G10 and GeoBlade, the two infringing products, used that information to tie one flow-entry back to another flow-entry. And because that information was in the flow record, it demonstrates the capability for that flow-entry to be correlated with other flow-entries. It says it has these fields in it, and those fields correlate with the other flow-entry in the accused products.⁵⁷⁴

⁵⁷³ Ex. 1069 (10/12/17 Trial Transcript in *NetScout* district court case), 25:14-26:12.

⁵⁷⁴ Ex. 1069 (10/12/17 Trial Transcript in *NetScout* district court case), 48:23-50:14.

451. In the above testimony, Dr. Almeroth refers to the below demonstrative which highlights use of the HTTP host, URL, User Agent, User Agent Profile, and HTTP Referrer fields as evidence of the claimed “conversational flow.”⁵⁷⁵ As illustrated by RFC1945, these HTTP headers fields were well-known to a POSITA before the priority date of the Challenged Patents. For example, a POSITA would have understood that HTTP Referrer fields were used to relate traffic flows. Accordingly, at least under Patentee’s “conversational flow” construction, Patentee’s reliance on the HTTP Referrer field as linking connection flows into a conversational flow demonstrates the obviousness of identifying conversation flows in Riddle’s system.

⁵⁷⁵ Ex. 1074 (Dr. Almeroth’s Direct Testimony Demonstrative Slide in *NetScout* district court case), 27-30; Ex. 1073 (S. Udick Declaration in *NetScout* district court case), ¶3.

header types such as Content-Type (MIME type) or User-Agent.”⁵⁷⁶ Thus, Riddle teaches classifying traffic using one of the HTTP fields, such as User-Agent. As discussed above regarding Dr. Almeroth’s testimony, I understand Patentee previously relied on this HTTP field when attempting to show an accused product infringes the Challenged Patents.

454. RFC1945 discloses 16 different HTTP header fields, including Content-Type and User-Agent.⁵⁷⁷ And RFC1945 teaches the use of such fields, such as the HTTP Referrer field, to associate web traffic together.⁵⁷⁸ A POSITA would have understood that in addition to Content-Type and User-Agent, any of the 14 other HTTP header fields, such as the HTTP Referrer field, are useable to classify web traffic.

455. Further, RFC1945 reflects common usage of the HTTP 1.0 protocol and Riddle demonstrates that a POSITA would have been aware of the HTTP protocol.⁵⁷⁹ For example, Riddle states: “The Hypertext Transfer Protocol is a simple protocol built on top of Transmission Control Protocol (TCP). It is the mechanism which underlies the function of the World Wide Web. The HTTP provides a

⁵⁷⁶ Riddle, 9:24-27. While this exact sentence does not appear in the ’864 Provisional, its content is supported by other sections of the ’864 Provisional. Ex. 1024 (’864 Provisional), 16 (lines 15-24), 67 (“MIME-type”), 68 (“MIME-Type”).

⁵⁷⁷ RFC1945, 37-46.

⁵⁷⁸ RFC1945, 44-45.

⁵⁷⁹ Riddle, 9:24-27.

method for users to obtain data objects from various hosts acting as servers on the Internet.”⁵⁸⁰

456. Riddle teaches using other “web aware” class attributes in defining traffic classes. Those “web aware” attributes include HTTP hosts and HTTP URIs:

At the application level, traffic classes may be defined for specific *URIs* within a web server. *Traffic classes may be defined having “Web aware” class attributes. For example, a traffic class could be created such as all URIs matching “*.html” for all servers, or all URI patterns matching “*.gif” for server X, or for access to server Y with URI pattern “/sales/*” from client Z, wherein “*” is a wildcard character, i.e., a character which matches all other character combinations.* Traffic class attributes left unspecified will simply match any value for that attribute. For example, a traffic class that accesses data objects within a certain directory path of a web server is specified by a URI pattern of the directory path to be managed, e.g. “/sales/*”.⁵⁸¹

As discussed above regarding Dr. Almeroth’s testimony, I understand Patentee previously asserted those two “web aware” HTTP fields show an accused product identifies packets are part of a conversational flow.

457. RFC1945 defines a URI as:

URIs have been known by many names: WWW addresses, Universal Doc-

⁵⁸⁰ Riddle, 8:41-45.

⁵⁸¹ Riddle, 8:64-9:11.

ument Identifiers, Universal Resource Identifiers, and finally the combination of Uniform Resource Locators (URL) and Names (URN). As far as HTTP is concerned, Uniform Resource Identifiers are simply formatted strings which identify--via name, location, or any other characteristic--a network resource.⁵⁸²

458. By disclosing general schemes for defining traffic classes, including that traffic classes may be defined based on URIs, Riddle discloses that traffic classes may be defined by HTTP host and HTTP URI fields. Moreover, a POSITA would have been aware of the common protocol HTTP/1.0 described in RFC1945 when considering the role of HTTP and TCP in Internet transmissions as set forth in Riddle.⁵⁸³

459. Based on the knowledge of a POSITA and the teachings of RFC1945, a POSITA would have been motivated to modify Riddle to use other header fields, such as the HTTP Referrer field, to classify based on the conversational flow of web traffic.⁵⁸⁴ Riddle provides motivation to do so by stating “[w]eb traffic may also be classified by HTTP header types *such as* Content-Type (MIME type) or User-Agent.”⁵⁸⁵ Riddle’s statement is open-ended and gives two examples of the HTTP header fields that may be used. As Riddle already discloses using other

⁵⁸² RFC1945, 10, 14 (endnote citations omitted).

⁵⁸³ Riddle, 9:24-27; 8:41-45.

⁵⁸⁴ RFC1945, 37-46.

⁵⁸⁵ Riddle, 9:25-27.

HTTP header fields in defining classes, using RFC1945's HTTP Referrer field when Riddle classifies flows amounts to nothing more than combining known prior art technologies used in their ordinary and predictable manner to examine and classify flows in Riddle's monitor.

460. Combining the teachings of Riddle and RFC1945 yields the ability to create a traffic class based upon URIs (such as HTTP host or HTTP URL), the user agent, and/or the HTTP Referrer. In doing so, a POSITA would have appreciated that Riddle in view of Ferdinand and further in view of RFC1945 creates traffic classes that contain the information that I understand Patentee has previously accused of infringement of the Challenged Patents (as discussed above regarding Dr. Almeroth's testimony).

2. Riddle in View of Ferdinand and Further in View of RFC1945 Renders Obvious Identifying A "Conversational Flow-Sequence."

461. As set forth in my analysis of '099 claim element 1.4 in Section VII.A.2.e above, Riddle discloses using HTTP header types, such as User-Agent, to classify HTTP traffic. As discussed above, Patentee asserted using HTTP fields like User-Agent to classify flows is an example of identifying a conversational flow.

462. To the extent Riddle does not disclose or render obvious identifying a "conversational flow sequence," Riddle in view of Ferdinand and further in view of

RFC1945 renders obvious this claim element, at least under Patentee's interpretation of "conversational flow."

463. RFC1945 further demonstrates identifying conversational flows through its teachings regarding HTTP header fields. Based on the knowledge of a POSITA and the teachings of RFC1945 as discussed above in this section, a POSITA would have been motivated to modify Riddle to use other header fields, such as the HTTP Referrer field, to classify based on the conversational flow of web traffic.⁵⁸⁶

464. Combining the teachings of Riddle, Ferdinand, and RFC1945 yields the ability to create a traffic class based upon URIs, the user agent, and/or the HTTP Referrer. In doing so, a POSITA would have appreciated that the combination of Riddle and Ferdinand and further in view of RFC1945 creates traffic classes that contain the information that I understand Patentee has previously accused of infringement of the Challenged Patents. For all the reasons discussed above in this section, a POSITA would have been motivated to combine the teachings of Riddle, Ferdinand, and RFC1945 to render obvious extracting selected packet portions and forming a function of the selected portions sufficient to identify that the accepted packet is part of a conversational flow-sequence.

465. As set forth in my analysis of the '099 Patent in Sections VII.A.2 and VII.A.3 above, Riddle and Ferdinand disclose or render obvious all the remaining

⁵⁸⁶ RFC1945, 37-46.

elements of '099 claims 1 and 2. As such, it is also my opinion that Riddle in view of Ferdinand and further in view of RFC1945 renders obvious claims 1 and 2 of the '099 Patent, at least under the Patentee's interpretation of "conversational flow."

F. For the '099 Patent, Riddle in View of Ferdinand and Baker and Further in View of RFC1945 Renders Obvious Dependent Claims 4 and 5.

466. As I discuss above in Section VII.E, Riddle in view of Ferdinand and further in view of RFC1945 renders obvious independent '099 claim 1. Further, as I discuss above in Section VII.B, Riddle in view of Ferdinand and further in view of Baker render obvious '099 claims 4 and 5. Because claims 4 and 5 depend from independent claim 1, it is also my opinion that a POSITA would have recognized that each and every limitation of the '099 claims 4 and 5 is disclosed or rendered obvious in light of Riddle in view of Ferdinand and Baker and further in view of RFC1945. Specifically, my opinions regarding Riddle in view of Ferdinand and Baker are exactly the same as those above in Section VII.B, but further include the teachings of RFC. Thus, as if fully set forth here, I incorporate the discussion from Section VII.B, regarding the obviousness of '099 Claims 4 and 5 over Riddle in view of Ferdinand and further in view of Baker.

467. As discussed above, all Challenged Claims require "conversational flows" or a "conversational-flow sequence." For example, '099 claim element 1.7 recites

“the particular conversational-flow sequence is associated with the operation of a particular application program.” While Riddle itself teaches identifying conversational flows, RFC1945 further demonstrates identifying conversational flows through the additional example of the use of HTTP header fields.

468. For all the reasons set forth in VII.E, a POSITA would have been motivated to combine the teachings of Riddle, Ferdinand, Baker, and RFC to render obvious identifying that an accepted packet is part of a “conversational flow-sequence” as recited in claim 1.

469. As set forth in my analysis in above Sections VII.B.2 and **Error! Reference source not found.** above, Riddle, Ferdinand, and Baker disclose or render obvious all the remaining elements of '099 claims 4 and 5. As such, it is also my opinion that Riddle in view of Ferdinand and Baker and further in view of RFC1945 renders obvious claims 4 and 5 of the '099 Patent.

VIII. THE CLAIMS OF THE '725 PATENT ARE UNPATENTABLE

470. For the '725 Patent, the challenged claims include independent claims 10 and 17 and dependent claims 12, 13, and 16. As I detail below, it is my opinion that Riddle in view of Baker renders obvious '725 claims 10, 12, 13, 16, and 17. It is also my opinion that Riddle in view of Baker and further in view of Yu renders obvious '725 claims 10, 12, 13, 16, and 17. Further, it is my opinion that Riddle in view of Baker and further in view of RFC1945 renders obvious '725 claims 10, 12,

13, 16, and 17.

A. For the '725 Patent, Riddle in View of Baker Renders Obvious Claims 10, 12, 13, 16, and 17.

471. It is my opinion that a POSITA would have recognized that each and every limitation of the '725 claims 10, 12, 13, 16, and 17 is disclosed or rendered obvious by the prior art. Specifically, it is my opinion that '725 claims 10, 12, 13, 16, and 17 are obvious in light of Riddle in view of Baker.

1. Reasons to Modify Riddle in View of Baker

472. As I discuss above regarding '099 claims 4 and 5 in Section VII.B.1, Riddle and Baker are in the same field of endeavor and contain overlapping disclosures with similar purposes. For example, Riddle discloses classifying packets in a data communications network into traffic classes based upon protocols such as FTP, HTTP, TCP, UDP, and/or Ethernet.⁵⁸⁷ And Baker, for example, teaches parsing, filtering, generating and analyzing data (or frames of data) transmitted over a data communications network based on one or more programmably configurable protocol descriptions which may be stored and retrieved from an associated memory.⁵⁸⁸ As if fully set forth here, I incorporate the discussion from Section VII.B.1.

473. Based on the teachings of Riddle and Baker and before the time of the purported invention, a POSITA would have been motivated and found it obvious to

⁵⁸⁷ Riddle, Fig. 1D, Table 2.

⁵⁸⁸ Baker, 3:32-4:6.

make the packet flow classifier of Riddle more configurable based on a POSITA's general knowledge and Baker's teachings.

474. In particular, it would have been obvious to a POSITA to modify Riddle's teaching of a traffic classification system with Baker's teaching of PDFs so that Riddle's traffic classification system could include receiving PDFs because a POSITA understood it would be advantageous for Riddle's classification system to allow for updating protocol descriptions.⁵⁸⁹ This would be highly desirable in Riddle's monitor because Riddle envisions that the network managers using the monitor do not know the technical details of the protocols.⁵⁹⁰

475. Moreover, Baker provides an explicit motivation to combine: "[I]t would be highly desirable to be able to parse all protocols with a single configurable software (or hardware) module so that support for additional protocols could be added to a system without requiring substantial modification to the system or its control logic."⁵⁹¹ A POSITA would have realized that combining Baker with Riddle provides the advantage of making Riddle's monitor more configurable without requiring substantial modification to the monitor itself.

476. Additionally, Baker states that its invention may be used with network monitors such as those disclosed in Riddle. Baker's invention "may be employed in any

⁵⁸⁹ Baker, 19:6-10.

⁵⁹⁰ Riddle, 15:28-31.

⁵⁹¹ Baker, 3:3-8.

system where it is useful to be able to examine and perform various operations on contiguous bit-fields in data structures, wherein each data structure is composed of predefined fields of one or more contiguous bits.”⁵⁹² Baker also discloses that the invention “may be incorporated in a network device, such as a network analyzer, bridge, router, or traffic generator, including a CPU and a plurality of input devices, storage devices, and output devices.”⁵⁹³ These are the same types of devices as Riddle’s monitor.⁵⁹⁴ Given Baker’s disclosures, this combination is nothing more than a simple substitution of one known element for another used in their ordinary and predictable manner to update Riddle’s protocols.

477. Further, Riddle discloses traffic classification based upon the Ethernet protocol and Baker discloses PDFs for the Ethernet protocol.⁵⁹⁵ And Baker describes PDFs for Generic Protocols which could be other protocols used in Riddle.⁵⁹⁶ Thus, a POSITA would have been motivated to modify the teachings of Riddle to increase the ability to handle updated PDFs as taught by Baker.⁵⁹⁷ This analysis applies to each of the ’725 claim elements described below that rely on Riddle in view of Baker.⁵⁹⁸

⁵⁹² Baker, 6:18-22.

⁵⁹³ Baker, 4:27-30

⁵⁹⁴ Riddle, 5:55-57, claim 8, Figs. 1A, 1B, 1C.

⁵⁹⁵ Riddle, Fig 1D, 7:47-53; Baker, Table 12, 21:32-22:1.

⁵⁹⁶ Baker, Table 13.

⁵⁹⁷ Baker, Table 13.

⁵⁹⁸ Baker, Table 13.

2. Independent '725 Claims 10 and 17

478. It is my opinion that independent claims 10 and 17 of the '725 Patent are obvious in light of Riddle in view of Baker.

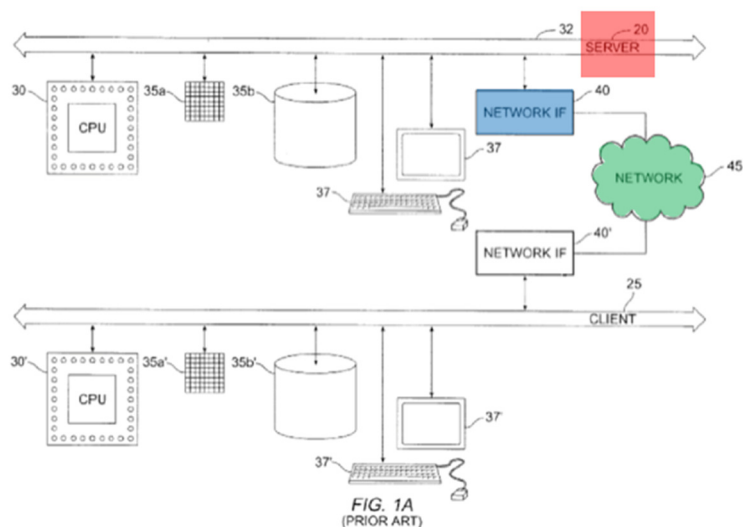
- a. *'725 Claim 10's Preamble and 17's Preamble: "A method of performing protocol specific operations on a packet passing through a connection point on a computer network, the method comprising"*

479. Riddle discloses all elements of this preamble. As I detail below, Riddle describes a method of performing protocol specific operations. And the combination of Riddle and Baker renders obvious performing protocol specific operations as shown below regarding '725 claim elements 10.3 and 17.3.

480. Riddle teaches performing protocol specific operations on packets passing through a network connection point. Riddle implements its methods using the C programming language and operating them on a computer system such as shown in Riddle's Figure 1A.⁵⁹⁹ As provided below, Figure 1A shows a computer network including client 25 or server 20, on which Riddle's invention is implemented. Client 25 and server 20 connect to the network at connection points, *e.g.*, network interfaces 40 and 40'.⁶⁰⁰

⁵⁹⁹ Riddle, 5:55-57.

⁶⁰⁰ Riddle, Fig. 1A, 6:9-15.



481. For networks connecting multiple clients and servers of Figure 1A, Riddle teaches network access through a “network routing means” and routers, such as router 75, shown below in Figure 1C, for implementing Riddle’s methods, or in another traffic classifier 304.⁶⁰¹ A POSITA would have understood, consistent with Riddle’s teachings of traffic classification for bandwidth management, that Riddle’s invention could be implemented in routers, such as router 75 depicted in Figure 1C. As shown in that figure, Riddle’s router 75 is coupled to the connection point. Thus, Riddle discloses a method of performing protocol specific operations on a packet passing through a connection point on a computer network.

⁶⁰¹ Riddle, 7:10-34, claim 8, Figs. 1C, 3.

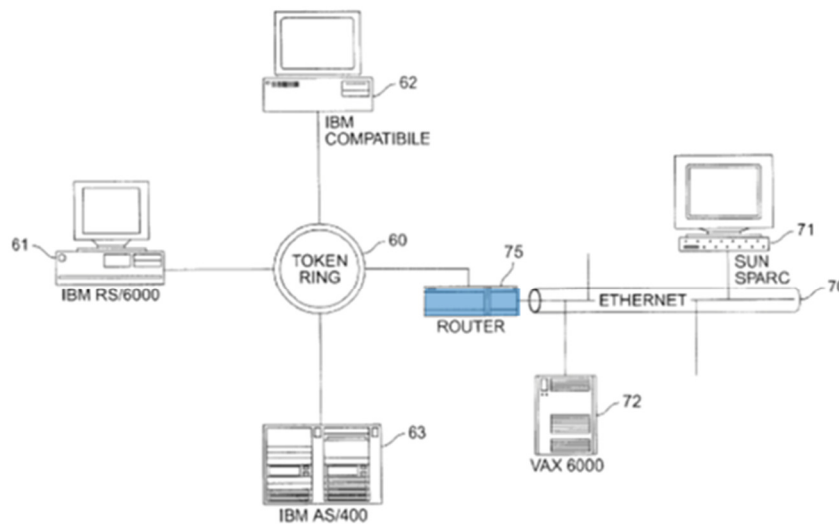


FIG. 1C
(PRIOR ART)

b. '725 Claim Elements 10.1 and 17.1: “(a) receiving the packet”

482. Riddle discloses this claim element alone and/or renders it obvious in view of Baker. Riddle teaches methods for automatically classifying packet flows for use in allocating bandwidth resources by a rule of assignment of a service level.⁶⁰² Riddle’s methods includes “applying individual instances of traffic classification paradigms to packet network flows based on selectable information obtained from a plurality of layers of a multi-layered communication protocol in order to define a characteristic class, then mapping the flow to the defined traffic class.”⁶⁰³ To classify packet flows, a POSITA would have understood that Riddle’s monitor receives packets at network interface 40, for example.

⁶⁰² Riddle, Abstract, 4:7-10.

⁶⁰³ Riddle, Abstract, 4:10-15.

483. Riddle's automatic classification methods include parsing a flow specification from a packet, which a POSITA would have understood occurs after the packet is received by the monitor.⁶⁰⁴ Further, Riddle discloses that "traffic classifier 304 detects services for incoming traffic."⁶⁰⁵ To detect services for incoming traffic, i.e., packets, a POSITA would have understood Riddle's monitor receives those packets.

484. Additionally, Baker discloses receiving network data:

In one preferred form, the system of the present invention may employ a CPU or other hardware implementable method for analyzing data from a network in response to selectively programmed parsing, filtering, statistics gathering, and display requests. Moreover, the system of the present invention may be incorporated in a network device, such as a network analyzer, bridge, router, or traffic generator, including a CPU and a plurality of input devices, storage devices, and output devices, *wherein frames of network data may be received from an associated network*, stored in the storage devices, and processed by the CPU based upon one or more programmably configurable protocol descriptions also stored in the storage devices.⁶⁰⁶

⁶⁰⁴ Riddle, claims 1, 8, Fig. 4A, step 402 ("parse flow specification from a packet of the flow").

⁶⁰⁵ Riddle, 12:30-31, Fig. 3.

⁶⁰⁶ Baker, 4:22-35.

- c. *'725 Claim Elements 10.2 and 17.2: “(b) receiving a set of protocol descriptions for a plurality of protocols that conform to a layered model, a protocol description for a particular protocol at a particular layer level including”*

485. Riddle discloses this claim element alone and/or renders it obvious in view of Baker. Moreover, the Board previously found Baker taught this claim element.⁶⁰⁷

(1) Riddle discloses receiving protocol descriptions

486. Riddle discloses using protocol descriptions for the creation of traffic classes. Riddle discloses “management of network bandwidth based on information ascertainable from multiple layers of OSI network model.”⁶⁰⁸ Riddle’s methods include “applying individual instances of traffic classification paradigms to packet network flows based on selectable information obtained from a plurality of layers of a multi-layered communication protocol in order to define a characteristic class, then mapping the flow to the defined traffic class.”⁶⁰⁹ And Riddle specifies that its “automatic classification is sufficiently robust to classify a complete enumeration of the possible traffic.”⁶¹⁰

⁶⁰⁷ Ex. 1062 (IPR2017-00863 Institution Decision), 11-12.

⁶⁰⁸ Riddle, 1:54-57.

⁶⁰⁹ Riddle, Abstract, 4:10-15.

⁶¹⁰ Riddle, Abstract, 4:15-17.

487. As provided below, Riddle’s Table 2 provides an exemplary list of information used to build traffic classes.⁶¹¹ Thus, as an example, a traffic class could be defined for an IP address and/or a port number, as well as a URI⁶¹² pattern.

Components of a Traffic Class Specifier

Inside (Client or Server)	Global	Outside (Server or Client)
IP Address/Domain Name	TCP or UDP Service	IP Address/Domain Name
Port Number	e.g., WWW,	Port Number
MAC Address	FTP, RealAudio, etc. URI pattern for Web Service, MIME type for Web Service IPX Service SNA Service LAT Service IP precedence	MAC Address

488. Riddle proposes classifying traffic “in protocol layer independent categories. For example, a particular instance of traffic may be classified according to its transport layer characteristics, e.g., Internet Protocol port number, as well as its application layer information, e.g., SMTP.”⁶¹³ A POSITA would have understood that Riddle’s invention also classifies traffic based upon the Ethernet protocol as shown in the prior art OSI network model of Figure 1D.

⁶¹¹ Riddle, 9:64-65.

⁶¹² Riddle, Table 1 (“A Universal Resource Identifier is the name of the location field in a web reference address. It is also called a URL or Universal Resource Locator.”)

⁶¹³ Riddle, 10:57-11:9.

(2) Baker teaches receiving protocol descriptions

489. Baker discloses storing protocol descriptions as “protocol description files (PDF).”⁶¹⁴ As shown below in Baker’s Figure 1, PDFs 22 are stored in data storage device 14 and accessible by network device control logic 16.⁶¹⁵

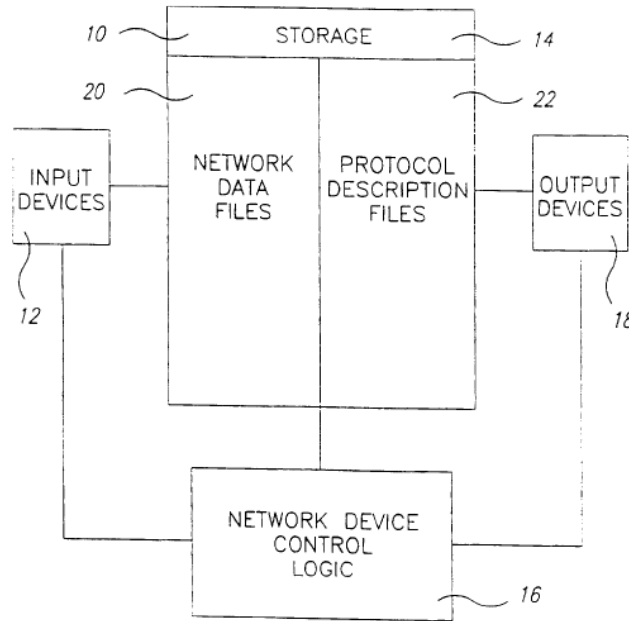


FIG. 1

490. Baker discloses that the network interface system stores the programmably configurable PDFs and is programmed to receive PDFs and run an initialization/compilation process (i.e., receiving a set of protocol descriptions) as follows:

[T]he initialization of the system includes a determination of the presence of PDF files and the extraction of the protocol and associated control record information from all of the PDF files found. The number of

⁶¹⁴ Baker, 19:6-10 (“In the presently preferred embodiment, each of these protocol description records with its associated field, statistics, lookup, and filter record information is also written to a protocol specific protocol description file (PDF).”).

⁶¹⁵ Baker, 10:10-25, Fig. 1.

PDF files is determined, and a ProtocolList is constructed consisting of a sorted vector of protocol records at least the size of the number of PDF files found. The name of each protocol record found is inserted in the ProtocolList. The PDF files are then read to memory in the sequence described above for the PDF file writing. The lookup records that indicate a next protocol are associated with the appropriate entries in the ProtocolList.⁶¹⁶

491. Moreover, the claims of Baker confirm that the PDFs are received. For example, claim 3 recites: “**storing** at least one programmably configurable protocol description in a memory,” “**retrieving** said at least one protocol description from said memory”, and “**providing** said at least one protocol description to a logic control module,” “**upon receiving** said at least one protocol description.”⁶¹⁷

492. Baker’s PDFs may include a protocol control record and a plurality of field data records.⁶¹⁸ An example of a protocol control record is shown below in Baker’s Table 1, and the field data records are shown below in Baker’s Table 2.

⁶¹⁶ Baker, 20:35-21:11.

⁶¹⁷ Baker, claim 3, claims 5-8 (claiming “storing,” “retrieving,” “providing,” and “receiving” “[programmable configurable] protocol descriptions”), claims 9-11 (claiming “storing,” “delivering,” and “retriev[ing]” “programmable configurable protocol descriptions”).

⁶¹⁸ Baker, 12:25-28.

TABLE 1

PROTOCOL CONTROL RECORD		
Offset	Name	Description
0-3	name_length	length of protocol name in bytes including NULL terminator
4-7	protocol_name	name of protocol control record is describing
8-11	filename	name of file control record is stored in
12-15	numBits	total bit length of protocol header control record is describing
16-17	numFields	number of fields required to describe protocol header
18-19	curField	index of field currently referenced
20-23	outFlag	flag indicating template has been output to file
24-27	dbW	display bit width for protocol header display
28-31	fields	field records that describe protocol header
32-25	options	pointer to option control record to use if this protocol has optional fields
36-39	Rt	pointer to protocol specific routing table

TABLE 2

FIELD SUB-RECORDS		
Offset	Name	Description
0-3	fplen	flag indicating value is actual length of frame (multiplier)
4-7	fblen	length of field in bits
8-11	fdwoff	byte offset from start of protocol header of 32-bit field containing value
12	fshl	number of bits to left shift 32-bit value
13	fshr	number of bits to right shift 32-bit value

14	ffmt	number indicating a display type (i.e., decimal, hex, ...)
15	fiflag	flag indicating value is actual length of protocol header (multiplier)
16	reserved	not used ... pad byte to align following fields
17	fmult	multiplier to apply to value prior to display
18	fswap	flag indicating the need to swap bytes and words in 32-bit field containing value
19	fdspfield	flag indicating that this field should be displayed
20-23	fname	pointer to field name (0=none)
24-27	ptr2stats	pointer to configured statistics structure/class (0=none)
28-31	ptr2np	pointer to lookup structure/class ... next protocol definition to use (0=none)
32-35	ptr2vary	pointer to vary field value structure/class (0=none)
36-39	ptr2caum	pointer to checksum structure/class (0=none)
40-43	ptr2flt	pointer to filter criteria structure (0=none)
44-47	ptr2rte	pointer to Route Table structure/class (0=none)

493. Baker specifically discloses supporting the Ethernet protocol, and also describes supporting protocols “for any possible protocol specification”:

It will be appreciated by those skilled in the art that any possible organization of fields for any possible protocol specification is contemplated for the network interface system 10 of the present invention.⁶¹⁹

Further, Baker specifically mentions configuring “IP, TCP, UDP and IPX checksum capabilities” as part of the application control logic, and provides source code

⁶¹⁹ Baker, 21:32-22:5, 12:21-24.

to support such capabilities for these protocols.⁶²⁰

494. Baker relates to programmably configurable protocol descriptions for layered models such as the OSI model.⁶²¹ A POSITA would have understood that the OSI model is a layered model. For example, as provided below, Riddle's Figure 1D shows that the OSI model is prior art to the Challenged Patents.⁶²²

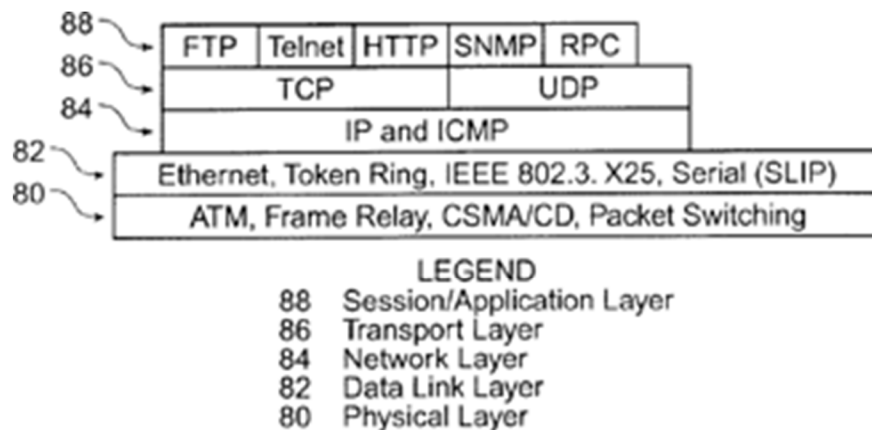


FIG. 1D
(PRIOR ART)

495. Baker provides two example protocol descriptions in Tables 12 and 13, as annotated below, that further confirm the protocol descriptions include protocols

⁶²⁰ Baker, 34:16-21, 135:28-141:8, 213:24-35.

⁶²¹ Baker, Abstract, 1:22-25 (“Bridges operate at the data-link layer and routers at the network layer of the OSI reference model”), 4:14-21 (“[N]etwork interface in accordance with the present invention may be configured and reconfigured ... to accommodate substantial network modifications (for example, the use of different data transmission hardware, protocols or protocol suites) without necessitating substantial system changes”), 12:21-24 (“It will be appreciated by those skilled in the art that any possible organization of fields for any possible protocol specification is contemplated for the network interface system 10 of the present invention.”).

⁶²² Riddle, Fig. 1D.

that conform to a layered model.⁶²³

TABLE 12
ETHERNET v2.0 PROTOCOL SPECIFICATION

0	15	23	47
Destination Hardware Address			
Source Hardware Address			
Ethernet Protocol Type			
Destination Hardware Address	-	destination hardware station address (48 bits)	
Source Hardware Address	-	source hardware station address (48 bits)	
Ethernet Protocol Type	-	upper layer protocol designator (16 bits) 0x8888=GP	

496. The Ethernet protocol definition shown above in Baker's Table 12 specifies one possible *upper level protocol*, the Generic Protocol (GP) which is indicated by placing hexadecimal value 0x8888 in the protocol type field.⁶²⁴

497. Baker's Table 13 below shows that the Generic Protocol (GP) Specification has an 8-bit upper level protocol identifier field which shows placeholders for a network manager to program upper level protocols.

⁶²³ Baker, Tables 12 and 13.

⁶²⁴ Baker, 21:32-22:1.

TABLE 13
 GENERIC PROTOCOL (GP) SPECIFICATION

0	7	15	23	31
Version No.	HeaderLen	Frame Type	Frame Length	
Checksum			Control	Hop Count
Src Socket			Dst Socket	
Src Address				
Dst Address				
Optional fields				

- Version Number (4 bits) - software version number
- HeaderLen (4 bits) - length of GP header in 32 bit words.
 0- 4 = illegal
 5 = No Optional fields,
 6-15 = 32-320 bits of options
- Frame Type (8 bits) - upper level protocol identifier
 0 = Unknown
 1 = GP1
 2 = GP2
 3-255 = Unknown
- Frame Length (16 bits) - frame length in bytes including header
- Checksum (16 bits) - Checksum of header including options
- Control (8 bits) - reserved (must be zero)

498. Table 14 further confirms that Baker’s PDFs use a layered model. Table 14 shows a variable “ParseLvl” that is described as “Zero based protocol level in ISO reference model of protocol being parsed (current protocol).”⁶²⁵ Again, a POSITA would understand the ISO reference model to be referring to the International Organization of Standardization’s OSI layered model discussed above.

⁶²⁵ Baker, Table 14.

499. To the extent that Riddle does not explicitly disclose this claim element, it would have been obvious to a POSITA to modify Riddle in view of Baker since both involve layered models in data communications networks. Baker relates to programmably configurable protocol descriptions for layered models such as the OSI model.⁶²⁶ Further, Figure 1D of Riddle shows that the OSI model is prior art to the Challenged Patents.⁶²⁷ Further, Riddle discloses traffic classification based upon the Ethernet protocol and Baker discloses PDFs for the Ethernet protocol.⁶²⁸ Baker also discloses PDFs for Generic protocols which could be the other protocols used in Riddle.⁶²⁹ Thus, a POSITA would be motivated to modify the teachings of Riddle to increase the ability to handle updated PDFs as taught by Baker.

⁶²⁶ Baker, Abstract, 1:22-25 (“Bridges operate at the data-link layer and routers at the network layer of the OSI reference model”), 4:14-21 (“[N]etwork interface in accordance with the present invention may be configured and reconfigured ... to accommodate substantial network modifications (for example, the use of different data transmission hardware, protocols or protocol suites) without necessitating substantial system changes”), 12:21-24 (“It will be appreciated by those skilled in the art that any possible organization of fields for any possible protocol specification is contemplated for the network interface system 10 of the present invention.”).

⁶²⁷ Riddle, Fig. 1D.

⁶²⁸ Riddle, Fig 1D, 7:47-53; Baker, Table 12, 21:32-22:1.

⁶²⁹ Baker, Table 13.

d. '725 Claim Elements 10.3 and 17.3: “(i) if there is at least one child protocol of the protocol at the particular layer level, the-one or more child protocols of the particular protocol at the particular layer level, the packet including for any particular child protocol of the particular protocol at the particular layer level information at one or more locations in the packet related to the particular child protocol”⁶³⁰

500. Riddle discloses this claim element alone and/or renders it obvious in view of Baker. Moreover, the Board previously found Baker taught this claim element.⁶³¹

501. First, the claimed “if there is at least one child protocol of the protocol at the particular layer level, the-one or more child protocols of the particular protocol at the particular layer level” relates to the protocol descriptions Riddle received in '725 claim elements 10.2 and 17.2.⁶³² Second, the claim language “*the packet* including for any particular child protocol of the particular protocol at the particular layer level information at one or more locations in the packet related to the particular child protocol” relates to Riddle’s received packets in '725 claim elements 10.1 and 17.1. Overall, this claim element describes that, if a child protocol exists, instructions are provided for identifying the child protocol.

⁶³⁰ It is my understanding that the '725 Patent’s Certificate of Correction corrects a typographical error in claim element 10.3.

⁶³¹ Ex. 1062 (IPR2017-00863 Institution Decision), 12-15.

⁶³² '725 Patent, Abstract, 3:60-4:21.

(1) Riddle teaches protocol descriptions that identify child protocols

502. Regarding protocol descriptions, Riddle describes programming its system to parse layered protocols, including identifying one or more child protocols at each layer.⁶³³ Riddle describes that “[t]raffic classes may be defined at any level of the IP protocol as well as for any other non-IP protocols.”⁶³⁴ Riddle also discloses that “[t]raffic class membership may be hierarchical. Thus, a flow may be classified by a series of steps through a traffic class tree, with the last step (i.e., at the leaves on the classification tree) mapping the flow to a policy.”⁶³⁵ Further, Riddle states:

A classification tree is a data structure representing the hierarchical aspect of traffic class relationships. *Each node of the classification tree represents a class, and has a traffic specification*, i.e. a set of attributes or characteristics describing the traffic associated with it. Leaf nodes of the classification tree may contain policies. According to a particular embodiment, *the classification process checks at each level if the flow being classified matches the attributes of a given traffic class. If it does, processing continues down to the links associated with that node in the tree.* If it does not, the class at the level that matches determines the policy for the flow being classified. If no policy specific match is

⁶³³ Riddle, Abstract, Fig. 1D, 4:10-17.

⁶³⁴ Riddle, 8:59-60.

⁶³⁵ Riddle, 9:20-24.

found, the flow is assigned the default policy.⁶³⁶

503. Additionally, even within a particular layer (such as the session/application layer described in Figure 1D), a particular protocol such as FTP may contain sub-components or child protocols. For example, Riddle explains its use of “subclassification” for FTP traffic:

Subclassification of traffic into a tree is performed by matching the hosts and then searching for particular services. Traffic *specifications are aggregate kinds of traffic for a traffic class, e.g., different components of FTP may reside under class FTP.*

Subclassification is performed by first locating a class that matches, and then performing finer grade matchings. Processing commences with a decision on what traffic is to be subclassified. A marker is placed in the match_all default node so that when match processing reaches the marker, the autotclassification processing depicted in flowchart 403, determines that it has not found an existing class for the traffic being classified.⁶³⁷

504. Riddle discloses examples such as identifying the child protocol TCP in the IP protocol, and identifying the child protocols HTTP and FTP within the TCP protocol.⁶³⁸ Also, Riddle states that the HTTP Protocol “is a simple protocol built

⁶³⁶ Riddle, 9:28-41.

⁶³⁷ Riddle, 11:24-36.

⁶³⁸ Riddle, 10:62-11:2; 13:10-22; Figs 2A, 2B.

on top of the Transmission Control Protocol (TCP).”⁶³⁹ HTTP, shown in the Application Layer of Figure 1D, is a child protocol of TCP, which is shown in the Transport Layer.⁶⁴⁰ Based on the plain meaning of child protocol, or Packet Intelligence’s former proposed construction of “a protocol that is encapsulated within another protocol,” HTTP is a child protocol of TCP since it is encapsulated within TCP.

505. Similarly, Riddle shows FTP is a child protocol of TCP:

File Transfer Protocol (FTP) is a standard TCP/IP protocol for transferring files from one machine to another. FTP clients establish sessions through TCP connections with FTP servers in order to obtain files.⁶⁴¹

506. Further, Riddle teaches examining data in a packet. For example, Riddle explains that HTTP “facilitates the transfer of data objects across networks via a system of uniform resource indicators,” or URLs.⁶⁴² Riddle notes that:

[C]lassification can extend to examination of the data contained in a flow's packets. Certain traffic may be distinguished by a signature even if it originates with a server run on a non-standard port, for example, an HTTP conversation on port 8080 would not be otherwise determinable as HTTP from the port number. Further analysis of the data is conducted in order to determine classification in instances where: 1) FTP

⁶³⁹ Riddle, 8:41-42.

⁶⁴⁰ Riddle, Fig. 1D.

⁶⁴¹ Riddle, 8:22:26, Fig. 1D.

⁶⁴² Riddle, 8:38-40, Table 1.

commands are used to define server ports, 2) HTTP protocol is used for non-web purposes. The data is examined for indication of push traffic, such as pointcast, which uses HTTP as a transport mechanism. These uses may be isolated and classified into a separate class. Marimba and *pointcast can be distinguished by looking into the data for a signature content header in the get request*. Pointcast has URLs that begin with “/FIDO-1/.”⁶⁴³

507. With regards to the claimed “the packet ...,” Riddle notes that “classification can extend to data contained in a flow’s packets.”⁶⁴⁴

508. Further, Riddle’s claim 1 describes “parsing a packet into a first flow specification” and “matching the first flow specification of the parsing step to a plurality of classes represented by a plurality of tree type nodes, each tree type node having a traffic specification.” And Riddle’s claim 4 further extends that method to include child classification when reciting “subclassification” process:

[I]f a matching classification tree type node was found in the matching step, said matching classification tree type node having *at least one child classification* tree type node, applying the matching, associating, and incorporating steps *to a particular child classification* tree type node as part of the classification.⁶⁴⁵

509. Further, claim 8 of Riddle explains that a processor “parse[s] a packet into a

⁶⁴³ Riddle, 11:48-63.

⁶⁴⁴ Riddle, 11:48-49.

⁶⁴⁵ Riddle, claim 4.

first flow specification” after which the processor “match[es] the first flow specification of the parsing step to a plurality of classes represented by a plurality of tree type nodes, each tree type node having a traffic specification and a mask.” Additionally, claim 10 includes “[t]he method of claim 8 wherein said matching step is applied to hierarchically-recognized classes.”

(2) Baker teaches protocol descriptions that identify child protocols

510. Baker teaches descriptions for layered protocols, including protocol descriptions identifying one or more child protocols at each of a plurality of layers in a packet.⁶⁴⁶ For example, Baker discloses “the protocol descriptions may take the form of one or more protocol description files for each supported network protocol and may include a protocol header record and plurality of field sub-records having data corresponding to an associated protocol and fields defined therein.”⁶⁴⁷ Further, Baker states “the system of the present invention also preferably includes logic for determining a next protocol description structure required to continue analyzing a network frame.”⁶⁴⁸

⁶⁴⁶ Baker, Abstract, 1:22-25, 2:1-7, 2:28-3:8.

⁶⁴⁷ Baker, 4:35-5:3.

⁶⁴⁸ Baker, 5:31-33.

TABLE 1

PROTOCOL CONTROL RECORD		
Offset	Name	Description
0-3	name_length	length of protocol name in bytes including NULL terminator
4-7	protocol_name	name of protocol control record is describing
8-11	filename	name of file control record is stored in
12-15	numBits	total bit length of protocol header control record is describing
16-17	numFields	number of fields required to describe protocol header
18-19	curField	index of field currently referenced
20-23	outFlag	flag indicating template has been output to file
24-27	dbW	display bit width for protocol header display
28-31	fields	field records that describe protocol header
32-25	options	pointer to option control record to use if this protocol has optional fields
36-39	Rt	pointer to protocol specific routing table

511. The relationship between children and parent protocols are shown below in Baker's Tables 1 and 2. For example, Table 1 shows field records that describe the protocols' header at bytes 28-31.

512. "The field records referenced at bytes 28-31 in [Table 1] above are preferably organized as shown in Table 2" shown below.⁶⁴⁹ Baker's Table 2 below shows at bytes 28-31 a pointer to a lookup structure/class for the next protocol or definition to use.

⁶⁴⁹ Baker, 13:15-20.

TABLE 2

<u>FIELD SUB-RECORDS</u>		
Offset	Name	Description
0-3	fplen	flag indicating value is actual length of frame (multiplier)
4-7	fblen	length of field in bits
8-11	fdwoff	byte offset from start of protocol header of 32-bit field containing value
12	fshl	number of bits to left shift 32-bit value
13	fshr	number of bits to right shift 32-bit value
14	ffmt	number indicating a display type (i.e., decimal, hex, . . .)
15	fflag	flag indicating value is actual length of protocol header (multiplier)
16	reserved	not used . . . pad byte to align following fields
17	fmult	multiplier to apply to value prior to display
18	fswap	flag indicating the need to Swap bytes and words in 32-bit field containing value
19	fsdspfield	flag indicating that this field should be displayed
20-23	fname	pointer to field name (0 = none)
24-27	ptr2stats	pointer to configured statistics structure/class (0 = none)
28-31	ptr2np	pointer to lookup structure/class . . . next protocol definition to use (0 = none)
32-35	ptr2vary	pointer to vary field value structure/class (0 = none)
36-39	ptr2csum	pointer to checksum structure/class (0 = none)
40-43	ptr2flt	pointer to filter criteria structure (0 = none)
44-47	ptr2rte	pointer to Route Table structure/class (0 = none)

513. Baker states: “The next protocol lookup records referenced in the field sub-record table (Table 2) at bytes 28-31 are preferably organized as shown in Table 4.”⁶⁵⁰ As annotated below, lookup structures such as those in Baker’s Table 4 “can be used for determining the next protocol control record to use. . . . This ability to specify branches on field values allows protocols with multiple overlapping structures to be specified and parsed dynamically.”⁶⁵¹

⁶⁵⁰ Baker, 14:20-30.

⁶⁵¹ Baker, 15:10-17.

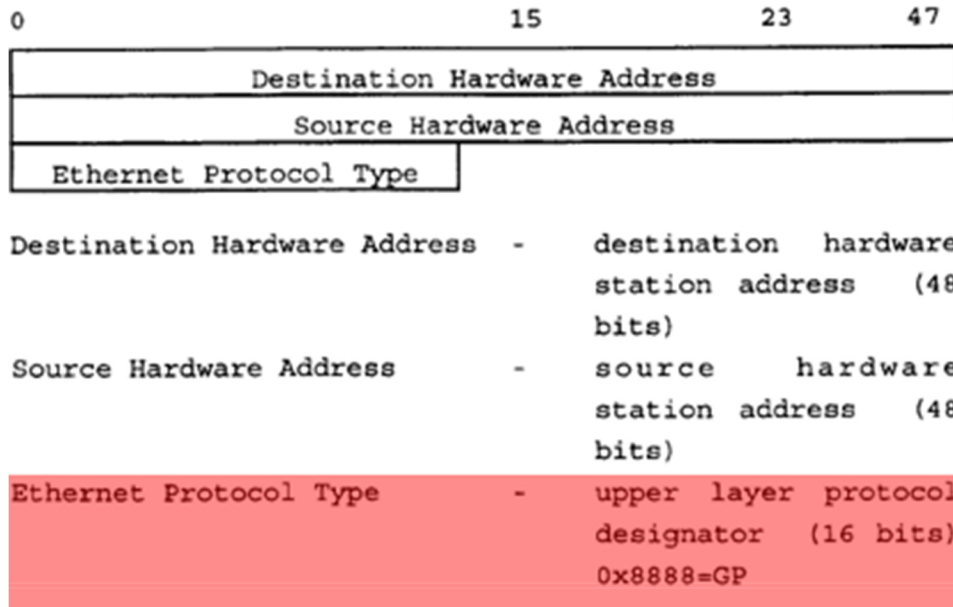
TABLE 4

LOOKUP STRUCTURE RECORD		
Offset	Name	Description
0-3	Protocol	pointer to protocol description structure
4-7	Next Index	index of field in protocol description to parse next
8-11	Minimum	minimum acceptable value for this range
12-15	Maximum	maximum acceptable value for this range
16-19	okbits	selects even only, odd only, or all values in range
20-23	Translation	pointer to associated human language equivalent

514. Baker provides two sample protocol descriptions, Ethernet and General Protocol, that further show this claim element has been met. The Ethernet protocol definition shown below in Table 12 specifies one possible *upper level protocol*, the Generic Protocol (GP) which is indicated by placing hexadecimal value 0x8888 in the protocol type field.⁶⁵²

⁶⁵² Baker, 21:32-22:1.

TABLE 12
ETHERNET v2.0 PROTOCOL SPECIFICATION



515. The Ethernet protocol description is further shown in Figs. 4-4D. These figures demonstrate one or more child protocols GP. For example, Baker's Figure 4D is shown below.

Ethernet Type Lookup Structure					
Protocol	Next Index	Minimum	Maximum	Mask	Translation
[None]	5	0x0000	0x8887	ALL	"Unknown"
Figure 5	5	0x8888	0x8888	ALL	"GP"
[None]	5	0x8889	0xFFFF	ALL	"Unknown"

Fig. 4D

516. Baker's Table 13 below shows that the Generic Protocol (GP) Specification has an 8-bit upper level protocol identifier field which shows placeholders for a network manager to program upper level protocols.⁶⁵³ Table 13 also shows Src

⁶⁵³ Baker, Table 13.

Socket for the socket of upper-layer protocol sender and a Dst Socket for the socket of Upper layer protocol receiver.⁶⁵⁴

TABLE 13
GENERIC PROTOCOL (GP) SPECIFICATION

0		7		15		23		31	
Version No.		HeaderLen		Frame Type		Frame Length			
Checksum				Control		Hop Count			
Src Socket				Dst Socket					
Src Address									
Dst Address									

Version Number (4 bits) - software version number
HeaderLen (4 bits) - length of GP header in 32 bit words.
0- 4 = illegal
5 = No Optional fields,
6-15 = 32-320 bits of options

Frame Type (8 bits) - upper level protocol identifier
0 = Unknown
1 = GP1
2 = GP2
3-255 = Unknown

Frame Length (16 bits) - frame length in bytes including header
Checksum (16 bits) - Checksum of header including options
Control (8 bits) - reserved (must be zero)

Hop Count (8 bits) - Count of number of networks traversed

Src Socket (16 bits) - Socket of Upper-layer protocol sender
Dst Socket (16 bits) - Socket of Upper-layer protocol receiver
Src Address (32 bits) - Sender protocol address
Dst Address (32 bits) - Receiver protocol address

517. The GP protocol description is further shown below in Baker's Figures 5C to 5E. For example, Figs. 5C, 5D, and 5E show the GP protocol description for GP, including four child protocols GP1, GP2, GP3, and GP4.

⁶⁵⁴ Baker, Table 13.

Frame Type Next Protocol Structure					
Protocol	Next Index	Minimum	Maximum	Mask	Translation
[None]	4	0x00	0x00	ALL	"Illegal Protocol"
GP1	4	0x01	0x01	ALL	"GP1"
GP2	5	0x02	0x02	ALL	"GP2"
[None]	4	0x03	0xFF	ALL	"Illegal Protocol"

Fig. 5C

Source Socket Next Protocol Structure					
Protocol	Next Index	Minimum	Maximum	Mask	Translation
[None]	8	0x0000	0x0142	ALL	"Unknown Protocol"
GP3	8	0x0143	0x018F	ODD	"GP3"
GP4	8	0x0143	0x018F	EVEN	"GP4"
[None]	8	0x0190	0xFFFF	ALL	"Illegal Protocol"

Fig. 5D

Destination Socket Next Protocol Structure					
Protocol	Next Index	Minimum	Maximum	Mask	Translation
[None]	9	0x0000	0x0142	ALL	"Unknown Protocol"
GP3	9	0x0143	0x018F	ODD	"GP3"
GP4	9	0x0143	0x018F	EVEN	"GP4"
[None]	9	0x0190	0xFFFF	ALL	"Illegal Protocol"

Fig. 5E

518. Further, Baker discloses receiving packets with specific locations for child protocol related information. Example Frames 1 and 2 that use the aforementioned PDFs are shown below. Baker's "Frame 1 shown below has a hardware length of eighty-two 8-bit bytes and consists of a fourteen byte Ethernet header, a twenty

byte GP header with no option bytes, and forty-eight bytes of application data.”⁶⁵⁵

Frame (1)																						
(1)	08	00	00	00	00	03	08	00	00	00	00	04	88	88						Ethernet Header	(14)	
(2)	35	00	00	44	B1	5F	00	01	08	00	01	47	01	02	03	04	05	06	07	08	GP Header	(20)
(3)	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	Data	(24)
(4)	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	Data	(24)

519. Baker’s “Frame 2 shown below has a hardware length of sixty 8-bit bytes and consists of a fourteen byte Ethernet header, a twenty-eight byte GP header including eight option bytes, and eighteen bytes of pad to achieve the sixty byte Ethernet minimum frame length requirement.”⁶⁵⁶

Frame (2)																						
(1)	08	00	00	00	00	01	08	00	00	00	00	02	88	88						Ethernet Header	(14)	
(2)	37	03	00	2A	FF	FF	00	05	08	00	01	00	01	02	03	04	05	06	07	08	GP Header	(20)
(3)	01																				GP NoOp Option	(1)
(4)	02	04	00	00																	GP MaxSizeOpt	(4)
(5)	00																				GP EOL Options	(1)
(6)	00	00																			GP Option Pad	(2)
(7)	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	Frame Pad	(18)

520. Additionally, each PDF protocol control record may include a “NumBits” attribute that is the “total bit length of the protocol header.”⁶⁵⁷ For example, the Ethernet Control Record shown below in Baker’s Figure 4 sets the NumBits attribute to 112, indicating the total length of the Ethernet header.

⁶⁵⁵ Baker, 25:29-34.

⁶⁵⁶ Baker, 26:10-14.

⁶⁵⁷ Baker, Table 1, 19:17, 56:30-57:11.

Ethernet Control Record					
Protocol Name	NumBits	NumFields	CurField	Fields	Options
Ethernet MAC Header	112	5	0	Figure 4a	[None]

Fig. 4

521. Further, as annotated below in Figure 4a, Baker’s Ethernet PDF “Type” field sub-record includes a “Bit Offset” indicating that the Type field is 96-bits offset from the start of the Ethernet header⁶⁵⁸:

Index	Field Name	Bit Offset	Bit Length	Left Shift	Right Shift	Check sum	Frame Length	Header Length	Statistics	Lookup Structure	Filter Structure	Format
0	Dst Vendor Address	0	24	0	16	0	0	0	[None]	Fig 4b	Fig 10.Idx 0	hex
1	Dst Station Address	24	24	0	16	0	0	0	[None]	[None]	Fig 10.Idx 1	hex
2	Src Vendor Address	48	24	0	16	0	0	0	[None]	Fig 4c	[None]	hex
3	Src Station Address	72	24	0	16	0	0	0	[None]	[None]	[None]	hex
4	Type	96	16	0	16	0	0	0	[None]	Fig 4d	[None]	hex

Fig. 4A

522. Applying control logic to the Ethernet Type field and associated lookup structure in Baker’s Figures 4a and 4d, respectively, the associated Protocol field found with the range containing 0x8888 value will be used to update the NextProtocol variable.⁶⁵⁹ Using Baker’s Figure 5C as an example, it may be seen how values may be used to continue parsing at different locations in the current protocol description.⁶⁶⁰ For example, a value of 0x01 would indicated “GP1”.⁶⁶¹

523. Thus, alone and/or in combination (by way of Riddle receiving Baker’s

⁶⁵⁸ Baker, Fig. 4A, 29:32-30:4.

⁶⁵⁹ Baker, 29:32-30:2.

⁶⁶⁰ Baker, 30:5-7.

⁶⁶¹ Baker, Fig. 5C.

PDFs) Riddle and Baker teach receiving a protocol description for a particular protocol at a particular layer level including if there is at least one child protocol of the protocol at the particular layer level, the one or more child protocols of the particular protocol at the particular layer level. It would have been obvious to a POSITA to modify Riddle in view of Baker since both involve layered models in data communications networks, which include child protocols. Baker relates to programmably configurable protocol descriptions for layered models such as the OSI model.⁶⁶² Further, Figure 1D of Riddle shows that the OSI model is prior art to the Challenged Patents.⁶⁶³

524. Further, Riddle discloses traffic classification based upon the Ethernet protocol and Baker discloses PDFs for the Ethernet protocol.⁶⁶⁴ Baker also discloses PDFs for Generic Protocols which could be the other protocols used in Riddle.⁶⁶⁵ Baker provides an explicit motivation to combine: “[I]t would be highly desirable

⁶⁶² Baker, Abstract, 1:22-25 (“Bridges operate at the data-link layer and routers at the network layer of the OSI reference model”), 4:14-21 (“[N]etwork interface in accordance with the present invention may be configured and reconfigured ... to accommodate substantial network modifications (for example, the use of different data transmission hardware, protocols or protocol suites) without necessitating substantial system changes”), 12:21-24 (“It will be appreciated by those skilled in the art that any possible organization of fields for any possible protocol specification is contemplated for the network interface system 10 of the present invention.”).

⁶⁶³ Riddle, Fig. 1D.

⁶⁶⁴ Riddle, Fig 1D, 7:47-53; Baker, Table 12, 21:32-22:1.

⁶⁶⁵ Baker, Table 13.

to be able to parse all protocols with a single configurable software (or hardware) module so that support for additional protocols could be added to a system without requiring substantial modification to the system or its control logic.”⁶⁶⁶ A POSITA would have realized that modifying Riddle’s traffic classification system with PDFs, as taught by Baker, provides the advantage of making Riddle’s network monitor more configurable without requiring substantial modification to Riddle. Thus, a POSITA would have been motivated to modify the teachings of Riddle to increase the ability to handle updated PDFs as taught by Baker.

525. Regarding the second part of the claim element, when Riddle or Baker receives packets, such as packets conforming to the OSI model, it has satisfied the portion of the claim element that requires “the packet including for any particular child protocol of the particular protocol at the particular layer level information at one or more locations in the packet related to the particular child protocol.” This portion of the claim element is further satisfied for the reasons disclosed above for ’725 claim elements 10.1, 10.3, 17.1, and 17.3.

- e. ’725 Claim Elements 10.4 and 17.4: “(ii) the one or more locations in the packet where information is stored related to any child protocol of the particular protocol”*

526. Riddle discloses this claim element alone and/or renders it obvious in view of Baker. For example, as previously discussed, Riddle discloses programming a

⁶⁶⁶ Baker, 3:3-8.

system to look for information in a packet relating to the TCP child protocol of IP, and the FTP and HTTP child protocols of TCP. Riddle additionally teaches that HTTP is a protocol built on top of the TCP protocol. Also, Riddle explicitly states that classification extends to examination of the data contained in the flow's packets. Additionally, PointCast traffic, which uses HTTP as a transport mechanism, "can be distinguished by looking into the data for a signature content header in the get request."⁶⁶⁷ Further, Baker identifies specific byte locations in Ethernet and GP PDFs that relate to the child protocols.⁶⁶⁸ Moreover, the Board previously found Baker taught this claim element.⁶⁶⁹

527. The analysis above for '725 claim elements 10.3 and 17.3 show Riddle and Baker disclose programming a system to identify the one or more locations in the packet where information is stored related to any child protocol of the particular protocol. I incorporate by reference the disclosure for '725 claim elements 10.3 and 17.3 as if fully set forth herein. For example, that discussion shows specific byte locations in Ethernet and GP PDFs that relate to the child protocols.

⁶⁶⁷ Riddle, 11:50-63.

⁶⁶⁸ Baker, Tables 12-13, Frames 1-2.

⁶⁶⁹ Ex. 1062 (IPR2017-00863 Institution Decision), 15-17.

f. '725 Claim Elements 10.5 and 17.5: “(iii) if there is at least one protocol specific operation to be performed on the packet for the particular protocol at the particular layer level, the one or more protocol specific operations to be performed on the packet for the particular protocol at the particular layer level”

528. Riddle discloses this claim element alone and/or renders it obvious in view of Baker. Moreover, the Board previously found Baker taught this claim element.⁶⁷⁰

529. The '725 Patent states that “protocol specific operations may include parsing and extraction operations to extract identifying information. The protocol specific operations may also include state processing operations defined for a particular state of a conversational flow of the packet.”⁶⁷¹ The '725 Patent describes that its protocol description language (“PDL”) files contain information on “*how to interpret* header information, *how to determine* from the packet header information the protocols at the next layer, and *what information to extract* for the purpose of identifying a flow, and ultimately, applications and services.”⁶⁷² The '725 Patent includes several exemplary PDL files for protocols, such as Ethernet, IEEE 802.2, IEEE 802.3, TCP, UDP, RPC, NFS, and HTTP, include information describing the

⁶⁷⁰ Ex. 1062 (IPR2017-00863 Institution Decision), 17-18.

⁶⁷¹ '725 Patent, Abstract, 2:21-31.

⁶⁷² '725 Patent, 9:29–35.

protocol and what information may be parsed or extracted.⁶⁷³ The PDL files provided in the '725 Patent do not include the instructions for parsing or extracting.⁶⁷⁴

530. Riddle discloses the protocol specific operations in at least two ways. First, Riddle discloses performing parsing and extraction operations to extract identifying information. In order to classify traffic, Riddle explicitly discloses parsing/extraction operations on flows using the method outlined below in annotated Figure 4A.⁶⁷⁵

⁶⁷³ '725 Patent, 51:5-55:25, col. 77-col.93.

⁶⁷⁴ '725 Patent, 51:5-55:25, col. 77-col.93.

⁶⁷⁵ Riddle, 12:42-53, Fig. 4A (Step 402's "parse flow specification from a packet of the flow").

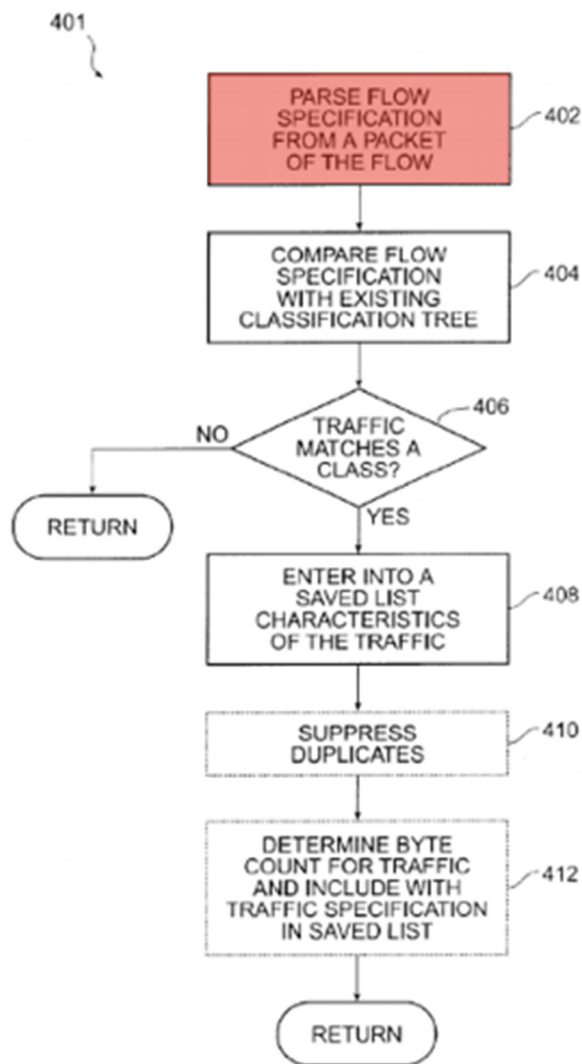


FIG. 4A

531. Further, as discussed above regarding '099 claim element 1.7, Riddle discloses performing state operations. I incorporate by reference that discussion as if fully set forth herein. And a POSITA would have understood that after parsing step 402, Riddle has extracted identifying characteristics to perform steps 404 (“compare flow specification with existing classification tree”), 406 (“traffic matches a class?”), and 408 (“enter into a saved list characteristics of the traffic”). Similarly,

a POSITA would have understood that Riddle has extracted identifying characteristics to perform Figure 4B's step 422 ("saved traffic known well?"), step 423 ("saved traffic a server at unregistered IP port?"), and step 426 ("saved traffic belongs to a service aggregate?").⁶⁷⁶

532. Like Riddle, Baker also discloses receiving configurable protocol descriptions that include information to perform parsing and extraction operations. For example, Baker teaches:

Parsing, the process wherein network frames are broken up into their individual protocols and fields, is necessary for filtering with offsets relative to protocol headers, gathering field based statistics, generating network traffic, routing data frames, verifying field values, and displaying network frames in human readable form. In conventional systems, the parsing process has an overall structure which incorporates control logic for each supported protocol. Therefore, additional control logic must be developed when support for a new protocol is added to a conventional system. As the development of additional control logic, whether implemented in hardware or software, may be both time consuming and expensive, *it would be highly desirable to be able to parse all protocols with a single configurable software (or hardware) module so that support for additional protocols could be added to a system without requiring substantial modification to the system or its control logic.*⁶⁷⁷

⁶⁷⁶ Riddle, 13:36-62, Fig. 4B.

⁶⁷⁷ Baker, 2:28-3:8.

533. Baker continues:

The present invention is directed *to improved systems and methods for parsing*, filtering, generating and analyzing data (or frames of data) transmitted over a data communications network. In one particularly innovative aspect of the present invention, a single logic control module, which may be implemented in hardware or software, is utilized to perform any of a number of data manipulation functions (for example, *parsing*, filtering, data generation or analysis functions) based upon one or more programmably configurable protocol descriptions which may be stored in and retrieved from an associated memory.⁶⁷⁸

534. Baker additionally states:

The system of the present invention also preferably includes *logic for extracting field values from particular network frames*, performing validation and error checking, and *making parsing decisions based upon field values and information in the programmably configurable protocol descriptions*.⁶⁷⁹

535. Baker further states:

Accordingly, it is an object of the present invention to provide an improved system for network analysis wherein the system may *determine which protocols and which protocol fields exist in a network frame (also referred herein as parsing) using common control logic combined with configurable protocol descriptions*.⁶⁸⁰

⁶⁷⁸ Baker, 3:32-4:6.

⁶⁷⁹ Baker, 5:4-9.

⁶⁸⁰ Baker, 7:1-6.

536. Baker provides flow charts for its parsing and extraction operations:

The flow chart shown in FIG. 11 outlines ParseFrame control logic in accordance with the present invention and shows how successive protocol headers are parsed, and how remaining information is parsed as application data and frame pad. The flow chart in FIG. 12 outlines ParseProtocol control logic in accordance with one form of the present invention and shows how fixed and optional fields may be parsed in a selected protocol. The flow chart shown in FIG. 13 outlines ParseFields control logic in accordance with the present invention and shows how decisions are made and operations performed on extracted field values.⁶⁸¹

537. Each of Baker's Figures 11, 12, and 13 show recursive parsing and extraction operations that will proceed until all appropriate information has been parsed.⁶⁸² For example, Baker describes the process outlined in Figure 13:

Referring back to the ParseFields control logic shown in FIG. 13, the ParseFields control logic parses the fields *in each protocol header* contained in a particular network frame by using field values obtained in accordance with information specified in associated protocol descriptions. The ParseFields control logic is applied for each protocol description required for a particular network frame. *If the ParseFields control logic were applied to the exemplary frame, "Frame (1)," described*

⁶⁸¹ Baker, 26:27-27:3, Figs. 11, 12, and 13.

⁶⁸² Baker, Fig. 11 (104, 106, 108, 128), (Fig. 12 (decision 154 "all bits parsed"), Fig. 13 (206 "another field to parse")).

*above, the network interface system 10 of the present invention would apply the ParseFields control logic with the protocol descriptions for the Ethernet protocol shown in Table 12, the GP shown in Table 13, and an unspecified Data protocol description.*⁶⁸³

538. Baker's claims further confirm that the received PDFs include parsing operations. For example, claim 3 of Baker shows that the programmably configurable protocol description, *i.e.*, PDF, contain the characteristics necessary to perform the parsing operations of a particular protocol. Baker's claim 3 requires:

A machine implemented process for parsing data transmitted over a data communications network, said process comprising the steps of: *storing at least one programmably configurable protocol description in a memory*, said at least one programmably configurable protocol description comprising a protocol control record and at least one field sub record for defining a plurality of characteristics of said data transmitted over said data communications network; *retrieving said at least one protocol description* from said memory; and providing said at least one protocol description to a logic control module, said logic control module, *upon receiving said at least one protocol description, being configured to parse data received from said data communications network based upon said characteristics defined by said protocol description.*⁶⁸⁴

539. Further, Baker's claim 9 requires:

⁶⁸³ Baker, 30:10-23.

⁶⁸⁴ Baker, claim 3.

A method for parsing data transmitted over a data communications network, said method comprising the steps of:

- storing in a first memory a plurality of programmably configurable protocol descriptions, said programmably configurable protocol descriptions defining a plurality of characteristics of said data transmitted over said data communications network;
- storing in a second memory a program for controlling a data parsing function to be executed by a processing unit, said program including instructions for causing said processing unit *to selectively retrieve at least one of said programmably configurable protocol descriptions from said first memory and to vary the execution of said data parsing function based upon said at least one retrieved protocol description*;
- delivering to said processing unit said program for controlling said data parsing function;
- enabling said processing unit to execute said data parsing function; and
- delivering to said processing unit said data transmitted over said data communications network.⁶⁸⁵

540. Thus, it would have been obvious to a POSITA to modify Riddle in view of Baker. Baker describes that Baker's invention is used with network monitors, such as those disclosed in Riddle. Baker's invention "may be employed in any system where it is useful to be able to examine and perform various operations

⁶⁸⁵ Baker, claim 9.

on contiguous bit-fields in data structures, wherein each data structure is composed of predefined fields of one or more contiguous bits.”⁶⁸⁶ Baker also discloses that his invention “may be incorporated in a network device, such as a network analyzer, bridge, router, or traffic generator, including a CPU and a plurality of input devices, storage devices, and output devices.”⁶⁸⁷ These are the same type of devices Riddle’s invention is used on.⁶⁸⁸ And a POSITA would have understood that the modifying Riddle’s monitor to use PDF files as taught by Baker’s amounts to nothing more than combining known prior art technologies used in their ordinary and predictable manner to organize Riddle’s protocol specific operations.

541. Additionally, Baker provides an explicit motivation to combine: “[I]t would be highly desirable to be able to parse all protocols with a single configurable software (or hardware) module so that support for additional protocols could be added to a system without requiring substantial modification to the system or its control logic.”⁶⁸⁹ A POSITA would have recognized that modifying Riddle with the teachings of Baker’s PDF files provides the advantage of

⁶⁸⁶ Baker, 6:18-22.

⁶⁸⁷ Baker, 4:27-30.

⁶⁸⁸ Riddle, 5:55-57, claim 8, Figs. 1A, 1B, 1C.

⁶⁸⁹ Baker, 3:3-8.

making Riddle's network monitor more configurable without requiring substantial modification to Riddle. Thus, a POSITA would be motivated to modify the teachings of Riddle to increase the ability to handle updated PDFs as taught by Baker.

- g. *'725 Claim Elements 10.6 and 17.6: "(c) performing the protocol specific operations on the packet specified by the set of protocol descriptions based on the base protocol of the packet and the children of the protocols used in the packet"*

542. Riddle alone discloses this claim element and/or renders it obvious in view of Baker. Moreover, the Board previously found Baker taught this claim element.⁶⁹⁰

543. The combination of Riddle and Baker renders obvious performing protocol specific operations on the packet specified by the set of protocol descriptions based on the base protocol of the packet and the children of the protocols used in the packet. Riddle's methods include "applying individual instances of traffic classification paradigms to packet network flows based on *selectable information* obtained from a plurality of layers of a multi-layered communication protocol in order to define a characteristic class, then mapping the flow to the defined traffic

⁶⁹⁰ Ex. 1062 (IPR2017-00863 Institution Decision), 19.

class.”⁶⁹¹ In Riddle, traffic may be classified at any level based upon selectable information, such as “any level of the IP protocol as well as for other non-IP protocols.”⁶⁹²

544. In order to classify traffic, Riddle describes parsing/extraction operations on flows using the method outlined in Figures 4A and 4B.⁶⁹³ After parsing step 402, a POSITA would have understood that Riddle’s monitor uses extracted identifying characteristics to compare the flow information to classification tree (step 404), determine whether flow matches a traffic class (step 406), and enter flow characteristics into a saved list (step 408). Similarly, a POSITA would have understood that Riddle’s monitor uses extracted identifying characteristics to determine whether the save class is well known (step 422), whether the saved traffic is a server connection port of an unregistered IP port (step 423), and whether the traffic class matches all components of a service aggregate flow (step 426).

545. Further, a POSITA would have understood that the combination of Riddle and Baker would use Baker’s PDFs in Riddle as discussed above with respect to ’725 claim elements 10.2 and 17.2. In doing so Riddle’s parsing operation, shown in steps 402, and extraction operation, such as steps 404, 406, and 408, would use

⁶⁹¹ Riddle, Abstract, 4:10-15.

⁶⁹² Riddle, 8:47-9:27, 10:57-11:9.

⁶⁹³ Riddle, 12:42-53, Fig. 4A (Step 402, 404, 406, 408); 13:36-62, Fig. 4B (Steps 422, 423, 426).

the parsing operations contained in Baker's PDFs.

546. Baker's parsing control logic performs protocol specific operations, e.g., parsing/extraction, on the packet as specified by the set of PDFs based on the base layer protocol (e.g., Ethernet) of the packet and the children of the protocols used in the packet (e.g., GP and others). For example, Baker describes part of the logic included in Figure 13:

Referring back to the ParseFields control logic shown in FIG. 13, the ParseFields control logic parses the fields *in each protocol header* contained in a particular network frame by using field values obtained in accordance with information specified in associated protocol descriptions. *The ParseFields control logic is applied for each protocol description required for a particular network frame. If the ParseFields control logic were applied to the exemplary frame, "Frame (1)," described above, the network interface system 10 of the present invention would apply the ParseFields control logic with the protocol descriptions for the Ethernet protocol shown in Table 12, the GP shown in Table 13, and an unspecified Data protocol description.*⁶⁹⁴

⁶⁹⁴ Baker, 30:10-23.

- h. '725 Claim Element 10.7: “wherein the protocol specific operations include one or more parsing and extraction operations on the packet to extract selected portions of the packet to form a function of the selected portions for identifying the packet as belonging to a conversational flow”*

547. Riddle discloses this element. Riddle’s “service aggregates” identifies conversational flows for applications using more than one connection by creating a function based on extracted information (i.e., signature), such as looking for a PointCast signature (“/FIDO-1/”) in the header of an HTTP get request.⁶⁹⁵ As shown above in ’725 claim elements 10.3-10.5 and 17.3-17.5, the combination of Riddle and Baker renders obvious the protocol specific operations include one or more parsing and extraction operations on the packet to extract selected portions of the packet.

548. Riddle also discloses extracting selected portions of the packet to form a function of the selected portions for identifying the packet as belonging to a conversational flow in at least two independent ways – through the disclosure of “service aggregates,” and through the disclosure of classifying PointCast traffic.

- (1) Riddle’s service aggregates teach the claimed “to form a function of the selected portions for identifying the packet as belonging to a conversational flow”

549. As discussed with respect to ’099 claim element 1.4 in Section VII.A.2.e,

⁶⁹⁵ Riddle, 11:10-2, 11:47-67, 15:28-31; Ex. 1016 (’903 Provisional), 7:16-25.

Riddle teaches forming a function of selected packet portions for identifying that packet as belong to a “conversational flow” by classifying separate packet flows by a common “service aggregate” traffic class for applications using multiple flows between a client and a server.⁶⁹⁶ I incorporate by reference that discussion as if fully set forth herein. Riddle, for example, discloses classifying based on a service aggregate flow that links separate application-specific conversations:

A service aggregate is provided for certain applications that use more than one connection in a particular conversation between a client and a server. For example, an FTP client in *conversation* with an FTP server employs a command channel and a transfer channel, which are distinct TCP sessions on two different ports. In cases where two or three TCP or UDP sessions exist *for each conversation* between one client and one server, it is useful to provide a common traffic class i.e., the service aggregate, containing *the separate conversations*. In practice, these types of *conversations* are between the same two hosts, but use different ports. According to the invention, a class is created with a plurality of traffic specifications, each matching various component *conversations*.⁶⁹⁷

550. The '864 Provisional further illustrates how a POSITA would have understood Riddle's service aggregates relate multiple connection flows based on an

⁶⁹⁶ Riddle, 11:10-36, 12:64-13:62, claim 2, Figs. 2A-2B, 4B; RFC765, 6-7; Ex. 1023 (German Court Translation), 35-36.

⁶⁹⁷ Riddle, 11:10-23.

FTP application's specific software program activity:

[T]he concept of “service aggregates” (service groups) [is] different traffic types that are associated together (ex. FTP has one stream that it uses to exchange commands and responses, and a second that the data files are actually sent over). Whenever we recognize the signature of one of these types of traffic, we create a traffic class (or class hierarchy) that can match all the components of the aggregate. This bundling is mainly a convenience to the user, makes it clearer what's going on, but also permits you to get group counts of all the parts that make up what the user thinks as the service.⁶⁹⁸

551. As I detail above in Section IV.A.6, Riddle describes creating traffic classes based on data relating to RTP and RTSP.⁶⁹⁹ A POSITA would have understood RTP and RTSP are analogous to FTP, and that those protocols use a separate control flow with one or more linked dataflows.⁷⁰⁰ As such, a POSITA would have appreciated this is another example of a common “service aggregate” traffic class for applications using multiple flows between a client and a server.⁷⁰¹

552. As such, Riddle's identification of service aggregates meets the claimed extracting selected packet portions “to form a function of the selected portions for identifying the packet as belonging to a conversational flow.”

⁶⁹⁸ '864 Provisional, 69.

⁶⁹⁹ Riddle, 12:1-12.

⁷⁰⁰ Ex. 1045, 4-5 (RFC1889 illustrating well-known RTP information); Ex. 1046, 9-10 (RFC2326 illustrating well-known RTSP information).

⁷⁰¹ Riddle, 11:10-23.

- (2) Riddle’s PointCast flows teach the claimed “to form a function of the selected portions for identifying the packet as belonging to a conversational flow”

553. As discussed with respect to ’099 claim element 1.4 in Section VII.A.2.e, Riddle teaches forming a function of selected packet portions for identifying that packet as belong to a “conversational flows” by classifying separate packet flows as being PointCast traffic.⁷⁰² I incorporate by reference that discussion as if fully set forth herein. Riddle, for example, discloses examining packet portions for URLs that begin with /FIDO-1/ to classify flows as corresponding to PointCast applications.⁷⁰³ And as discussed, Patentee’s ’903 Provisional acknowledges it was known that separate PointCast flows include a signature specific to PointCast traffic and those flows are associated with the PointCast Network application.⁷⁰⁴

554. As such, Riddle’s identification of PointCast flows meets the claimed extracting selected packet portions “to form a function of the selected portions for identifying the packet as belonging to a conversational flow.”

⁷⁰² Riddle, 11:47-67, 15:28-31; Ex. 1016 (’903 Provisional), 7:16-25; Ex. 1032 (Wall Street Journal PointCast article), 1; Ex. 1033 (Computer World PointCast article); Ex. 1034 (Christian Science Monitor PointCast article); Ex. 1036 (’558 Patent), 33:28-44.

⁷⁰³ Riddle, 11:47-67.

⁷⁰⁴ Ex. 1016 (’903 Provisional), 7:16-25, 28:22-24.

- i. *'725 Claim Element 17.7: “wherein the packet belongs to a conversational flow of packets having a set of one or more states, and wherein the protocol specific operations include one or more state processing operations that are a function of the state of the conversational flow of the packet, the state of the conversational flow of the packet being indicative of the sequence of any previously encountered packets of the same conversational flow as the packet”*

555. Riddle discloses this claim element. Riddle discloses “conversational flows” in two independent ways: (1) Riddle identifies “conversational flow[s]” through Riddle’s disclosure of “service aggregates,” and (2) Riddle identifies “conversational flow[s]” through Riddle’s ability to classify PointCast traffic. I have described both of these “conversational flow[s]” above in ’725 claim element 10.7. I incorporate that discussion as if fully set forth herein.

556. Riddle discloses that these conversational flows have one or more states (e.g., TCP session for FTP-cmd followed by TCP session for FTP-data; PointCast HTTP connection, get request, “inbound” traffic), and that classification includes performing one or more state processing operations that are indicative of any previously encountered packets for the same conversational flow.⁷⁰⁵ For example, Riddle discloses keeping track of when traffic with the same identifying characteristics was last encountered.⁷⁰⁶

⁷⁰⁵ Riddle, 9:20-22, 11:10-23, 11:48-67, 13:10-22, 14:63.

⁷⁰⁶ Riddle, 12:56-57.

557. Riddle further discloses performing protocol specific operations that include five different state processing operations that are a function of the state of the conversational flow of the packet.

558. *First*, the Challenged Patents' claims confirm that state operations include "searching for one or more *patterns* in the packet portions."⁷⁰⁷ Riddle explicitly states it searches for *patterns*:

a traffic class could be created such as all URIs matching "*.html" for all servers, or all *URI patterns* matching "*.gif" for server X, or for access to server Y with URI pattern "/sales/*" from client Z, wherein '*' is a wildcard character, i.e., a character which matches all other character combinations. Traffic class attributes left unspecified will simply match any value for that attribute. For example, a traffic class that accesses data objects within a certain directory path of a web server is *specified by a URI pattern* of the directory path to be managed, e.g. "/sales/*".⁷⁰⁸

559. Riddle further discloses that "Web traffic may also be classified by HTTP header types such as Content-Type (MIME type) or User-Agent."⁷⁰⁹ Moreover, in Riddle's preferred embodiment, classification extends to examination of the data contained in a flow's packets.⁷¹⁰ Accordingly, at block 208 or 212, an operation

⁷⁰⁷ '646 claim 13, '789 claim 27; '789 claim 17, '789 claim 46 ("searching the parser record for the existence of one or more reference strings").

⁷⁰⁸ Riddle, 8:67-9:11.

⁷⁰⁹ Riddle, 9:24-26.

⁷¹⁰ Riddle, 11:48-49.

may be performed that searches for one or more patterns, such as searching for URI patterns matching “*.html”, “*.gif”, or a directory path, e.g. “/sales/*”.

560. *Second*, ’789 Patent claim 47 confirms that creating a new flow-entry for future packets to be identified with the flow is a state operation.⁷¹¹ When Riddle classifies flows it creates a new flow entry and suppresses duplicate entries:

In a decisional step 406, a determination is made if traffic matches one of the classes being classified. If this is so, then in a step 408, an entry is made in a list of identifying characteristics, such as protocol type (SAP), IP protocol number, server port, traffic type if known, MIME type, a time of occurrence of the traffic. In an optional step 410, duplicate instances having the same identifying characteristics are suppressed, in favor of keeping a count of the duplicates and a most recent time traffic with these identifying characteristics was encountered.⁷¹²

561. Accordingly, at Riddle’s blocks 206, 208, 210, and/or 212 a state operation is performed every time a new flow entry is created per step 408 of Figure 4.

562. *Third*, state operations are performed at blocks 206 or 210 because they relate to FTP applications which Riddle proposes handling as “service aggregates.”⁷¹³ Riddle further discloses using subclassification for FTP:

⁷¹¹ ’789 claim 47 (“wherein one of the state operations specified for at least one of the states includes *creating a new flow-entry* for future packets to be identified with the flow, the new flow-entry including identifying information for future packets to be identified with the flow-entry.”).

⁷¹² Riddle, 12:48-57.

⁷¹³ Riddle, 11:10-23.

Subclassification of traffic into a tree is performed by matching the hosts and then searching for particular services. Traffic *specifications are aggregate kinds of traffic for a traffic class, e.g., different components of FTP may reside under class FTP. Subclassification is performed by first locating a class that matches, and then performing finer grade matchings.* Processing commences with a decision on what traffic is to be subclassified. A marker is placed in the match_all default node so that when match processing reaches the marker, the autoclassification processing depicted in flowchart 403, determines that it has not found an existing class for the traffic being classified.⁷¹⁴

563. Thus a POSITA would have understood that Riddle's subclassification takes place at block 208 or 212 because these blocks relate to FTP applications. In order to perform subclassification, an operation would be performed to determine if the flow was for an FTP command or data channel.

564. *Fourth*, '751 claim 16 shows that determining metrics related to the flow is a state operation.⁷¹⁵ Further, claims 11-13 of the '751 Patent clarify that reporting metrics can be part of a state operation. These claims require:

11. A method according to claim 10, further including *reporting one or more metrics* related to the flow of a flow-entry from one or more of the statistical measures in the flow-entry.

12. A method according to claim 11, *wherein the reporting* is carried out

⁷¹⁴ Riddle, 11:24-36.

⁷¹⁵ '751 claim 16 ("wherein one or more metrics related to the state of the flow are determined as part of the state operations specified for the state of the flow").

from time to time, and wherein the one or more metrics are base metrics related to the time interval from the last reporting time.

13. A method according to claim 12, wherein *the reporting is part of the state operations* for the state of the flow.⁷¹⁶

565. Riddle discloses determining and reporting metrics related to flows such as byte count, most hits, time most recently seen, most data transferred, moving average, bytes per second, most recently used, most hits, and number of bytes received.⁷¹⁷ For example, Riddle states “[i]n an optional step 412, a byte count of traffic of this type has been detected is included.”⁷¹⁸ And Riddle states that displaying results to a user include:

The list may be sorted by any well-known criteria such as: 1) most “hits” during a recent interval, 2) most recently-seen (most recent time first), 3) most data transferred (bytes/second) during some interval, or a moving average. The user may choose an interval length or display cutoff point (how many items, how recent, at least B bytes per second, or other thresholds).⁷¹⁹

566. Further, Riddle states “[i]n a related embodiment in place of step 425, a display of traffic classes, sorted by most recently used, most hits, number of bytes received during any interval, which is determined by a plurality of time stamps, is

⁷¹⁶ ’751 Patent, claims 11-13.

⁷¹⁷ Riddle, 12:53-13:8, 14:1-5.

⁷¹⁸ Riddle, 12:57-59.

⁷¹⁹ Riddle, 13:1-8.

available on demand to a network manager.”⁷²⁰

567. *Fifth*, the Challenged Patents’ claims show that updating a flow entry is a state operation.⁷²¹ Regarding Figure 4A, Riddle states:

In an optional step 410, duplicate instances having the same identifying characteristics are suppressed, in favor of keeping a count of the duplicates and *a most recent time traffic with these identifying characteristics was encountered*. In an optional step 412, *a byte count of traffic of this type has been detected is included*.⁷²²

568. In Riddle, the state of the conversational flow of the packet is also indicative of the sequence of any previously encountered packets of the same conversational flow as the packet. The state of the flow is determined by the relationship of packets and the entire conversational flow.⁷²³ The Challenged Patents are clear that the state of the flow includes “parameters such as the time, length of the conversational flow, data rate, etc.”⁷²⁴ Accordingly, Riddle discloses determining the state

⁷²⁰ Riddle, 14:1-5.

⁷²¹ E.g., ’646 claim 15 (“wherein the state operations include updating the flow-entry, including identifying information for future packets to be identified with the flow-entry”); ’751 claim 14, ’789 Patent claims 15, 30, 45.

⁷²² Riddle, 12:53-59, Fig. 4A (Step 402’s “parse flow specification from a packet of the flow”; Step 404’s “compare flow specification with existing classification tree”; Step 406’s “traffic matches a class?”; Step 408’s “enter into a saved list characteristics of the traffic”; Step 410’s “suppress duplicates”; Step 412’s “determine byte count for traffic and include with traffic specification in saved list”).

⁷²³ ’099 Patent, 5:27-34.

⁷²⁴ ’099 Patent, 5:27-34.

of the flow, at least, by determining metrics such as a count of the duplicates, the most recent time traffic with the same identifying characteristics was encountered, and a byte count of the detected traffic.⁷²⁵

569. Further, the '903 Provisional, which the '725 Patent incorporates-by-reference, acknowledges that an exemplary “pending” state occurs when the state processor is to perform a string search on a packet, and that an “identified state” that occurs when the protocol identifier (i.e., the application program) for the flow is known.⁷²⁶ Further, the '903 Provisional contemplates a “single-packet protocol recognition of flows” where, for example, “only a single state transition has to occur to be able to pinpoint the application that produced the packet.”⁷²⁷

570. In Riddle, since traffic class membership is hierarchical, “a flow may be classified by a series of steps through a traffic class tree, with the last step (i.e., at the leaves on the classification tree) mapping the flow to a policy.”⁷²⁸ “[T]he clas-

⁷²⁵ Riddle, 12:53-13:8, 14:1-5, Fig. 4A (Step 410’s “suppress duplicates”; Step 412’s “determine byte count for traffic and include with traffic specification in saved list”).

⁷²⁶ Ex. 1016 ('903 Provisional), 104:1-2 (“Once a Flow Record enters the “MSG Pend 1” state, the next packet will cause the state processor to perform a string search for a group of substrings.”), 76:16-20 (“The Lookup and Update Engine also outputs ... the protocol identifier for the flow ... the corresponding flow is in the IDENTIFIED state.”), 103:5-106:8.

⁷²⁷ Ex. 1016 ('903 Provisional), 12:22-26, 31:33-35.

⁷²⁸ Riddle, 9:20-25.

sification process checks at each level if the flow being classified matches the attributes of a given traffic class.”⁷²⁹ Based on this recursive process, the state of the flow (e.g., a “branch” attribute match) determines the next classification check (e.g., the “leaf” attribute). For example, a flow containing an HTTP header will be identified as an “HTTP” flow in a first state. Further investigation into the flow based on that identification includes a search for “/FIDO-1/” in the payloads of subsequent packets. If found, the flow enters a new state and Riddle identifies this flow as part of the PointCast conversational flow. And future packets within the flow are automatically considered as “PointCast” since the state of the flow is “identified.” As such, a POSITA would have understood that Riddle identifies future flows with similar paths through the classification tree.

571. Additionally, Baker performs protocol specific operations that include state operations. For example, and as shown below, Baker’s Figure 11 describes frame parsing control logic (i.e., how successive protocol headers are parsed), and parsing the remaining information as application data and frame pad.⁷³⁰

572. For example, Baker discloses:

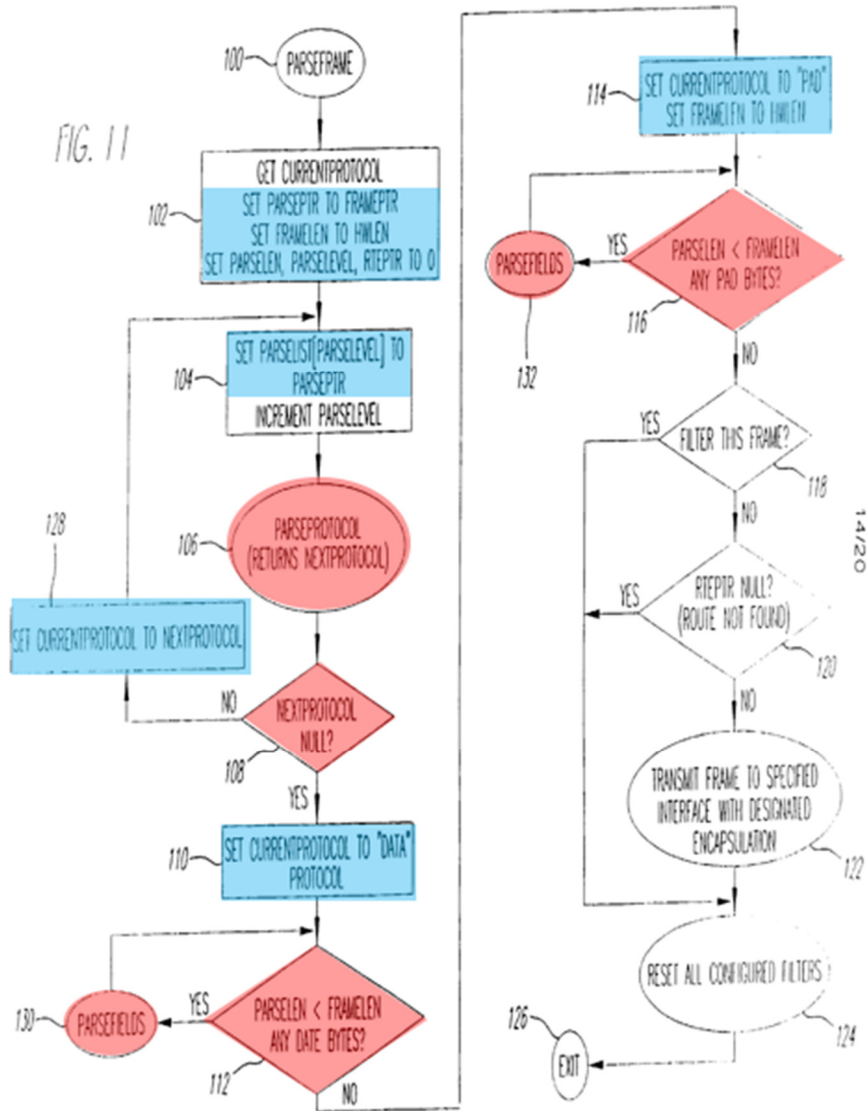
The ParseFrame control logic systematically parses through each network frame (at 104 to 108 and 128) until all known protocol headers

⁷²⁹ Riddle, 9:28-41.

⁷³⁰ Baker, 26:26-32, Fig. 11.

have been parsed. Any remaining frame bits are parsed as application data (at 110, 112 and 130) and/or pad data (at 114, 116 and 132).⁷³¹

573. Figure 11 shows setting states (blue boxes) followed by specific operations (red boxes):



574. Baker explains the processing in Fig. 11:

⁷³¹ Baker, 36:28-36, 37:17-38:24, Fig. 11.

Referring again to Fig. 11, once the system has received a network frame (at 100), defined by an interface number (SrcIntf), a frame location (FramePtr) and a hardware length (HwLen), the frame is resolved into its protocol and field components using the system of the present invention.

Using the exemplary frame, “Frame (2),” described above as an example, the system (at 102) in Fig. 11 would obtain from the receiving network interface device SrcIntf, the receiving interface number, FramePtr, a pointer length. to the frame, and HwLen, the hardware frame length. The hardware length of frame (2) is 480 bits. ParseLen, the number of bits in the frame that have been parsed, ParseLvl and CurField, the index of the protocol field being processed are reset to zero, and CurrentProtocol, is set up with the initial protocol description structure of the receiving interface number which for frame (2) is the Ethernet Protocol description 35 defined in Figs. 4 - 4d. FrameLen is set to the value of HwLen, and ParsePtr is set to the value of FramePtr. Each field in the Ethernet Protocol description as shown in Fig. 4a is parsed (at 106) using the ParseProtocol control logic shown in Fig. 13.

The ParseProtocol control logic updates ProtoParseLen, the number of bits parsed in the CurrentProtocol, HeaderLen, the protocol header size determined during parsing, and returns NextProtocol, a reference to the next applicable protocol description structure to use during parsing. ParseProtocol also updates ParsePtr and ParseLen by adding ProtoParseLen to them. If NextProtocol is NULL, the remaining frame bits will be treated as Data and/or Pad bits.

After the Ethernet protocol fields in frame (2) are parsed (at 106) by the ParseProtocol control logic shown in Fig. 13, HeaderLen, ParseLen and ProtoParseLen will be 112 bits, NextProtocol will refer to the GP shown in Figs. 5-S(e), and ParsePtr will point at the start of line 2 in frame (2). CurrentProtocol will be updated with the NextProtocol value of GP (at 130) and the GP fields in frame (2) are parsed (at 106) by the ParseFields control logic shown in Fig. 13, which will update HeaderLen and ProtoParseLen to be 160 bits, and return NextProtocol as NULL. ParsePtr will point at the start of line 3 in frame (2), and ParseLen will be updated to 272 bits.⁷³²

575. Moreover, Baker's Figure 12 shows protocol-specific option operations.⁷³³

If the CurrentProtocol supports optional fields, then those frames are parsed using the ParseFields control logic.⁷³⁴

⁷³² Baker, 37:17-38:24.

⁷³³ Baker, 37:1-16, 38:25-39:19, Fig. 12.

⁷³⁴ Baker, 38:25-33.

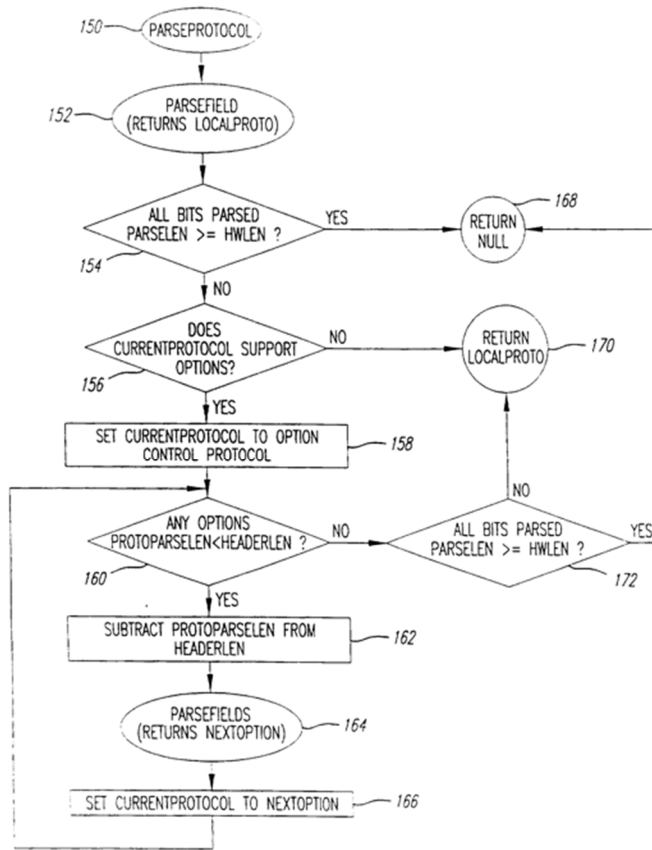
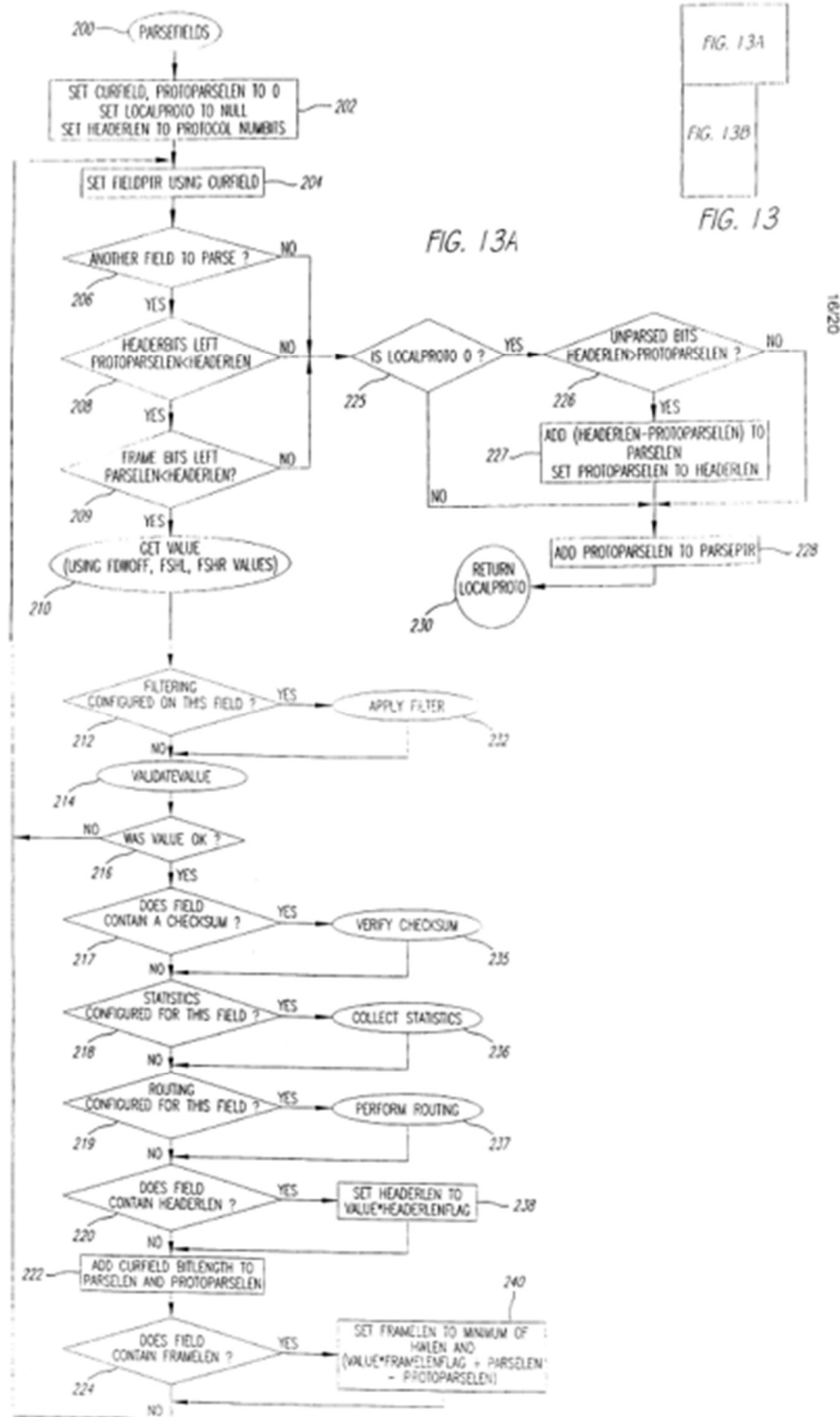


FIG. 12

576. Baker's ParseFields control logic is exemplified in combined Figure 13

(below):



577. Baker further describes the ParseFields logic as:

[T]he ParseFields control logic parses the fields in each protocol header contained in a particular network frame by using field values obtained

in accordance with information specified in associated protocol descriptions. The ParseFields control logic is applied for each protocol description required for a particular network frame. If the ParseFields control logic were applied to the exemplary frame, "Frame (1)", 11 described above, the network interface system 10 of the present invention would apply the ParseFields control logic with the protocol descriptions for the Ethernet protocol shown in Table 12, the GP shown in Table 13, and an unspecified Data protocol description.

The ParseFields routine is entered (at 200) with 25 ParsePtr pointing at the start of a protocol header in a particular network frame and Current Protocol set to an appropriate protocol description. Parsing starts at Protocol bit and field zero when CurField and ProtoParseLen are cleared (at 202), also, HeaderLen is set to the configured protocol control record NumBits value, and LocalProto, the local next protocol return value variable is cleared. Using the Ethernet protocol description shown in Fig. 4 as an example, HeaderLen would be set to 112 bits.

The control loop (at 204 through 224) continues until the last field has been parsed (at 206), all bits in the header have been parsed (at 208), or all bits in the frame have been parsed (at 209).

For each field a value is retrieved by the system (at 210). If there is a filter criteria for the field it is applied (at 232) by the ApplyFilter control logic. The System Filter Status is set to FILTER FRAME and NextCriteriaIndex is set to zero for every configured filter channel prior to the start of frame processing and after each frame is processed (at 124 in Fig. 11).⁷³⁵

⁷³⁵ Baker, 30:11-31:9

578. The ParseFields logic provides protocol-specific operations and sets states. The ParseFields logic applies, for example, the protocol descriptions for the Ethernet protocol, the GP protocol, other unspecified protocols, or “application data”.⁷³⁶ The ParseFields logic includes filter logic to identify packets based on a filter criteria, which could be implemented using an offset, bit-length, comparison value, and/or mask values.⁷³⁷ While Baker specifically refers to two different states based on a filter match, Baker contemplates additional states.⁷³⁸ Therefore, a POSITA would have understood that Baker’s filters could be used to identify additional protocols or applications based on comparison values to application data, and set the flow states accordingly.

579. In view of Baker, it would have been obvious to a POSITA to use flexible parsing operations with Riddle’s “service aggregates” and URI searches. As Baker explains: “[I]t would be highly desirable to be able to parse all protocols with a single configurable software (or hardware) module so that support for additional protocols could be added to a system without requiring substantial modification to the system or its control logic.”⁷³⁹ Accordingly, a POSITA would have been motivated to place Riddle’s aggregation methods into a single, configurable

⁷³⁶ Baker, 30:17-23; 36:34-35.

⁷³⁷ Baker, 2:11-17.

⁷³⁸ Baker, 17:13-15 (“These states *may* include: PASS_FRAME (accept this frame) and FILTER FRAME (discard this frame).”).

⁷³⁹ Baker, 3:3-8.

system, as taught by Baker, because this would allow Riddle's classification system to easily update protocol descriptions.

580. For the above reasons, it is my opinion that Riddle in view of Baker renders obvious claim 10 of the '725 Patent.

3. Dependent '725 Claim 12

581. Riddle discloses all the limitations of this claim. Claim 12 depends from independent claim 10 and recites:

A method according to claim 10, wherein which protocol specific operations are performed is step (c) depends on the contents of the packet such that the method adapts to different protocols according to the contents of the packet.

582. As I showed above, Riddle teaches protocol specific operations performed in '725 claim element 10.6 in two ways: (i) performing parsing and extraction operations to extract identifying information; and (ii) performing state operations.

583. At least the state operations disclosed in Riddle depend on the contents of the packet and therefore the method adapts to different protocols according to the contents of the packet. For example, Riddle discloses the following state processing operations:

- Searching for patterns/referencing strings;
- Creating a new flow-entry for future packets to be identified with the flow;
- Performing state operations related to "service aggregates";

- Determining metrics; and
- Updating flow entries.

584. Riddle teaches that each of these state operations depends on the content of the packet. For searching state operation (1), Riddle's various searches for patterns/referencing strings will depend on the contents of the packet.⁷⁴⁰ For creating new flow-entry (2), Riddle teaches creating a new flow-entry only for new flows.⁷⁴¹ For service aggregates, Riddle describes creating a subset of flows only that meet the requirements of a service aggregate.⁷⁴² For determining metrics (4), Riddle calculates metrics (e.g., byte count, most hits, most recently seen, most data transferred, moving average, bytes per second, most recently used, most hits, and number of bytes received) depending on the contents of the packet.⁷⁴³ And or updating entries (5), Riddle will only updates flow entries for previously encountered flows.⁷⁴⁴

585. Whether these state operations are performed depends on the content of the packet. For example, (1) Riddle's invention will search for patterns/referencing strings for certain traffic classes, e.g., web traffic⁷⁴⁵, (2) a new flow-entry is only

⁷⁴⁰ Riddle, 8:67-9:11, 9:24-26, 11:48-49.

⁷⁴¹ Riddle, 12:48-57, 13:36-62, Figs. 4A-4B.

⁷⁴² Riddle, 11:10-36.

⁷⁴³ Riddle, 12:53-13:8, 13:36-14:5, Fig. 4A-4B.

⁷⁴⁴ Riddle, 12:53-59, Fig. 4A.

⁷⁴⁵ Riddle, 8:64-9:5, 9:24-26, 11:48-49.

created for new flows⁷⁴⁶, (3) state operations related to “service aggregates” are performed for appropriate traffic, e.g., FTP traffic⁷⁴⁷, (4) the metrics that are determined (e.g., byte count, most hits, time most recently seen, most data transferred, moving average, bytes per second, most recently used, most hits, and number of bytes received) will depend on the packet⁷⁴⁸, and (5) updating flow entries depends on the packet’s characteristics such as time encountered and byte count.⁷⁴⁹

4. Dependent ’725 Claim 13

586. Riddle in view of Baker renders obvious this claim. Claim 13 depends from independent claim 10 and recites:

A method according to claim 10, wherein the protocol descriptions are provided in a protocol description language.

587. The ’725 Patent discloses an embodiment in which “a protocol description language,” as recited in claim 10, is a PDL file:

A set of PDL files is used to describe what information is relevant to packets [¶] There is one file for each packet type and each protocol. Thus, there is a PDL file for Ethernet packets and there is a PDL file for frame relay packets.⁷⁵⁰

⁷⁴⁶ Riddle, 12:48-57, Fig. 4A.

⁷⁴⁷ Riddle, 11:10-36 ;13:52-59.

⁷⁴⁸ Riddle, 12:48-13:8, 14:1-5, Fig. 4A

⁷⁴⁹ Riddle, 12:48-59, Fig. 4A.

⁷⁵⁰ ’725 Patent, 41:30-40.

588. Similarly, Baker teaches that “protocol description files (PDF)” contain protocol descriptions: “In the presently preferred embodiment, each of these protocol description records with its associated field, statistics, lookup, and filter record information is also written to a protocol specific protocol description file (PDF).”⁷⁵¹ Baker also discloses that its PDFs include a protocol control record and a plurality of field data records.⁷⁵² And as I describe with respect to ’725 claim elements 10.2-10.6 and 17.2-17.6, Baker teaches an exemplary PDF for Ethernet packets and another for Generic Protocols.⁷⁵³ Accordingly, a POSITA would have understood that a PDF, such as those taught by Baker, provide a “protocol description language.”

589. As I discuss above in Section VII.B.2 regarding ’099 claim 4, the ’725 patent describes a “protocol description language” as follows:

Input to the compiler includes a set of files that describe each of the protocols that can occur. These files are in a convenient protocol description language (PDL) which is a high level language. PDL is used for specifying new protocols and new levels, including new applications. The PDL is independent of the different types of packets and protocols that may be used in the computer network. A set of PDL files is used to describe what information is relevant to packets and packets

⁷⁵¹ Baker, 19:6-10.

⁷⁵² Baker, 12:25-28.

⁷⁵³ Baker, Tables 12-13.

that need to be decoded. The PDL is further used to specify state analysis operations. Thus, the parser subsystem and the analyzer subsystems can adapt and be adapted to a variety of different kinds of headers, layers, and components and need to be extracted or evaluated, for example, in order to build up a unique signature.⁷⁵⁴

The protocol description language (PDL) files 336 describes both patterns and states of all protocols that an occur at any layer, including how to interpret header information, how to determine from the packet header information the protocols at the next layer, and what information to extract for the purpose of identifying a flow, and ultimately, applications and services. ... This information is input into compiler and optimizer 310.⁷⁵⁵

590. The '725 Patent discloses examples of PDL files that include commands for a particular protocol.⁷⁵⁶ Similarly, Baker discloses that each PDF file describes commands for a particular protocol.⁷⁵⁷ As the following examples illustrate, Baker's disclosures regarding its PDF files teach the '725 claim 13's protocol descriptions provided in a protocol description language.

591. First, according to the '725 Patent, an exemplary command in a high-level

⁷⁵⁴ '725 Patent, 41:24-37.

⁷⁵⁵ '725 Patent, 9:29-40; Fig. 4.

⁷⁵⁶ '725 Patent, 45:1-94:67 (including PDL files for several protocols).

⁷⁵⁷ Baker, 11:22-25 (“[E]ach of these protocol description records with its associated field, statistics, lookup, and filter record information is also written to a protocol specific protocol description file (PDF).”).

protocol description language is a “HEADER” attribute that “describe[s] the length of the protocol header.”⁷⁵⁸ Baker similarly discloses a “numBits” attribute that describes “the total bit length of the protocol header.”⁷⁵⁹

592. Second, according to the '725 Patent, another exemplary command in a high-level protocol description language is a “PROTOCOL” definition used to “define the order of the FIELDS and GROUPs within the protocol header.”⁷⁶⁰ Similarly, Baker discloses a “fields” attribute that references the associated “field records that describe the protocol header” where each field record includes, for example, a “fblen” attribute describing “the length of the field in bits” and a “fdwoff” attribute describing “the byte offset from the start of protocol header,” among other attributes.⁷⁶¹

593. Third, according to the '725 Patent, a further exemplary command in a high-level protocol description language is a “CHILDREN” attribute “used to describe how children protocols are determined.”⁷⁶² Similarly, Baker discloses a “ptr2np”

⁷⁵⁸ '725 Patent, 48:41-50, col. 73 (“HEADER { LENGTH=14 }”).

⁷⁵⁹ Baker, Table 1 (“numBits” attribute), 11:32, 57:1 (fread(&num_bits, sizeof(num_bits), 1, fp); // Read fixed header length in bits).

⁷⁶⁰ '725 Patent, 47:34-48:18, col. 79 (PROTOCOL section provides “Detailed packet layout for the IP datagram. This includes all fields and format. All offsets are relative to the beginning of the header.”).

⁷⁶¹ Baker, Table 1 (“fields” attribute), Table 2 (“fblen” and “fdwoff” attributes), 11:36-40, 11:49-50, 147:35-36.

⁷⁶² '725 Patent, 49:45-55, col. 79 (“CHILDREN { DESTINATION=Protocol }”).

attribute of each field record that includes a “pointer to lookup structure/class . . . next protocol definition to use (0 = none)” and the “next protocol lookup records” are described with reference to Table 4 as including a “Protocol” attribute describing the “pointer to protocol description record,” among other attributes.⁷⁶³

594. As such, a POSITA would have understood that Baker’s PDFs are protocol descriptions provided in a protocol description language. This is because, upon initialization, Baker teaches that the system is able to “extract[] the protocol and associated control record information” from the file, construct a ProtocolList, and read the PDF file into memory in the sequence described at col. 11:26-12:6. Accordingly, Baker’s PDFs include commands in a high-level language that describe protocols that may be encountered by the monitor and, for example, how to interpret header information and how to determine from the packet header information the protocol at the next layer. Baker thus teaches protocol description files that are provided in a protocol description language, and are written in a protocol description language as described in by the ’725 Patent at col. 41:24-37.

595. As discussed previously, it would have been obvious to a POSITA to modify Riddle’s protocol descriptions in view of Baker, by providing those protocol

⁷⁶³ Baker, Table 2 (“ptr2np” attribute), Table 4 (“Protocol” attribute), 8:13-15, 148:10-12, 165:37-166:5, 177:4-26.

description in a protocol description language. A POSITA would have been motivated to provide Riddle's protocol descriptions in a protocol description language, as taught by Baker, because this would allow Riddle's classification system to easily update protocol descriptions via PDFs, as discussed above with respect to '725 claim elements 10.2, 10.6, 17.2, and 17.6.

5. Dependent '725 Claim 16

596. Riddle discloses all the limitations of this claim. Claim 16 depends from independent claim 10 and recites:

A method according to claim 10, wherein the protocol specific operations further include one or more state processing operations that are a function of the state of the flow of the packet.

597. The discussion above for '725 claim element 17.7 shows this element has been met. Specifically, that discussion shows that Riddle discloses performing state processing operations that are a function of the state of the flow of the packet such as whether saved traffic belongs to a service aggregate and determining metrics.⁷⁶⁴

598. For all the above reasons, it is my opinion that Riddle in view of Baker renders obvious claims 10, 12, 13, 16, and 17 of the '725 Patent.

⁷⁶⁴ Riddle, 12:53-13:8, 13:36-14:5, Figs. 4A-4B.

B. For the '725 Patent, Riddle in View of Baker and Further in View of Yu Renders Obvious Claims 10, 12, 13, 16, and 17.

599. It is my opinion that a POSITA would have recognized that each and every limitation of the '725 claims 10, 12, 13, 16, and 17 is disclosed or rendered obvious in light of Riddle in view of Baker and further in view of Yu. Specifically, my opinions regarding Riddle in view of Baker are exactly the same as those above in Section VIII.A, but further include the teachings of Yu. Thus, as if fully set forth here, I incorporate the discussion from Section VIII.A regarding the obviousness of '725 claims 10, 12, 13, 16, and 17 over Riddle in view of Baker.

600. As discussed above, all of the Challenged Claims require “conversational flows” or a “conversational-flow sequence.” For example, '725 claim element 10.7 recites “form[ing] a function of the selected portions for identifying the packet as belonging to a conversational flow.” While Riddle itself teaches identifying conversational flows, Yu further demonstrates identifying conversational flows through its “flow classification.”⁷⁶⁵ Further, Yu teaches flow classification logic that “keeps track of the flow’s state until matching criteria is met” when identifying whether a packet belongs to a conversational flow.⁷⁶⁶

⁷⁶⁵ Yu, 1:56-60, 3:32-36, 3:47-49; 4:1-8.

⁷⁶⁶ Yu, 4:62-64.

601. As discussed with respect the obviousness of '099 claims 1 and 2 over Riddle in view of Ferdinand and further in view of Yu in Section VII.C, Yu teaches identifying the claimed “conversational flow” by Yu’s flow classifier linking multiple “streams” into a “flow” based on application or application data.⁷⁶⁷ And as discussed in Section VII.C, Yu teaches state tracking that binds policy decisions to each stream of a flow so that actions are taken on future packets without intervention from the “host” application.⁷⁶⁸ Moreover, as discussed in Section VII.C, Yu specifies using hash values to find corresponding policies to reduce further complicated pattern-matching.⁷⁶⁹ I incorporate by reference that discussion as if fully set forth herein.

602. As discussed with respect the obviousness of '099 claims 1 and 2 over Riddle in view of Ferdinand and further in view of Yu in Section VII.C, I explain how a POSITA would have been motivated to combine Yu’s teachings into Riddle’s monitor. I incorporate by reference that discussion as if fully set forth herein.

603. For the same reasons, it is my opinion that combining the teachings of Riddle in view of Baker and further in view of Yu renders obvious all the claim elements relating to “conversational flows,” as well as carrying out state processing operations that are a function of the packet’s conversational flow state.

⁷⁶⁷ Yu, 1:56-60, 3:32-49; 4:1-8.

⁷⁶⁸ Yu, 4:57-5:13.

⁷⁶⁹ Yu, 4:23-29.

604. As set forth in my analysis of the '725 Patent in Sections VIII.A.2 through VIII.A.5 above, Riddle and Baker disclose or render obvious all the remaining elements of '725 claims 10, 12, 13, 16, and 17. Thus, it is my opinion that Riddle in view of Baker and further in view of Yu renders obvious '725 claims 10, 12, 13, 16, and 17.

C. For the '725 Patent, Riddle in View of Baker and Further in View of RFC1945 Renders Obvious Claims 10, 12, 13, 16, and 17.

605. It is my opinion that a POSITA would have recognized that each and every limitation of the '725 Patent claims 10, 12, 13, 16, and 17 is disclosed or rendered obvious in light of Riddle in view of Baker and further in view of RFC1945. Specifically, my opinions regarding Riddle in view of Baker are exactly the same as those above in Section VIII.A, but further include the teachings of RFC1945. Thus, as if fully set forth here, I incorporate the discussion from Section VIII.A regarding the obviousness of '725 Patent claims 10, 12, 13, 16, and 17 over Riddle in view of Baker.

606. As discussed above, all of the Challenged Claims require “conversational flows.” For example, '725 claim element 10.7 reciting “form[ing] a function of the selected portions for identifying the packet as belonging to a conversational flow.” While Riddle discloses identifying packets as belonging to a conversational flow, RFC1945 further demonstrates identifying packets as belonging to a conversational flow through the additional example of the use of HTTP header fields.

607. As discussed with respect to the obviousness of '099 claims 1 and 2 over Riddle in view of Ferdinand and further in view of RFC1945 in Section VII.E, I understand that Patentee and its technical expert have taken the position that the HTTP Referrer field can be used to create a conversational flow. I incorporate by reference that discussion as if fully set forth herein.

608. As discussed with respect the obviousness of '099 claims 1 and 2 over Riddle in view of Ferdinand and further in view of RFC1945 in Section VII.E, I explain how a POSITA would have been motivated to combine RFC1945's teachings into Riddle's monitor. I incorporate by reference that discussion as if fully set forth herein.

609. For the same reasons, it is my opinion that combining the teachings of Riddle, Baker, and RFC1945 renders obvious all the claim elements relating to "conversational flows," at least under Patentee's interpretation of that term.

610. As set forth in my analysis of the '725 Patent in Sections VIII.A.2 through VIII.A.5 above, Riddle and Baker disclose or render obvious all the remaining elements of '725 claims 10, 12, 13, 16, and 17. Thus, it is my opinion that Riddle in view of Baker and further in view of RFC1945 renders obvious '725 claims 10, 12, 13, 16, and 17 at least under Patentee's interpretation of "conversational flow."

IX. THE CLAIMS OF THE '646 PATENT ARE UNPATENTABLE

611. For the '646 Patent, the challenged claims include independent claims 1, 7,

and 16 as well as dependent claims 2, 3, and 18. As I detail below, it is my opinion that Riddle in view of Ferdinand and Wakeman renders obvious '646 claims 1-3, 7, 16, and 18. It is also my opinion that Riddle in view of Ferdinand and Wakeman and further in view of Yu renders obvious '646 claims 1-3, 7, 16, and 18. Moreover, it is my opinion that Riddle in view of Ferdinand and further in view of RFC1945 renders obvious '646 claims 1-3, 7, 16, and 18.

A. For the '646 Patent, Riddle in View of Ferdinand and Wakeman Renders Obvious Claims 1-3, 7, 16, and 18.

612. It is my opinion that a POSITA would have recognized that each and every limitation of the '646 claims 1-3, 7, 16, and 18 is disclosed or rendered obvious by the prior art. Specifically, it is my opinion that '646 claims 1-3, 7, 16, and 18 are obvious in light of Riddle in view of Ferdinand and Wakeman.

1. Reasons to Modify Riddle in View of Ferdinand and Wakeman

613. As described above with respect to the '099 Patent in Section VII.A.1, a POSITA would have been motivated and found it obvious to combine the teachings of Riddle and Ferdinand.

614. Riddle, Ferdinand, and Wakeman are in the same field of endeavor and contain overlapping disclosures with similar purposes. For example, as described above in Sections IV.A and VII.A.1, Riddle discloses a packet monitor that parses packets, stores flow-entries lists of packet identifying information, looks up flow-

entries, and carries out state operations to identify a previously-encounter conversational flow or to store a new conversational flow.⁷⁷⁰

615. And as described above in Section IV.F, Wakeman discloses a network switch that includes a content addressable memory (CAM) cache coupled to a switch engine and forwarding database.⁷⁷¹ Wakeman teaches using the switch for source and destination of Ethernet packets.⁷⁷²

616. As I describe below regarding each relevant claim element, a POSITA would have found it obvious to modify Riddle's lookup engine with a cache subsystem as taught by Wakeman. In particular, a POSITA would have been motivated to employ a CAM-cache with Riddle's lookup engine because caches were well known to reduce look-up times to improve system responsiveness.⁷⁷³ And Wakeman details that its CAM-cache could be used for source and destination of Ethernet packets.⁷⁷⁴ Based on Wakeman's teachings, a POSITA would have appreciated the benefits of using a CAM-cache to manage commonly encountered entries of Riddle's flow-entry lists as those lists also include destination and source addresses for Ethernet packets to improve system performance.

⁷⁷⁰ E.g., Riddle, 8:47-9:27, 12:26-53, claim 8, Figs. 4A-4B.

⁷⁷¹ E.g., Wakeman, 3:26-28, 3:54-4:20, Figs. 2-3.

⁷⁷² Wakeman, 1:15-18.

⁷⁷³ '646 Patent, 2:36-51; '646 Prosecution History, 197-198 (09/10/2003 Office Action, p.7).

⁷⁷⁴ Wakeman, 1:15-18.

2. Independent '646 Claim 1

617. It is my opinion that independent claim 1 of the '646 Patent is obvious in light of Riddle in view of Ferdinand and in light of Riddle in view of Ferdinand and Wakeman.

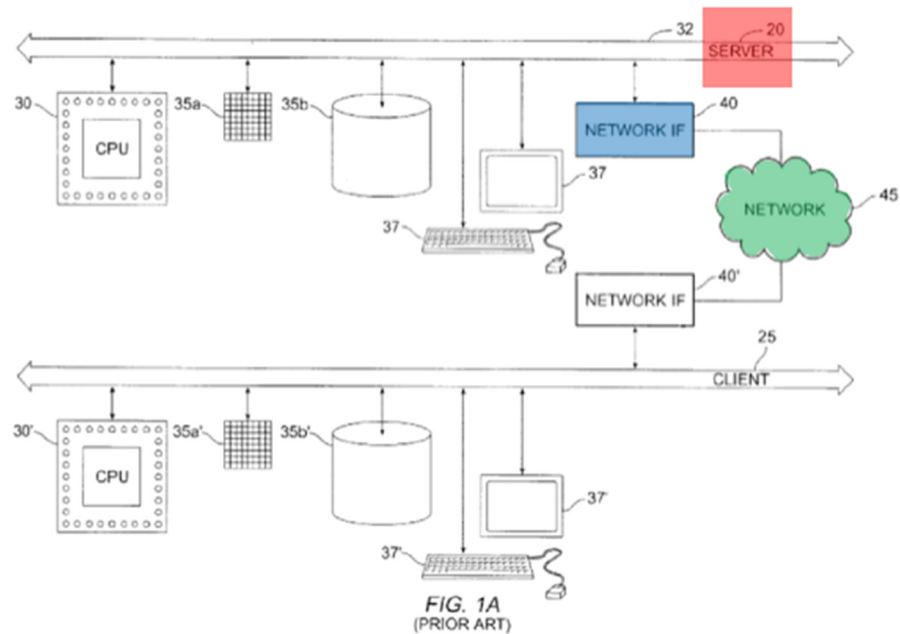
- a. *'646 Claim 1's Preamble: "A packet monitor for examining packet[s]⁷⁷⁵ passing through a connection point on a computer network, each packet[] conforming to one or more protocols, the monitor comprising"*

618. Riddle discloses all elements of this preamble. As discussed with respect to '099 claim 1's preamble in Section VII.A.2.a, Riddle teaches a packet monitor for examining and classifying packets passing through a network connection point using traffic classifier 304.⁷⁷⁶ I incorporate by reference that discussion as if fully set forth herein. For example, Riddle's classifier 304 parses and examines multiple packets of traffic flow, such as traffic flows a, b, and c shown in Figure 3. As shown below in Figure 1A, Riddle details that its traffic classification system may be implemented in server 20 (shown below in red), which acts as a packet monitor.⁷⁷⁷

⁷⁷⁵ It is my understanding that corrections to claim language are based on '646 Patent's Certificates of Correction.

⁷⁷⁶ Riddle, 4:6-17, 7:21-24, 12:27-41, 14:22-40, Fig. 3.

⁷⁷⁷ Riddle, 5:53-67, 6:9-15, 7:21-24, Figs. 1A-1C.



619. As I detail above regarding '099 claim 1's preamble, Riddle describes that its packet monitor examines packets as the packets pass through connection points of network connection 45 (shown above in green).⁷⁷⁸ And for networks connecting multiple clients and servers of Figure 1A, Riddle teaches examining packets via a network routing means, routers (e.g., router 75) or in another traffic classifier 304.⁷⁷⁹ Riddle details that router 75 acts as a "system gateway ... which may also be a gateway having a firewall or a network bridge."⁷⁸⁰ Based on these disclosures, a POSITA would have understood that Riddle's system gateway is a connection point on the computer network. And as the '646 Patent teaches, a connection point

⁷⁷⁸ Riddle, 5:53-67, 6:9-15, Figs. 1A-1B.

⁷⁷⁹ Riddle, 7:10-34, claim 8, Figs. 1C, 3.

⁷⁸⁰ Riddle, 7:21-24.

is simply where the packet monitor is connected to the network.⁷⁸¹

620. Further, Riddle describes “each packet conforming to one or more protocols,” as recited in ’646 claims 1’s preamble. Riddle, for example, states that its system “relates to digital packet telecommunications, and particularly to management of network bandwidth based on information ascertainable from multiple layers of OSI network model.”⁷⁸² Riddle specifies that its system address the lack of “teaching in the prior art of methods for automatically classifying packet traffic based upon information gathered from a [sic] multiple layers in a multi-layer protocol network.”⁷⁸³

621. As shown in Riddle’s Figure 1D provided below, it was well known in the art before the priority date of the Challenged Patents that the OSI network model diagrams the relationship between the layers of the TCP/IP protocol suite. These layers include the application layer 88, the transport layer 86, the network layer 84, the data link layer 82, and the physical layer 80.⁷⁸⁴

⁷⁸¹ ’646 Patent, 4:54-5:8, Fig. 1 showing connection points 121, 123, 125.

⁷⁸² Riddle, 1:54-57.

⁷⁸³ Riddle, 3:36-39.

⁷⁸⁴ Riddle, 7:35-8:46, Fig. 1D.

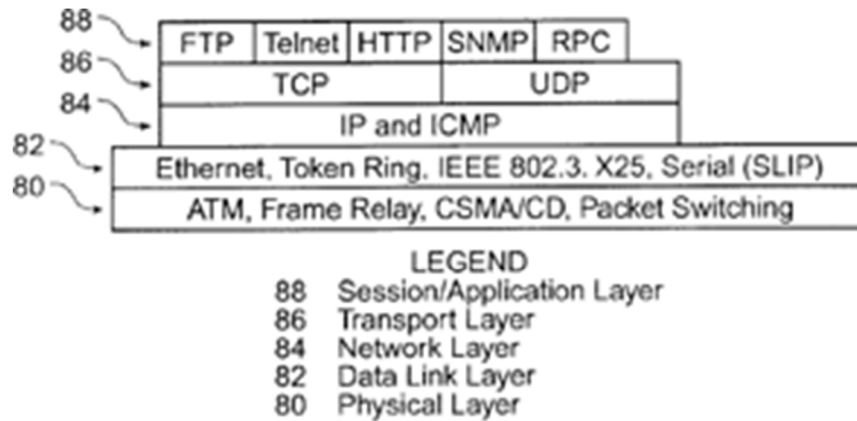


FIG. 1D
(PRIOR ART)

622. Riddle describes these prior art layers in detail:

FIG. 1D is illustrative of the constituents of the Transmission Control Protocol/Internet Protocol (TCP/IP) protocol suite. *The base layer of the TCP/IP protocol suite is the physical layer 80*

Overlying the physical layer is the data link layer 82. The data link layer provides the function and protocols to transfer data between network resources and to detect errors that may occur at the physical layer. ...

Network layer protocols 84 overlay the datalink layer and provide the means for establishing connections between networks. ...

The transport layer protocols 86 provide end-to-end transport services across multiple heterogenous networks. The User Datagram Protocol (UDP) provides a connectionless, datagram oriented service which provides a non-reliable delivery mechanism for streams of information. The Transmission Control Protocol (TCP) provides a reliable session-based service for delivery of sequenced packets of information across the Internet. TCP provides a connection oriented reliable mechanism for information delivery.

The session, or application layer 88 provides a list of network applications and utilities, a few of which are illustrated here. For example, File Transfer Protocol (FTP) is a standard TCP/IP protocol for transferring files from one machine to another. FTP clients establish sessions through TCP connections with FTP servers in order to obtain files. Telnet is a standard TCP/IP protocol for remote terminal connection. A Telnet client acts as a terminal emulator and establishes a connection using TCP as the transport mechanism with a Telnet server. The Simple Network Management Protocol (SNMP) is a standard for managing TCP/IP networks. ... The Hypertext Transfer Protocol (HTTP) facilitates the transfer of data objects across networks via a system of uniform resource indicators (URI).

The Hypertext Transfer Protocol is a simple protocol built on top of Transmission Control Protocol (TCP). It is the mechanism which underlies the function of the World Wide Web. The HTTP provides a method for users to obtain data objects from various hosts acting as servers on the Internet.⁷⁸⁵

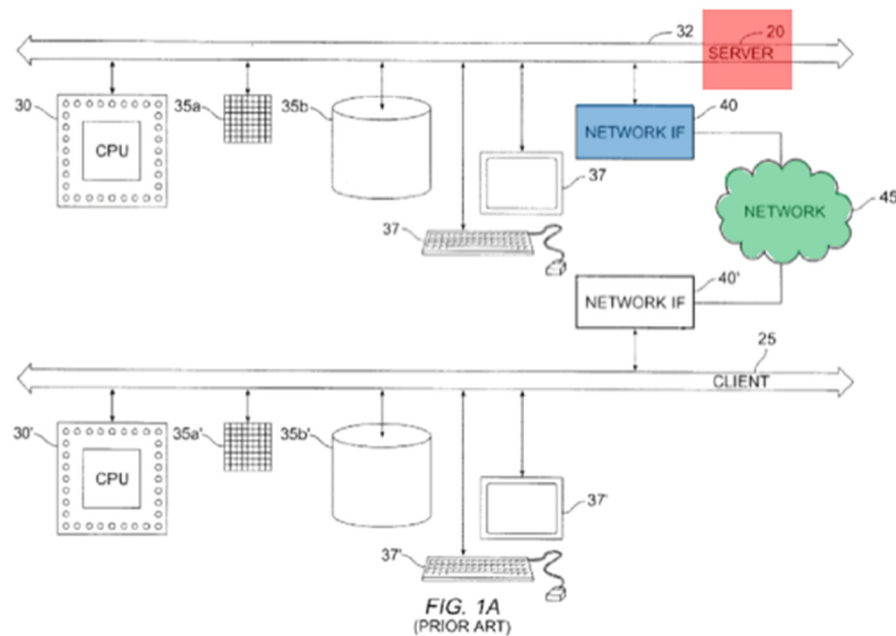
623. Based upon these disclosures, a POSITA would have understood that Riddle discloses receiving each packet conforming to one or more protocols, such as the protocols disclosed in the OSI model.

b. '646 Claim Element 1.1: “(a) a packet acquisition device coupled to the connection point and configured to receive packets passing through the connection point”

624. Riddle discloses this claim element. As discussed with respect to '099 Claim

⁷⁸⁵ Riddle, 7:35-8:46.

1's preamble in Section VII.A.2.a, Riddle teaches a packet acquisition device (e.g., network interface 40) connected to the connection point.⁷⁸⁶ I incorporate by reference that discussion as if fully set forth herein. As shown below in annotated Figures 1A and 1B, Riddle illustrates an exemplary traffic classification system with packet monitor (server 20) coupled to a packet acquisition device (network interface 40) receiving packets over network connection 45. And Riddle teaches that its memory subsystem 35a holds data, e.g., packets, in preparation for parsing and examination.⁷⁸⁷



⁷⁸⁶ Riddle, 6:5-15, Figs. 1A-1B.

⁷⁸⁷ Riddle, 6:1-23, claim 8, Figs. 1A-1B; Ex. 1027 (Packer Application, Appendices, incorporated-by-reference into Riddle), 71-72.

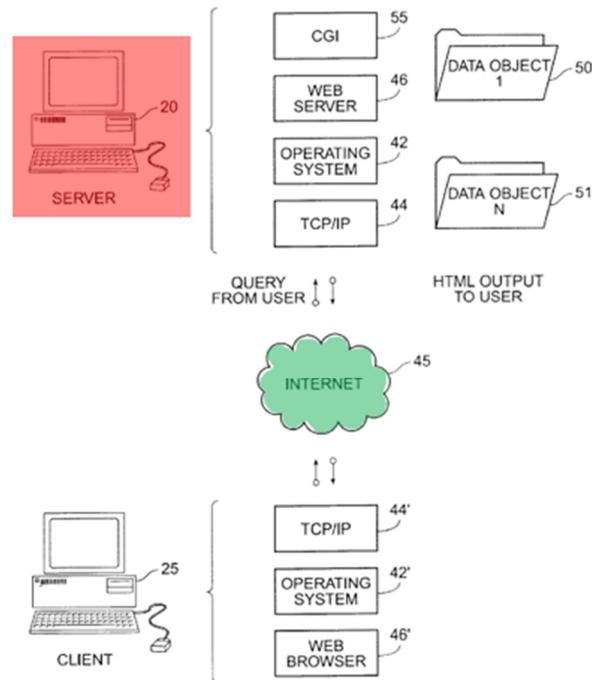


FIG. 1B
(PRIOR ART)

625. For networks connecting multiple clients and servers of Figure 1A, Riddle describes a network routing means, which may include router 75, depicted in Figure 1C that a POSITA would have understood as having a packet acquisition device coupled to one or more connection points.⁷⁸⁸ And as I explain above regarding the '646 claim 1's preamble, Riddle details that router 75 acts as a “system gateway,”⁷⁸⁹ which a POSITA would have understood is a connection point.

626. With respect to receiving packets, Riddle describes methods to automatically classifying packet flows to help allocate bandwidth resources.⁷⁹⁰ Riddle teaches

⁷⁸⁸ Riddle, 7:21-34, 16:54-60.

⁷⁸⁹ Riddle, 7:21-24.

⁷⁹⁰ Riddle, Abstract, 4:7-10.

“applying individual instances of traffic classification paradigms to packet network flows based on selectable information obtained from a plurality of layers of a multi-layered communication protocol in order to define a characteristic class, then mapping the flow to the defined traffic class.”⁷⁹¹ Riddle specifies that its system’s “automatic classification is sufficiently robust to classify a complete enumeration of the possible traffic.”⁷⁹² Further, Riddle’s methods provide “techniques to automatically classify a plurality of heterogeneous packets in a packet telecommunications system for management of network bandwidth in systems such as a private area network, a wide area network or an internetwork.”⁷⁹³ As such, in order to classify packet flows, a POSITA would have understood that Riddle’s acquisition device is configured to receive packets.

- c. *'646 Claim Element 1.2: “(b) a memory for storing a database comprising flow-entries for previously encountered conversational flows to which a received packet may belong, a conversational flow being an exchange of one or more packets in any direction as a result of an activity corresponding to the flow”*

627. Riddle renders obvious this claim element alone or in view of Ferdinand.

- (1) Riddle teaches the claimed “memory for storing a database comprising flow-entries”

628. As discussed with respect to '099 claim element 1.5 in Section VII.A.2.f,

⁷⁹¹ Riddle, Abstract, 4:10-15.

⁷⁹² Riddle, Abstract, 4:15-17.

⁷⁹³ Riddle, 4:55-60.

Riddle teaches its monitor includes storage subsystem 35 and stores flow-entry lists of previously-encountered flows.⁷⁹⁴ I incorporate by reference that discussion as if fully set forth herein. For example, Riddle's Figure 4A shows the process of parsing a flow specification from a packet and then storing the flow specifications in the saved lists 308.⁷⁹⁵ And Riddle teaches incorporating saved lists 308 into the classification tree (for example, traffic tree 302), where each node of the tree represents a traffic class.⁷⁹⁶ In Figure 4B, Riddle discloses accessing previously-encountered flow-entries stored in the lists 308.⁷⁹⁷ Riddle's Figures 4A and 4B are provided below with step 420 (in green) showing retrieving previously stored flow-entry data and steps 426 and 428 (in red) showing testing and creating a traffic class that "will match all components of the service aggregate."⁷⁹⁸

⁷⁹⁴ Riddle, 6:1-23, 6:43-50, 12:37-59, Figs. 1A-1B, 3.

⁷⁹⁵ Riddle, 12:42-59, Fig. 4A.

⁷⁹⁶ Riddle, 9:28-33, 8:47-50.

⁷⁹⁷ Riddle, 13:35-62, Fig. 4B.

⁷⁹⁸ Riddle, 13:52-59; '864 Provisional, 69.

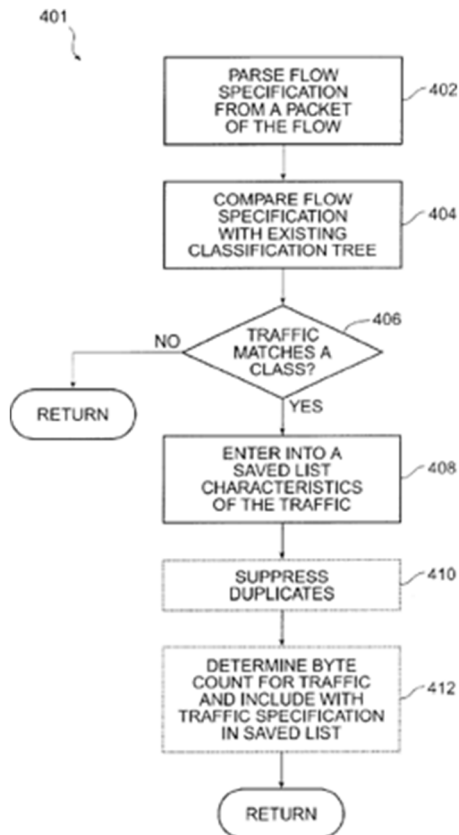


FIG. 4A

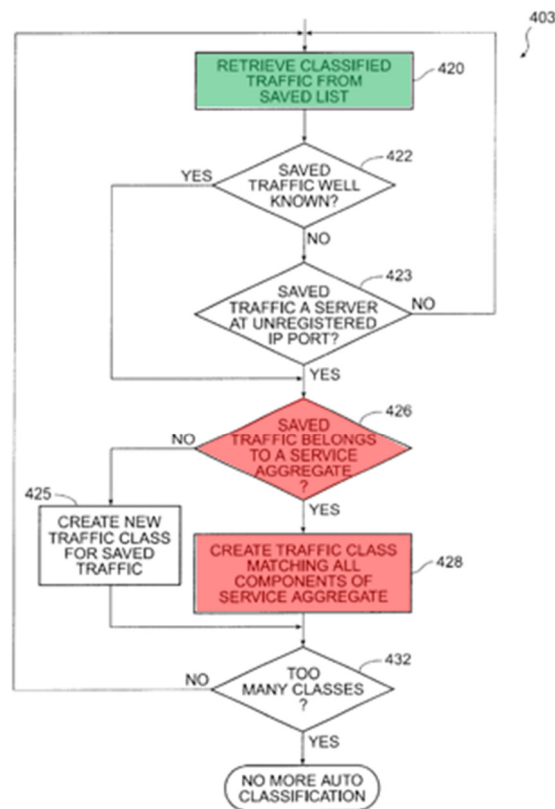


FIG. 4B

629. And as I discuss above regarding '099 claim element 1.5, Riddle alone renders obvious storing Riddle's flow-entry lists in a database. I incorporate by reference that discussion as if fully set forth herein. For example, as I discuss above regarding '099 claim element 1.2, Riddle teaches that relational database 306 stores the heuristics for determining traffic classes.⁷⁹⁹ And it was well known to a POSITA to store data relating to the network traffic in databases allows for faster lookup.⁸⁰⁰ Thus, storing Riddle's flow-entries in a database as a set of tables

⁷⁹⁹ Riddle, 12:32-35.

⁸⁰⁰ Riddle, 6:1-15, 15:1-15; Ferdinand, 23:19-23, 28:16-24; '099 Prosecution History, 213-214 (06/25/2003 Office Action, p.3).

amounts to nothing more than a simple implementation leading to predictable results.

630. As I discuss above regarding '099 claim element 1.5, Riddle in view of Ferdinand renders obvious storing Riddle's flow-entry lists in a database. I incorporate by reference that discussion as if fully set forth herein. For example, Ferdinand describes organizing and storing classification information in STATS database 36.⁸⁰¹ Riddle and Ferdinand both describe classifying the same types of traffic, such as FTP and other protocol types like TCP and UDP.⁸⁰² And like Riddle, Ferdinand discloses displaying the results of its analysis to a user.⁸⁰³

631. A POSITA would have been motivated to implement Riddle's lists in a flow-entry database, based on Ferdinand's teachings to allow multiple network operators to access simultaneously Riddle's classification information and to increase functionality of storing data in a database including searching, analyzing, and modifying the flow-entries.⁸⁰⁴ Modifying Riddle's monitor would have led to predictable results given that Riddle's saved lists include similar information as that saved in Ferdinand's database. As such, storing Riddle's classification information in a

⁸⁰¹ Ferdinand, 23:3-22, 28:14-17, Figs. 7A-7C.

⁸⁰² Riddle, 10:1-18 (Table 2); Ferdinand, 29:4-30:10, 39:23-40:16.

⁸⁰³ Riddle, 12:64-13:9, 14:1-5; Ferdinand, 60:10-15, Fig. 22.

⁸⁰⁴ Riddle, 12:37-38, 12:61-63.

flow-entry database as a set of tables is nothing more than an obvious implementation based on Ferdinand's teachings.

- (2) Riddle's service aggregates teach the claimed "previously encountered conversational flows to which a received packet may belong, a conversational flow being an exchange of one or more packets in any direction as a result of an activity corresponding to the flow"

632. As I discuss above with respect to '099 claim element 1.4 in Section VII.A.2.e, Riddle teaches storing "conversational flows" by classifying separate packet flows by a common "service aggregates" traffic class for applications using multiple flows between a client and a server.⁸⁰⁵ I incorporate by reference that discussion as if fully set forth herein. Riddle, for example, discloses classifying based on a service aggregate flow that links separate application-specific conversations:

A service aggregate is provided for certain applications that use more than one connection in a particular conversation between a client and a server. For example, an FTP client in *conversation* with an FTP server employs a command channel and a transfer channel, which are distinct TCP sessions on two different ports. In cases where two or three TCP or UDP sessions exist *for each conversation* between one client and one server, it is useful to provide a common traffic class i.e., the service aggregate, containing *the separate conversations*. In practice, these types of *conversations* are between the same two hosts, but use

⁸⁰⁵ Riddle, 11:10-36, 12:64-13:62, claim 2, Figs. 2A-2B, 4B; RFC765, 6-7; Ex. 1023 (German Court Translation), 35-36.

different ports. According to the invention, a class is created with a plurality of traffic specifications, each matching various component *conversations*.⁸⁰⁶

633. As further discussed above in '099 claim element 1.4, Riddle specifies that its parser subsystem checks whether the parser packet portion is part of a service aggregate (i.e., a conversational flow) shown in Figure 4B.⁸⁰⁷ As shown below, Riddle illustrates this check for “service aggregate” conversational flows in Figure 4B’s flowchart. And when describing command options for controller 304, Riddle details in Table 3 specifies the controller may “detect services in both directions.”⁸⁰⁸

634. As I detail above in Section IV.A.6, Riddle describes creating traffic classes based on data relating to RTP and RTSP.⁸⁰⁹ A POSITA would have understood RTP and RTSP are analogous to FTP, and that those protocols use a separate control flow with one or more linked dataflows.⁸¹⁰ And, as detailed above, a POSITA would have appreciated this is another example of a common “service aggregate” traffic class for applications using multiple flows between a client and a server.

⁸⁰⁶ Riddle, 11:10-23.

⁸⁰⁷ Riddle, 13:36-62, Fig. 4B.

⁸⁰⁸ Riddle, 14:28-40.

⁸⁰⁹ Riddle, 12:1-12.

⁸¹⁰ Ex. 1045, 4-5 (RFC1889 illustrating well-known RTP information); Ex. 1046, 9-10 (RFC2326 illustrating well-known RTSP information).

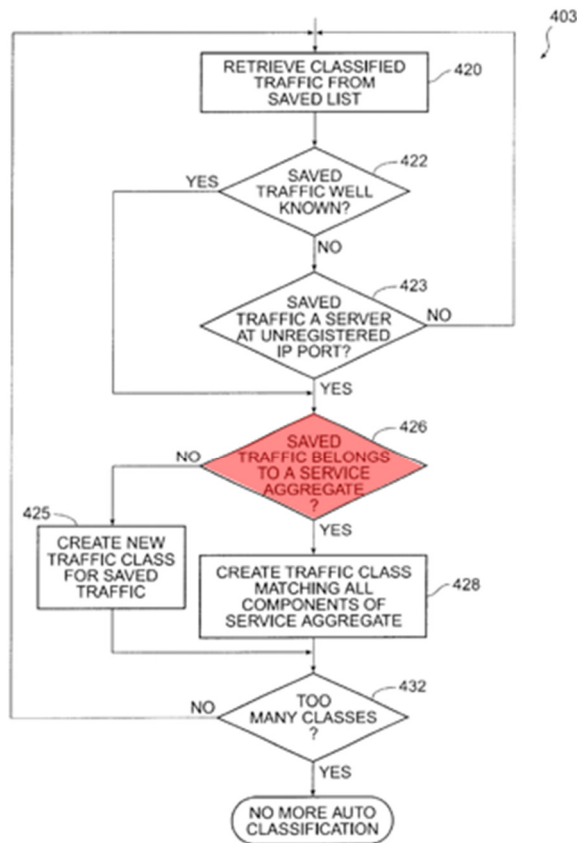


FIG. 4B

As such, Riddle teaches the claimed storing of “previously encountered conversational flows to which a received packet may belong, a conversational flow being an exchange of one or more packets in any direction as a result of an activity corresponding to the flow.”

635. Moreover, as discussed above in Section III.J, I agree with the German court’s finding that an FTP communication, which comprises two TCP connections, taught a “conversational flow.” Riddle similarly teaches identifying multiple flows as part of an FTP communication. As the German court recognized, the identification of flows as part of an FTP communication (as Riddle discusses) teaches

identifying packets as being part of a conversational flow-sequence.

- (3) Riddle’s PointCast flows teach the claimed “previously encountered conversational flows to which a received packet may belong, a conversational flow being an exchange of one or more packets in any direction as a result of an activity corresponding to the flow”

636. As discussed with respect to ’099 claim element 1.4 in Section VII.A.2.e, Riddle teaches storing “conversational flows” by classifying separate packet flows as being PointCast traffic.⁸¹¹ I incorporate by reference that discussion as if fully set forth herein. Riddle, for example, discloses examining packet portions for URLs that begin with /FIDO-1/ to classify flows as corresponding to PointCast Network applications.⁸¹² And as discussed, Patentee’s ’903 Provisional acknowledges it was known that separate PointCast flows include a signature specific to PointCast traffic and those flows are associated with the PointCast Network application.⁸¹³

- d. ’646 Claim Element 1.3: “(c) a cache subsystem coupled to the flow-entry database memory providing for fast access of flow-entries from the flow-entry database”*

637. Riddle in view of Ferdinand renders obvious this claim element. Similarly,

⁸¹¹ Riddle, 11:47-67, 15:28-31; Ex. 1016 (’903 Provisional), 7:16-25; Ex. 1032 (Wall Street Journal PointCast article), 1; Ex. 1033 (Computer World PointCast article); Ex. 1034 (Christian Science Monitor PointCast article); Ex. 1036 (’558 Patent), 33:28-44.

⁸¹² Riddle, 11:47-67.

⁸¹³ Ex. 1016 (’903 Provisional), 7:16-25, 28:22-24.

Riddle in view of Wakeman renders obvious this claim element. As discussed above regarding '646 Claim Element 1.2, Riddle discloses a memory for storing a flow-entry database or renders it obvious in view of Ferdinand. I incorporate by reference that discussion as if fully set forth herein.

638. Riddle describes examining flow-entries from stored flow-entry lists that would have rendered obvious utilizing a cache with the flow-entry database memory:

In [Figure 4B's] step 420, an instance of saved traffic is retrieved from the saved traffic list 308. Next in a decisional step 422, the instance of saved traffic is examined to determine whether it is well known (e.g. registered SAP, protocol type, assigned port number) and a name representing its type exists. If this is so then processing continues with a test of whether the saved traffic belongs to a service aggregate in step 426.⁸¹⁴

639. Further, using a cache subsystem to speed up processing of flows was well known in the art. For example, the '646 Patent states that it was well-known at the time of the priority date to use caches to provide fast access to a subset of the contents that are likely to be accessed in larger, slower memory:

Because of the high speed that packets may be entering the system, it is desirable to maximize the hit rate in a cache system. *Typical prior-art*

⁸¹⁴ Riddle, 13:40-47.

cache systems are used to expediting memory accesses to and from microprocessor systems. Various mechanisms are available in such prior art systems to predict the lookup such that the hit rate can be maximized. Prior art caches, for example, can use a lookahead mechanism to predict both instruction cache lookups and data cache lookups. Such lookahead mechanisms are not available for a cache subsystem for the packet monitoring application. When a new packet enters the monitor, the next cache access, for example from the lookup engine, may be for a totally different conversational flow than the last cache lookup, and there is no way ahead of time of knowing what flow the next packet will belong to.⁸¹⁵

640. Ferdinand discloses several examples of well-known caches used in network monitors. For example, Ferdinand teaches that its monitor includes two sets of 64KB cache: “Monitor 10 also includes a 64Kbyte instruction cache and a 64Kbyte data cache, implemented by SRAM.”⁸¹⁶

641. With respect to its flow-entry database (STATS 36), Ferdinand details that the database is coupled to a cache:

STATS defines the database and it contains subroutines for updating the statistics which it keeps. STATS ... provides an initialization routine whose major function is to allocate statistics records at startup from cache-able memory. It provides lookup routines in order to get at the statistics.⁸¹⁷

⁸¹⁵ ’646 Patent, 2:36-51.

⁸¹⁶ Ferdinand, 18:27-29.

⁸¹⁷ Ferdinand, 28:14-21.

And Ferdinand teaches storing SunNet Manager (“SNM”) files in a cache coupled to a workstation database.⁸¹⁸

642. Based on a POSITA’s own knowledge and Ferdinand’s teachings, a POSITA would have been motivated and found it obvious to modify the flow-entry database memory with a common routing component link a cache subsystem. Before the priority date of the Challenged Patents, a POSITA would have known that caches were commonly-used components in routing devices.⁸¹⁹ Based on Ferdinand’s teachings, a POSITA would have been motivated to modify Riddle’s database memory with a cache because caches were well known to reduce look-up times.⁸²⁰ A POSITA would have appreciated that a cache, such as Ferdinand’s, was easily modified to store Riddle’s flow-entry lists, because an innate function of a cache is to store information, such as Riddle’s flow-entry list. In view of Ferdinand, using a cache for storing Riddle’s flow-entry data amounts to nothing more than an obvious implementation of known prior art technologies used in the ordinary and predictable manner to provide a cache for frequently accessed flow entries in Riddle’s monitor.

⁸¹⁸ Ferdinand, 54:18-22.

⁸¹⁹ ’646 Patent, 2:36-51; ’646 Prosecution History, 193-200 (09/10/2003 Office Action, p.3).

⁸²⁰ ’646 Patent, 2:36-51; ’646 Prosecution History, 197-198 (09/10/2003 Office Action, p.7).

643. As I detail below with respect to '646 claim 3, Wakeman teaches another example of a well-known cache subsystem. Wakeman discusses prior-art network switch 10 as having forwarding database (FSB) 12, which is shown below in Figure 1. Wakeman explains that databases, such as FSB 12, are “typically implemented either as a hardware content addressable memory (CAM) or as RAM.”⁸²¹ And Wakeman states that a hardware CAM “is very fast and can typically retrieve mappings in less than 100 ns.”⁸²²

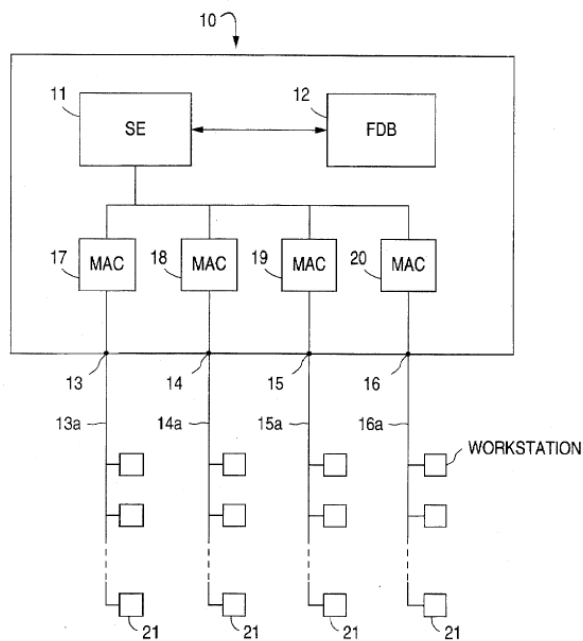


FIG. 1
(PRIOR ART)

644. Wakeman discloses a CAM cache:

⁸²¹ Wakeman, 1:55-56.

⁸²² Wakeman, 1:56-58.

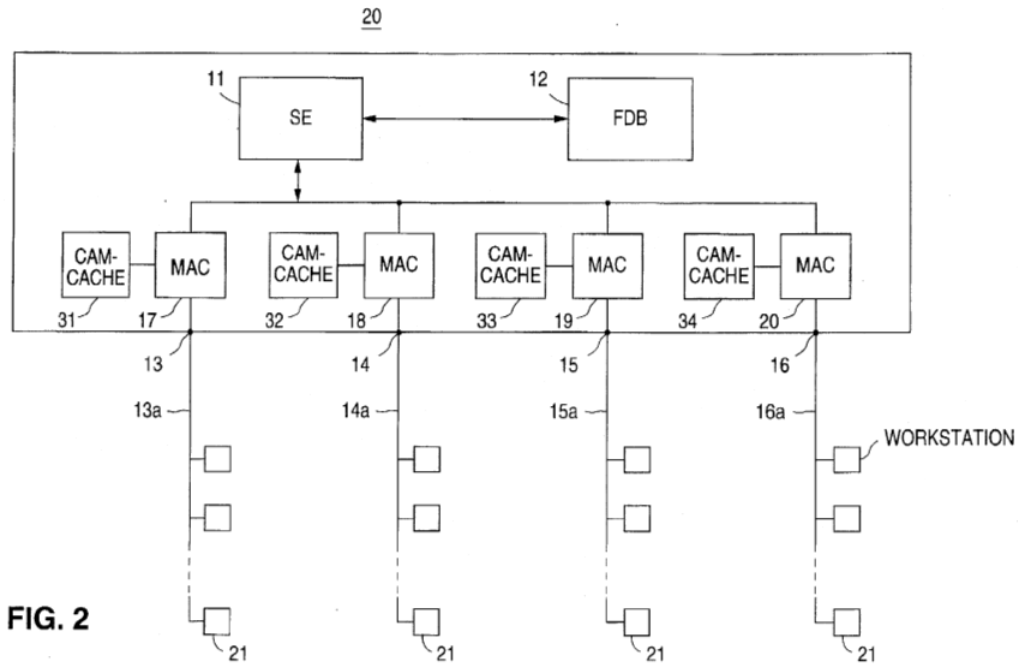
Upon receiving an incoming data packet, the MAC associated with the source port will, after extracting the destination address from the packet, access its associated CAM-cache to find the correct address-to-port mapping. If the correct mapping is contained in the CAM-cache, the packet may be immediately forwarded to the destination port without having to access the much larger and slower forwarding database.

Where the CAM-cache does not contain the correct mapping, the MAC then causes the correct mapping to be retrieved from the forwarding database. The packet may then be forwarded to the correct destination port. The CAM-cache is then updated with this mapping so that succeeding packets having the same destination address-to-port mapping may be quickly forwarded to the destination port by accessing only the fast CAM-cache, thereby eliminating the need to access the much slower forwarding database.⁸²³

645. As shown below in Figure 2, Wakeman teaches network switch 20 having switch engine 11, forwarding database 12, media access controllers 17-20, and CAM-Caches 31-34.⁸²⁴

⁸²³ Wakeman, 2:31-49.

⁸²⁴ Wakeman, 1:20-28, 3:36-45, Fig. 2.



646. Wakeman describes that “CAM caches 31-34 may be distributed across ports 13-16 of switch 20, where one of the CAM caches described above may service more than one port.”⁸²⁵ And Wakeman describes the CAM cache’s internal architecture and operation as follows:

CAM cache 31 which, in accordance with the preferred embodiment, includes a FIFO 35, a memory 36, a learning and aging block 38, and logic 37. These elements are well understood in the art and thus will not be discussed below. The extracted source and destination addresses of the first packet are queued in FIFO 35 which, in turn, provides the destination address to memory 36. If the correct destination mapping is contained in memory 36, there is thus a match and memory 36 provides the correct destination port to logic 37 which, in turn, forwards the port location and a

⁸²⁵ Wakeman, 5:28-31.

“hit” signal to MAC 17. MAC 17 then alerts SE 11 of the correct destination port. SE 11 informs MAC 18 that a packet is “addressed” thereto and directs the first packet to MAC 18 which, in turn, forwards the packet to segment 14a where it will be accepted by the workstation having the correct destination address. Thus, where the correct destination mapping is contained in CAM cache 31, accessing and searching FDB 12 is wholly unnecessary. Since the accessing speed of CAM cache is much faster than that of FDB 12, the inclusion of CAM caches 31-34 in a network switch as described above results in an increase in forwarding speed. Note that although the FDB 12 in switch 20 is preferably a RAM, CAM caches 31-34 will decrease the access time and thus increase forwarding speeds irrespective of the particular construction of FDB 12 (e.g., where FDB 12 is a hardware CAM as opposed to RAM).

If the correct destination mapping is not contained in memory 36, logic 37 sends a “miss” signal to MAC 17 which then alerts SE 11 of the destination address extracted from the packet. SE 11 then searches FDB 12 to locate the correct destination mapping and, having retrieved the correct destination mapping, forwards the packet as described earlier with reference to prior art switch 10.⁸²⁶

647. As provided below, Wakeman’s Figure 3 illustrates the internal architecture of the CAM-Cache.

⁸²⁶ Wakeman, 3:54-4:20.

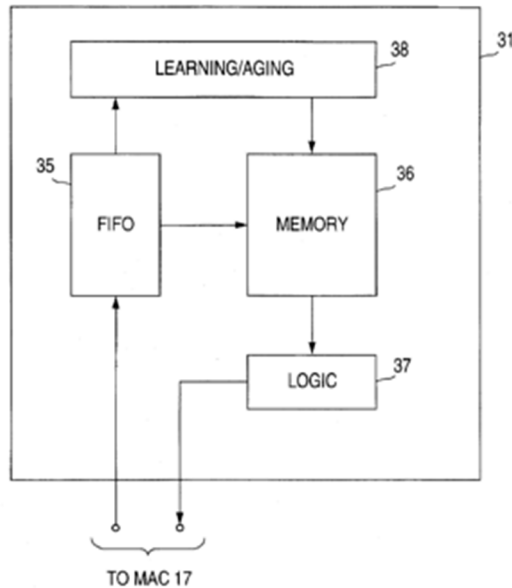


FIG. 3

648. For the additional reasons set forth in Wakeman’s teachings, it would have been obvious to a POSITA to modify Riddle’s memory storing the flow-entry database with a cache subsystem. Based on Wakeman’s teachings, a POSITA would have been motivated to use a CAM-cache because caches were well known to reduce look-up times.⁸²⁷ Wakeman envisions that its CAM-cache would be used for source and destination of Ethernet packets.⁸²⁸ And Riddle’s monitor may be used in an Ethernet network connecting a host (e.g., SPARC workstation 71) and another host (e.g., VAX6000 computer 72), and can classify traffic classes defined by

⁸²⁷ ’646 Patent, 2:36-51; ’646 Prosecution History, 197-198 (09/10/2003 Office Action, p.7).

⁸²⁸ Wakeman, 1:5-18.

inside and outside IP addresses.⁸²⁹ In view of Wakeman, it would have been obvious to a POSITA to modify Riddle's memory storing the flow-entry database to improve the database by providing faster access and retrieval of likely-to-be-accessed flow-entries, such as for Ethernet packet source and destination address.⁸³⁰ In making such a modification, a POSITA would have further modified Riddle's memory, which would contain the flow-entry database and is accessible by criteria such as "most 'hits' during a recent interval" or "most recently-seen (most recent time first),"⁸³¹ with the CAM-cache. The flow-entry database of the Riddle-Ferdinand combination with the teachings of Wakeman's CAM-cache would have provided for fast access of a set of likely-to-be-accessed flow-entries.⁸³²

- e. *'646 Claim Element 1.4: "(d) a lookup engine coupled to the packet acquisition device and to the cache subsystem and configured to lookup whether a received packet belongs to a flow-entry in the flow-entry database, the looking up being [via] the cache subsystem; and"*

649. Riddle in view of Ferdinand or Wakeman renders obvious this claim element. As discussed above regarding '646 claim element 1.2, Riddle alone or in view of Ferdinand renders obvious a memory for storing a flow-entry database. I incorporate by reference that discussion as if fully set forth herein.

⁸²⁹ Riddle, 7:24-28, 8:58-62, Fig. 1C.

⁸³⁰ Wakeman, 1:15-18.

⁸³¹ Riddle, 12:65-13:10.

⁸³² Wakeman, 4:31-40, 5:22-27, Fig. 3 (logic 38 maintains likely-to-be-accessed flow-entries).

650. As discussed with respect to '099 Claim Element 1.6 in Section VII.A.2.g, Riddle discloses a lookup engine configured to determine whether a received packet belongs to a flow-entry in the flow-entry database.⁸³³ I incorporate by reference that discussion as if fully set forth herein. For example, and as shown in Figure 4B, Riddle teaches looking up whether a flow matches a traffic class in relation to classifying a service aggregate based on a plurality of indicators.⁸³⁴ From the pertinent steps in flowcharts of Figures 4A-4B, a POSITA would have understood that Riddle discloses looking up whether a received packets belongs to a flow-entry (e.g., class A) in traffic tree 302 corresponding to a conversational flow.

651. Thus, Riddle discloses an entity that examines flow entries to determine if a received packet belongs to a stored flow-entry. A POSITA would understand that the lookup engine in Riddle is a processor and that corresponding code performs the functions discussed above as Riddle discloses that its monitor is “for automatically classifying heterogeneous packets in a packet telecommunications environment of the present invention is implemented in the C programming language and is operational on a computer.”⁸³⁵ And as discussed above regarding '646 claim element 1.2, Riddle alone or in view of Ferdinand renders obvious a flow-entry database.

⁸³³ Riddle, 12:37-49, claim 8, Figs. 3, 4A-4B.

⁸³⁴ Riddle, 13:42-47, claim 5, Fig. 4B.

⁸³⁵ Riddle, 5:53-57.

652. Further, Riddle teaches that its lookup engine is coupled to the packet acquisition device. For example, as shown in Figure 1A, Riddle teaches that it was known in the art for the engine's processor (e.g., CPU 30) to be couple to the packet acquisition device (e.g., network interface 40).⁸³⁶ Similarly, Riddle's claim 8 recites a system having a packet acquisition device (e.g., network routing means) coupled to the lookup engine (e.g., a processor means).

653. Further, a POSITA would have understood that Riddle's lookup engine and packet acquisition device work together to achieve the goal of classifying traffic.⁸³⁷ And in view of Ferdinand's and Wakeman's teachings regarding caches, a POSITA would have been motivated to modify Riddle's lookup engine with a cache subsystem and the packet acquisition device. As discussed in regarding '646 claim element 1.3, it would have been obvious to POSITA to modify Riddle with a cache subsystem to improve Riddle's performance based upon the admitted prior art, Ferdinand's and Wakeman's teachings, or the knowledge of a POSITA. When Riddle's monitor is modified with a cache subsystem as discussed regarding '646 claim element 1.3, then a POSITA would have understood the looking up would be done via the cache subsystem.

⁸³⁶ Riddle, 6:5-15, Fig. 1A.

⁸³⁷ Riddle, 4:15-17.

f. '646 Claim Element 1.5: “(e) a state processor coupled to the lookup engine and to the flow-entry-database memory, the state processor being to perform any state operations specified for the state of the flow starting from the last encountered state of the flow in the case that the packet is from an existing flow, and to perform any state operations required for the initial state of the new flow in the case that the packet is [not] from an existing flow”

654. Riddle discloses this claim element. As discussed above regarding '646 claim element 1.2, Riddle alone or in view of Ferdinand renders obvious a memory for storing a flow-entry database. I incorporate by reference that discussion as if fully set forth herein. As I detail below, Riddle discloses traversing steps of a traffic classification tree, which teaches the claimed “state processor” limitations.

655. As discussed with respect to '099 claim elements 1.9 and 1.10 in Sections VII.A.2.j and VII.A.2.k, Riddle discloses a state processor coupled to the flow-entry database memory and being able to perform state operations for existing and new flows.⁸³⁸ And as discussed above regarding '099 claim element 1.7 in Section VII.A.2.h, Riddle details traversing steps of a traffic classification tree.⁸³⁹ Riddle's classification is a common data structure in which each of the tree's leaf nodes corresponds to last encountered flow states.⁸⁴⁰ I incorporate by reference those discussions as if fully set forth herein.

⁸³⁸ Riddle, 5:53-57, 12:42-13:23, 14:1-5, claim 8, Fig. 4A.

⁸³⁹ Riddle, 10:19-56, Figs. 2A-2B, 3.

⁸⁴⁰ Riddle, 9:28-41.

656. Further, Riddle discloses that its processor performs state operations:

A system for automatically classifying traffic in a packet telecommunications network, said network having any number of flows, including zero, comprising:

a plurality of network links upon which said traffic is carried; a network routing means; and,

a processor means operative to:

parse a packet into a first flow specification... thereupon,

match the first flow specification of the parsing step to a plurality of classes represented by a plurality of said classification tree type nodes, each said classification tree type node having a traffic specification and a mask, according to the mask; thereupon,

*if a matching classification tree type node was not found in the matching step, associating said first flow specification with one or more newly-created classification tree type nodes; thereupon, incorporating said newly-created classification tree type nodes into said plurality of said classification tree type nodes.*⁸⁴¹

657. A POSITA would have understood that Riddle's state processor, lookup engine, and flow-entry database memory work together to achieve the goal of classifying traffic.⁸⁴² Thus, a POSITA would have understood that the state processor is coupled to the lookup engine and to the flow-entry-database memory.

658. Riddle describes storing previously-encountered flows:

⁸⁴¹ Riddle, claim 8, 5:53-57.

⁸⁴² Riddle, 4:15-17.

A traffic classifier 304 detects services for incoming traffic. ... A plurality of saved lists 308 stores classified traffic pending incorporation into traffic tree 302. *In select embodiments, entries for each instance of traffic may be kept. In alternate embodiments, a copy of an entry and a count of duplicate copies for the entry is maintained.*⁸⁴³

659. As discussed above regarding '099 claim element 1.7, Riddle teaches performing state operations of on existing flows. For example, Riddle's Figure 2A illustrates FTP operations for traversing a particular transition pattern that includes (1) comparing the packets' source (client) IP to the range of IP addresses defined for subnet B and, if so, (2) determining if the sequence of packets involves the FTP protocol.⁸⁴⁴ In doing so, Riddle teaches testing for subclassifications when encountering packet information relating to FTP applications such as Figure 2A's classes 206, 210.⁸⁴⁵ When testing for such subclassifications, Riddle's classifier performs an operation to determine if the flow is an FTP command flow or an FTP data flow.⁸⁴⁶

660. Moreover, as discussed above regarding '099 claim element 1.7, Riddle teaches performing state operations for flows involving FTP applications to deter-

⁸⁴³ Riddle, 12:30-41.

⁸⁴⁴ Riddle, 10:19-39.

⁸⁴⁵ Riddle, 11:24-36.

⁸⁴⁶ Riddle, 11:12-15.

mine if the flow belongs to a service aggregate (i.e., a conversational flow-sequence).⁸⁴⁷ As shown in Riddle's Figure 2A, a POSITA would have understood that traversing this example transition pattern (Client IP/Subnet B/FTP/FTP-cmd/FTP service aggregate) as a result of a particular conversational flow-sequence of packets, Riddle teaches an association with the FTP-application program initiated on the user's (client's) computer.

661. For example, Riddle discloses applying policies associated with "leaf" nodes of the classification tree.⁸⁴⁸ Based on Riddle's teachings, a POSITA would have understood that each traffic class, such as FTP-server or World-Wide-Web traffic classes, can have its own respective policies.⁸⁴⁹ As a result, Riddle's processor ultimately applies a policy based on the state of the flow. Packer, which Riddle incorporates-by-reference, illustrates this application in Figure 5F provided below.⁸⁵⁰

⁸⁴⁷ Riddle, 11:10-23; 13:52-57; Fig. 4B.

⁸⁴⁸ Riddle, 9:29-42, Packer, 18:3-5 ("The processing steps of flowchart 511 determine a class and a policy for a flow[.]", Fig. 5F.

⁸⁴⁹ Riddle, 10:36-39.

⁸⁵⁰ Riddle, 9:29-42, Claim 3; Packer, Fig. 5F.

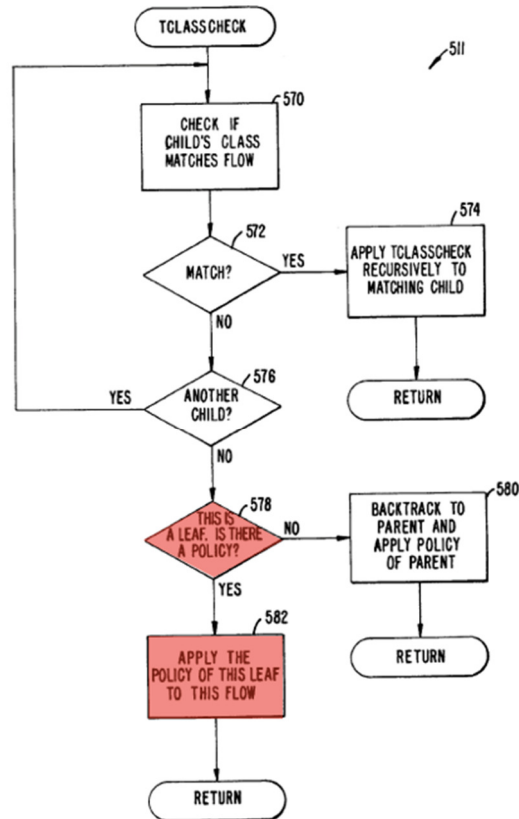


FIG. 5F.

662. As I discuss with respect to '646 claim element 1.2, Riddle teaches storing previously-encountered conversational flows.⁸⁵¹ Riddle discloses performing state operations on existing conversational flows, such as updating flow-entries when traffic matches a class and suppressing duplicates.⁸⁵² When suppressing duplicates for existing flows, Riddle teaches performing the state operation of updating flow-entries.⁸⁵³

⁸⁵¹ Riddle, 12:30-41.

⁸⁵² Riddle, 12:42-59.

⁸⁵³ Riddle, 12:42-59, claim 15, Fig. 4A.

663. For new conversational flows, Riddle discloses the state operation of creating a new flow-entry.⁸⁵⁴ The Challenged Patents describe creating a new flow-entry as an example of a state operation.⁸⁵⁵

664. For both existing and new conversational flows, Riddle describes performing the state operation of determining statistical metrics.⁸⁵⁶ The Challenged Patents describe determining statistical metrics as an example of a state operation.⁸⁵⁷ As such, a POSITA would have understood that Riddle determines the packet's state of conversational flow, at least by determining metrics like the count of duplicates, most recent time the monitor encountered a flow with the same identifying characteristics, and byte count of the detected flow.⁸⁵⁸

665. In addition, as discussed above regarding '099 claim element 1.7, the Challenged Patents explain that state operations may include (a) searching for one or more patterns in the parsed packet information, (b) creating a new flow-entry, (c)

⁸⁵⁴ Riddle Figure 4A's step 408.

⁸⁵⁵ E.g., '789 Patent, claim 47.

⁸⁵⁶ Riddle, 12:42-13:8, 14:1-5, Fig. 4A.

⁸⁵⁷ E.g., '646 Patent, 10:38-46, 16:25-43; '751 Patent, claim 16.

⁸⁵⁸ Riddle, 12:53-13:8, 14:1-5, Fig. 4A (steps 410, 412).

determining metrics that relate to the examined flow, and (d) updating a flow-entry.⁸⁵⁹ As detailed above in '099 claim element 1.7, Riddle teaches each of these types of state operations.⁸⁶⁰

666. Riddle further describes displaying a hierarchical classification tree that shows state operations related to FTP. For example, Riddle presents classes “to host 1,” “tcp,” and “FTP.”⁸⁶¹ As such, a POSITA would have understood that transitioning from one class to the next in this hierarchy involves operations.

667. Regarding Figure 4A, Riddle also describes analyzing information identifying the characteristics of the traffic as the classifier parses packets of a flow and matches the parsed packets to a class.⁸⁶² Riddle describes that, after an initial classification, “sub-classification” proceeds in a sequential manner by performing finer grade matchings as characteristics such as the hosts and services are identified, leading to matching of a flow with operation of a particular application program.⁸⁶³ In accordance with the classification tree, Riddle describes advancing through the sequence of packets in a particular traffic flow to parse and classifying the packets.

⁸⁵⁹ '646 Patent, claim 13; '789 Patent, claims 17, 27, 46 (searching for patterns); '789 Patent, claim 47 (creating new flow-entry); '751 Patent, claims 11-13, 16 (determining metrics); '646 Patent, claim 15; '751 Patent, claim 13; '789 Patent, claims 15, 30, 45.

⁸⁶⁰ E.g., Riddle, 11:48-63, 12:48-13:8, 13:59-14:5, Figs. 4A-4B.

⁸⁶¹ Riddle, 13:11-22.

⁸⁶² Riddle, 12:42-48.

⁸⁶³ Riddle, 11:25-31, 13:11-22.

This results in performing a corresponding state operation at each node to update the identifying characteristics of the flow.

668. To the extent Patentee asserts that the “lookup engine,” “state processor,” and “parser subsystem” of ’646 claim elements 1.4-1.5 and 2.1 require separate pieces of hardware, it would have been obvious to a POSITA to modify Riddle’s processor and programming code to be separate hardware components. This is because using dedicated hardware for various functions, especially functions as common as parsing, data lookup, and state processing, would have been well understood by a POSITA. As acknowledged by the Challenged Patents, such a modification would have been obvious to a POSITA:

Each of the individual hardware elements through which the data flows in the system are now described with reference to FIGS. 10 and 11. Note that while we are describing a particular hardware implementation of the invention embodiment of FIG. 3, *it would be clear to one skilled in the art that the flow of FIG. 3 may alternatively be implemented in software running on one or more general-purpose processors, or only partly implemented in hardware.* An implementation of the invention that can operate in software is shown in FIG. 14. The hardware embodiment (FIGS. 10 and 11) can operate at over a million packets per second, while the software system of FIG. 14 may be suitable for slower networks. *To one skilled in the art it would be clear that more and more of the system may*

*be implemented in software as processors become faster.*⁸⁶⁴

669. Further, Ferdinand discloses its monitor can include separate hardware components for performing various functions, such real time parser (RTP) 32, database 36, boot/load 22, and memory transport module 34, event manager 38, and control module 42.⁸⁶⁵ As provided below, Ferdinand's Figure 5 illustrates its monitor having separate hardware components.

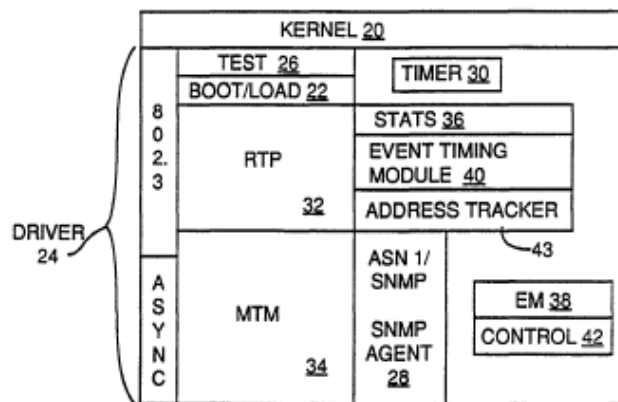


FIG 5

670. Desiring increased performance, a POSITA would have been motivated to utilize dedicated hardware components for parsing, lookups, and state processing. On the other hand, a POSITA would have understood that Riddle's use of a processor for these functions is less expensive and a more extensible solution than using dedicated hardware components.

⁸⁶⁴ '646 Patent, 17:7-21.

⁸⁶⁵ Ferdinand, 19:5-13.

671. For the above reasons, it is my opinion that Riddle in view of Ferdinand renders obvious claim 1 of the '646 Patent. And it is my opinion that Riddle in view of Ferdinand and Wakeman renders obvious claim 1 of the '646 Patent.

3. Dependent '646 Claim 2

672. Riddle discloses all the limitations of this claim. Claim 2 depends from independent claim 1.

- a. *'646 Claim Element 2.1: "a parser subsystem coupled to the packet acquisition device and to the lookup engine such that the acquisition device is coupled to the lookup engine via the parser subsystem, the parser subsystem configured to extract identifying information from a received packet"*

673. Riddle discloses this claim element. As discussed with respect to '099 claim elements 1.3 and 1.4 in Sections VII.A.2.d and VII.A.2.e, Riddle teaches a parser subsystem configured to extract selected portions of the accepted packet.⁸⁶⁶ I incorporate by reference those discussions as if fully set forth herein. For example, Riddle discloses a processor programmed to perform extraction operations:

[A] processor means operative to: parse a packet into a first flow specification, wherein said first flow specification contains at least one instance of any one of the following: a protocol family designation, a direction of packet flow designation, a protocol type designation, a pair of hosts, a pair of ports, in HTTP protocol packets, a pointer to a MIME type.⁸⁶⁷

⁸⁶⁶ Riddle, 6:1-15, 12:26-53, claim 8, Figs. 1A, 4A.

⁸⁶⁷ Riddle, claim 8.

674. Further, Riddle explains that its parser subsystem operates to apply “individual instances of traffic classification paradigms to packet network flows based on selectable information obtained from a plurality of layers of a multi-layered communication protocol in order to define a characteristic class, then mapping the flow to the defined traffic class.”⁸⁶⁸ As part of this examination and extraction process, Riddle’s system employs a relational database (knowledge base 306) to store heuristics (i.e., operations) for determining traffic classes based on parsing/extracting portions of a packet and matching those portions’ attributes to a traffic class.⁸⁶⁹

675. As shown in Figure 4A’s flowchart, Riddle’s traffic classification monitor includes parsing and extracting packet portions: “parse flow specification from a packet of the flow” (step 402), “traffic matches a class?” (step 406), and “enter into saved list characteristics of the traffic” (step 408).⁸⁷⁰ Following the parsing, rules are checked and if the flow matches a traffic class an entry is made into a saved list with the extracted identifying information such as “protocol type (SAP), IP protocol number, server port, traffic type if known, MIME type, a time of occurrence of the traffic.”⁸⁷¹

676. A POSITA would have understood that Riddle’s parser subsystem, packet

⁸⁶⁸ Riddle, 4:10-15.

⁸⁶⁹ Riddle, 12:26-41, 9:28-42, 9:48-49.

⁸⁷⁰ Riddle, 12:42-53, Fig. 4A.

⁸⁷¹ Riddle, 12:50-53.

acquisition device, and lookup engine work together to achieve the goal of classifying traffic.⁸⁷² Thus, a POSITA would have understood that Riddle's parser subsystem is coupled to the packet acquisition device and to the lookup engine such that the acquisition device is coupled to the lookup engine via the parser subsystem.

- b. '646 Claim Element 2.2: *“wherein each flow-entry is identified by identifying information stored in the flow-entry, and wherein the cache lookup uses a function of the extracted identifying information”*

677. Riddle in view of Ferdinand renders obvious this claim element. This claim element is similar to '646 claim elements 1.2 and 7.4, and is disclosed or rendered obvious by the prior art for the same reasons. I incorporate by reference my discussion regarding '646 claim elements 1.2 and 7.4 as if fully set forth herein.

678. As shown in Figure 4A's flowchart, Riddle teaches parsing packet and extracting a flow specification used to classify the packet flow:

FIG. 4A depicts a flowchart 401 of processing steps for automatically classifying traffic. In a step 402, a flow specification is parsed from the flow being classified. Then in a step 404, the flow specification parsed from the flow in step 402 is compared with the traffic specifications in each node of the classification tree. Rules are checked starting from most specific to least specific. In a decisional step 406, a determination is made if traffic matches one of the classes being classified. If this is so, then in a step 408,

⁸⁷² Riddle, 4:15-17.

an entry is made in a list of identifying characteristics, such as protocol type (SAP), IP protocol number, server port, traffic type if known, MIME type, a time of occurrence of the traffic.⁸⁷³

And as shown in Figure 4A's step 408, Riddle discloses entering characteristics of the traffic into the saved list.⁸⁷⁴

679. As discussed regarding '646 claim elements 1.3 and 1.4, Riddle in view of Ferdinand, Wakeman, or by common knowledge of POSITA would have rendered obvious the modification of Riddle's lookup engine with a cache subsystem such that the looking up is via the cache subsystem. Based on the teachings of Riddle in view of Ferdinand and/or Wakeman, a POSITA would have understood that Riddle's lookup engine modified with the cache subsystem results in the cache lookup using a function of the extracted identifying information.

680. As such, it is my opinion that Riddle in view of Ferdinand and Wakeman renders obvious dependent claim 2 of the '646 Patent.

4. Dependent '646 Claim 3

681. Riddle in view of Ferdinand and Wakeman renders obvious this claim.

Claim 3 depends from dependent claim 2 and recites: "A packet monitor according to claim 2, wherein the cache subsystem is an associative cache subsystem including one or more content addressable memory cells (CAMs)."

⁸⁷³ Riddle, 12:42-53.

⁸⁷⁴ Riddle, 12:42-59, Fig. 4A.

682. The '646 Patent describes the well-known cache design paradigm that includes content addressable memory cells:

The cache subsystem 1115 is an associative cache that includes a set of content addressable memory cells (CAMs) each including an address portion and a pointer portion pointing to the cache memory (e.g., RAM) containing the cached flow-entries. The CAMs are arranged as a stack ordered from a top CAM to a bottom CAM. The bottom CAM's pointer points to the least recently used (LRU) cache memory entry. Whenever there is a cache miss, the contents of cache memory pointed to by the bottom CAM are replaced by the flow-entry from the flow-entry database 324. This now becomes the most recently used entry, so the contents of the bottom CAM are moved to the top CAM and all CAM contents are shifted down. Thus, the cache is an associative cache with a true LRU replacement policy.⁸⁷⁵

Consistent with the '646 Patent, a POSITA would have understood that an associative cache was a well-known cache design paradigm in which a memory block can appear in any cache line and that the cache line is a form of a content addressable memory.

683. Riddle in view of Wakeman renders obvious claim 3. In Section IV.F, I detailed the features of Wakeman, which is incorporated by reference as if fully set forth herein. As detailed above, a POSITA would have understood that Wakeman's

⁸⁷⁵ '646 Patent, 19:56-20:2.

CAM cache can be a fully associative cache. This is due to Wakeman not disclosing a cache placement policy, which indicates there being no constraints on placement within Wakeman's cache. And at the time of Challenged Patent's priority date (as well as today), an associative cache was a common type of cache readily known to a POSITA. Using an associative cache, or any other type of well-known cache, was readily understood by a POSITA for expediting memory accesses to and from processors.⁸⁷⁶

684. As set forth regarding '646 claim element 1.3, it would have been obvious to a POSITA to couple Riddle's flow-entry to a cache subsystem. A POSITA would have been motivated to do so to improve performance as taught by the admitted prior art, Wakeman, Ferdinand, or the knowledge of a POSITA. For example, in view of Wakeman, a POSITA would have been motivated to use a CAM-cache because such caches were well-known to reduce look-up times. Wakeman envisions that its CAM-cache would be used for source and destination addresses of Ethernet packets. A POSITA would have appreciated the benefits to using a CAM-cache to store Riddle's flow-entry lists as those lists also include destination and source addresses for Ethernet packets based on Wakeman's teachings.

685. As such, it is my opinion that Riddle in view of Ferdinand and Wakeman

⁸⁷⁶ '646 Patent, 2:36-51; '646 Prosecution History, 193-200 (09/10/2003 Office Action, p.3).

renders obvious dependent claim 3 of the '646 Patent.

5. Independent '646 Claim 7

686. It is my opinion that independent claim 7 of the '646 Patent is obvious in light of Riddle in view of Ferdinand or in light of Riddle in view of Ferdinand and Wakeman.

- a. *'646 Claim 7's Preamble: "A packet monitor for examining packets passing through a connection point on a computer network, each packet conforming to one or more protocols, the monitor comprising"*

687. Riddles discloses all elements of this preamble. This preamble is identical to the preamble of '646 claim 1, and is disclosed by the prior art for the same reasons. I incorporate by reference my discussions regarding the preamble of '646 claim 1 as if fully set forth herein.

- b. *'646 Claim Element 7.1: "a packet acquisition device coupled to the connection point and configured to receive packets passing through the connection point"*

688. Riddle discloses this claim element. This claim element is identical to '646 claim element 1.1, and is disclosed by the prior art for the same reasons. I incorporate by reference my discussions regarding '646 claim element 1.1 as if fully set forth herein.

- c. *'646 Claim Element 7.2: "an input buffer memory coupled to and configured to accept a packet from the packet acquisition device"*

689. Riddle alone discloses this claim element and/or renders it obvious in view

of Ferdinand. As discussed with respect to '099 claim element 1.1 in Section VII.A.2.b, Riddle teaches buffering packets in a router's queue in preparation for parsing and examination.⁸⁷⁷ I incorporate by reference that discussion as if fully set forth herein. For example, Riddle teaches that its monitor includes storage subsystem 35 of interface 40 for storing data.⁸⁷⁸

690. To the extent Riddle does not disclose an input buffer memory, a POSITA would have been motivated and found it obvious to modify Riddle's memory storage with an input buffer based upon a POSITA's own knowledge of network devices and/or the teachings of Ferdinand. Before the priority date of the Challenged Patents, a POSITA would have known that an input buffer memory, such as queues, were found in every routing device because, for example, a POSITA would have understood that an input buffer memory temporarily stores incoming packets until the device is ready to process the packets. In doing so, the input buffer memory avoids packet loss because it provides a mechanism to store packets that may otherwise be dropped. Ferdinand discloses an exemplary input memory, such a frame buffer, which is used to accept packets in network monitors:

The available memory is divided into four blocks during system initializa-

⁸⁷⁷ Riddle, 2:51-54, 6:1-23, 7:21-24, claim 8, Figs. 1A-1B; Ex. 1027 (Packer Application, Appendices, incorporated-by-reference into Riddle), 71-72.

⁸⁷⁸ Riddle, 6:1-23.

tion. *One block includes receive frame buffers. They are used for receiving LAN traffic and for receiving secondary link traffic.* These are organized as linked lists of fixed sized buffers.⁸⁷⁹

691. Based on Ferdinand's teachings, a POSITA would have been motivated to modify Riddle's monitor with an input buffer memory so as to temporarily store received packets and to improve performance by limiting packet drops. And using a buffer with Riddle's monitor amounts to nothing more than an obvious implementation to temporarily store packets in Riddle's monitor based on Ferdinand's teachings.

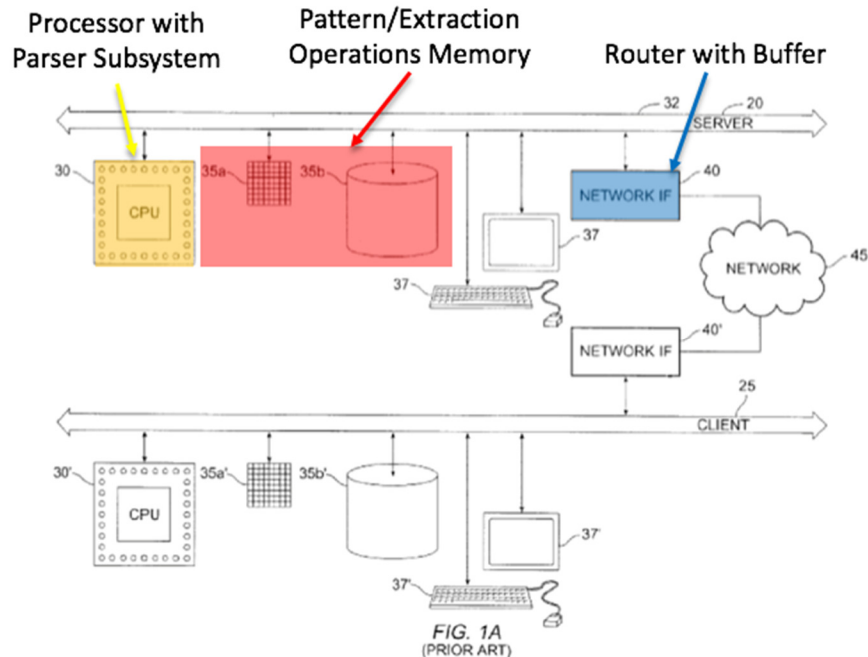
d. '646 Claim Element 7.3: "a parser subsystem coupled to the input buffer memory, the parsing subsystem configured to extract selected portions of the accepted packet and to output a parser record containing the selected portions"

692. Riddle discloses this claim element. As discussed with respect to '099 claim elements 1.3 and 1.4 in Sections VII.A.2.d and VII.A.2.e, Riddle teaches a parser subsystem configured to extract selected portions of the accepted packet.⁸⁸⁰ I incorporate by reference those discussions as if fully set forth herein. For example, Riddle discloses a processor programmed to perform extraction operations as shown below in Figure 1A.⁸⁸¹

⁸⁷⁹ Ferdinand, 26:2-7, 41:17-31, 49:11-12.

⁸⁸⁰ Riddle, 6:1-15, 12:26-53, claim 8, Figs. 1A, 4A.

⁸⁸¹ Riddle, claim 8, Fig. 1A.



- (1) Riddle discloses the claimed “parser subsystem configured to extract select portions of the accepted packet”

693. Riddle explains that its parser subsystem operates to apply “individual instances of traffic classification paradigms to packet network flows based on selectable information obtained from a plurality of layers of a multi-layered communication protocol in order to define a characteristic class, then mapping the flow to the defined traffic class.”⁸⁸² As part of this examination and extraction process, Riddle’s system employs a relational database (knowledge base 306) to store heuristics (i.e., operations) for determining traffic classes based on parsing/extracting portions of a packet and matching those portions’ attributes to a traffic class.⁸⁸³

⁸⁸² Riddle, 4:10-15.

⁸⁸³ Riddle, 12:26-41, 9:28-42, 9:48-49.

694. As shown in Figure 4A's flowchart, Riddle's traffic classification monitor includes parsing and extracting packet portions: "parse flow specification from a packet of the flow" (step 402), "traffic matches a class?" (step 406), and "enter into saved list characteristics of the traffic" (step 408).⁸⁸⁴ Following the parsing, rules are checked and if the flow matches a traffic class an entry is made into a saved list with the extracted identifying information such as "protocol type (SAP), IP protocol number, server port, traffic type if known, MIME type, a time of occurrence of the traffic."⁸⁸⁵

695. And Riddle specifies extracting flow specification portions from the flows:

A method for automatically classifying traffic in a packet communications network, said network having any number of flows, including zero, comprising the steps of: parsing a packet into a first flow specification, wherein said first flow specification contains at least one instance of any one of the following: a protocol family designation, a direction of packet flow designation, a protocol type designation, a pair of hosts, a pair of ports, in HTTP protocol packets, a pointer to a MIME type.⁸⁸⁶

As previously discussed, a POSITA would have understood that Riddle's "pair of hosts" refers to the network-layer source and destination addresses (e.g., IP addresses) and the "pair of ports" refers to the transport-layer source and destination

⁸⁸⁴ Riddle, 12:42-53, Fig. 4A.

⁸⁸⁵ Riddle, 12:50-53.

⁸⁸⁶ Riddle, claims 1, 11.

port numbers.

696. Similarly, Riddle describes extracting information from a packet for Figure 4B's step 422 (determining whether traffic is well known), step 423 (determining whether traffic belongs to a server connection port of an unregistered IP port), and step 426 (whether traffic belongs to a service aggregate).⁸⁸⁷

- (2) Riddle discloses the claimed “parser subsystem configured ... to output a parser record containing the selected portions”

697. The '646 Patent describes that a “parser record” may include a signature, a hash, the packet itself, flags related to the packet, or parts of the packet's payload:

In one embodiment, the parser passes data from the packet—a parser record—that includes the signature (i.e., selected portions of the packet), the hash, and the packet itself to allow for any state processing that requires further data from the packet. An improved embodiment of the parser subsystem might generate a parser record that has some predefined structure and that includes the signature, the hash, some flags related to some of the fields in the parser record, and parts of the packet's payload that the parser subsystem has determined might be required for further processing, e.g., for state processing.⁸⁸⁸

698. Riddle discloses that its parser subsystem extracts selected portions of the accepted packet and outputs a parser record containing the selected portions in the

⁸⁸⁷ Riddle, 13:36-62, Fig. 4B.

⁸⁸⁸ '646 Patent, 9:29-39.

same way. As shown in Figure 4A's flowchart, Riddle teaches flow parsing (step 402) in order to compare the flow with traffic classes (steps 404 and 406). Riddle further describes extracting identifying characteristics, i.e. signature, for the flow.⁸⁸⁹ And Riddle specifies parsing and extracting portions of the received packet. For example, Riddle discloses extracting patterns and/or reference strings from headers.⁸⁹⁰

699. To the extent Patentee asserts that the parser record requires hashing a signature, i.e., identifying characteristics, for the packet, such a modification would have been obvious to a POSITA based upon a POSITA's own knowledge and Riddle's disclosures. For example, the '646 Patent states that hashing signatures and the benefits of doing so were well known to a POSITA.⁸⁹¹ This is demonstrated by Riddle. As discussed above in Section IV.A.1, Riddle incorporates-by-reference Packer as though fully set forth in Riddle. Packer teaches using hash tables 402 and 408 to index flows for TCP connections.⁸⁹² A POSITA would have known that a hash table of TCP flows, like Packer's, uses a hash function for identifying characteristics of the TCP connection (such as source address, destination address, and/or ports). And a POSITA would have known that one benefit of using a hash function

⁸⁸⁹ Riddle, 12:42-59.

⁸⁹⁰ Riddle, 8:67-9:27.

⁸⁹¹ '646 Patent, 9:22-28.

⁸⁹² Ex. 1031 (Packer, incorporated-by-reference into Riddle), 15:43-51, Fig. 4A.

is to decrease lookup times. Thus, the addition of a hash function to Riddle's parser record is nothing more than the use of a known technique to improve similar devices in the same way.

700. Further, a POSITA would have understood that Riddle's packet acquisition device, memory, and parser subsystem work together to achieve the goal of classifying traffic.⁸⁹³ Thus, a POSITA would have understood that the parser subsystem is coupled to the input buffer memory so that packets can be read and contents extracted and processed.

- e. '646 Claim Element 7.4: "a memory [for] storing a database of one or more flow-entries for any previously encountered conversational flows, each flow-entry identified by identifying information stored in the flow-entry"*

701. Riddle renders obvious this element alone or in view of Ferdinand. This claim element is similar to '646 claim element 1.2, and is disclosed or rendered obvious by the prior art for the same reasons. I incorporate by reference my discussion regarding '646 claim element 1.2 as if fully set forth herein.

702. Riddle teaches the claimed "each flow-entry identified by identifying information stored in the flow-entry." For example, Riddle discloses entering characteristics of the traffic into the saved list (step 408).⁸⁹⁴ As discussed regarding '646 claim element 1.2, Riddle discloses identifying conversational flows in the form of

⁸⁹³ Riddle, 4:15-17.

⁸⁹⁴ Riddle, 12:42-59, Fig. 4A.

identifying service aggregate flows and PointCast flows. And Riddle discloses that each flow-entry includes a flow specification and indicators, i.e., identifying information, to identify entries.⁸⁹⁵

703. To the extent Patentee asserts '646 claim elements 7.2 and 7.4 require separate physical memories, i.e., one input buffer memory and one memory for storing a database, the combination of Riddle and Ferdinand renders these claim elements obvious. Regarding the input buffer memory, the combination of Riddle and Ferdinand would have resulted in a distinct input buffer memory as described regarding '646 claim element 7.2.

704. Riddle already discloses two distinct memories, a memory subsystem 35a and a file storage subsystem 35b.⁸⁹⁶ Thus, the combination of Riddle and Ferdinand would have resulted in at least two separate memories. Using different memories for different functionalities was well understood by a POSITA as confirmed by Ferdinand, which discloses dividing memory into four separate blocks with each block devoted to a different functionality.⁸⁹⁷ As discussed above, the advantages of using different memories for different functionalities was well known to a POSITA. And a POSITA would have been motivated to provide Riddle's high

⁸⁹⁵ Riddle, 12:42-50, claims 1-2, 5, 8, 11, Figs. 4A-4B.

⁸⁹⁶ Riddle, 6:5-8, Fig. 1A.

⁸⁹⁷ Ferdinand, 26:2-18.

speed, real-time monitor with dedicated memories because, for example, dedicating memories to processors or processing functions ensures memory access times are reduced and system performance increased due to reduced contention for memory.

- f. '646 Claim Element 7.5: “a lookup engine coupled to the output of the parser subsystem and to the flow-entry memory and configured to lookup whether the particular packet whose parser record is output by the parser subsystem has a matching flow-entry, the looking up using at least some of the selected packet portions and determining if the packet is of an existing flow”*

705. Riddle discloses this claim element. This claim element is substantially similar to '646 claim element 1.4, and is disclosed by Riddle for the reasons previously discussed. I incorporate by reference my discussion regarding '646 claim element 1.4 as if fully set forth herein. For example, Riddle's Figure 4B depicts looking up whether a flow matches a traffic class in relation to classifying a service aggregate based on a plurality of indicators.⁸⁹⁸ As another example, Riddle's Figure 4A depicts a flowchart that describes parsing packet flows and automatically classifying those packets.⁸⁹⁹ Figure 4A includes step 410 where duplicates are suppressed. By suppressing duplicates, Riddle's monitor looks up whether the particular packet whose parser record is output by the parser subsystem has a matching flow-entry,

⁸⁹⁸ Riddle, 13:42-47, claim 5, Fig. 4B.

⁸⁹⁹ Riddle, 12:42-63, Fig. 4A.

by way of using at least some of the selected packet portions (i.e., identifying characteristics) and determining if the packet is of an existing flow.⁹⁰⁰

706. As discussed regarding '646 claim element 1.4, a POSITA would have understood that Riddle's lookup engine is a processor and that corresponding code performs the functions discussed above.⁹⁰¹ For example, Riddle's claim 8 recites that the matching of a parsed flow specification to a traffic class may be performed by a "processor means." Further, a POSITA would have understood that Riddle's lookup engine, parser subsystem, and memory work together to achieve the goal of classifying traffic. Thus, a POSITA would have understood that Riddle's lookup engine is coupled to the output of the parser subsystem and to the flow-entry memory.

- g. *'646 Claim Element 7.6: "a cache subsystem coupled to and between the lookup engine and the flow-entry database memory providing for fast access of a set of likely-to-be-accessed flow-entries from the flow-entry database; and"*

707. Riddle in view of Ferdinand and/or Wakeman renders obvious this claim element. This claim element is substantially similar to '646 claim elements 1.3 and 1.4 and is disclosed in the prior art for the same reasons previously discussed. I incorporate by reference my discussion from '646 claim elements 1.3 and 1.4 as if

⁹⁰⁰ Riddle, 12:43-59, Fig. 4A (steps 402, 408, 410).

⁹⁰¹ Riddle, 5:53-57.

fully set forth herein.

708. As discussed in '646 claim element 1.3, a POSITA would have found it obvious to modify Riddle with a cache subsystem to improve performance based upon the admitted prior art, Ferdinand, Wakeman, or the knowledge of a POSITA. Ferdinand teaches using frame and transmit buffers as well as a cache couple to the database. Before the priority date of the Challenged Patents, a POSITA would have known that caches, such as queues, were found in every routing device. A POSITA would have been motivated to modify Riddle's database memory with a cache, as taught by Ferdinand, because caches were well known to reduce look-up times. Based on Ferdinand's teachings, a POSITA would have appreciated the simple implementation of coupling a cache to Riddle's lookup engine and flow-entry storage for fast access to the flow-entries. Using a cache with Riddle's modified flow-entry database amounts to nothing more than an obvious implementation based on Ferdinand's teachings.

709. Further, a POSITA would have understood that the combination of Riddle and Ferdinand results in the cache subsystem being coupled to and between the lookup engine and the flow-entry database in order to achieve Riddle's goal of classifying traffic. A POSITA would have found it logical to position the cache subsystem between the lookup engine and flow-entry database to achieve Riddle's

goals.⁹⁰²

710. The combination of Riddle and Ferdinand would have provided for fast access of a set of likely-to-be-accessed flow-entries. As discussed above, it was well known in the art that caches reduce look-up times.⁹⁰³ And Ferdinand teaches its caches allow for fast access to rewritable memory.⁹⁰⁴ In view of Ferdinand, a POSITA would have appreciated that a cache was easily modified to store Riddle's flow-entries for fast access to likely-to-be-accessed flow-entries.

711. Similarly, in view of Wakeman, a POSITA would have been motivated to modify Riddle's flow-entry storage with a cache in order to, for example, reduce look-up times. And a POSITA would have found it logical to position the cache subsystem between the lookup engine and a flow-entry database to achieve Riddle's goals.

712. The combination of Riddle and Wakeman would have provided for fast access of a set of likely-to-be-accessed flow-entries. Wakeman discloses that its CAM-cache provides fast access:

Since the accessing speed of CAM cache is much faster than that of FDB 12, the inclusion of CAM caches 31-34 in a network switch as described

⁹⁰² '646 Patent, 2:36-51; '646 Prosecution History, 193-200 (09/10/2003 Office Action, p.3).

⁹⁰³ '646 Patent, 2:36-51; '646 Prosecution History, 197-198 (09/10/2003 Office Action, p.7).

⁹⁰⁴ Ferdinand, 18:25-29, 28:16-20.

above results in an increase in forwarding speed. Note that although the FDB 12 in switch 20 is preferably a RAM, CAM caches 31-34 will decrease the access time and thus increase forwarding speeds irrespective of the particular construction of FDB 12 (e.g., where FDB 12 is a hardware CAM as opposed to RAM).⁹⁰⁵

The packet is then forwarded to the destination port, and the CAM-cache is updated with this mapping so that succeeding packets having the same destination address-to-port mapping may be forwarded to the destination port by accessing only the fast CAM-cache and, by eliminating the need to access the much slower forwarding database, increasing the forwarding speed of the switch.⁹⁰⁶

Applicants have found that where CAM caches 31-34 are of such a size that approximately 90% of all packet forwarding is service by CAM caches 31-34 without resort to FDB 12, switch 20 achieves forwarding speeds as much as ten times faster as compared to with conventional network switches utilizing only a RAM FDB 12.⁹⁰⁷

713. Further, Wakeman describes the CAM's fast access is for a set of likely-to-be-accessed entries. Wakeman discloses that learning/aging logic 38 is responsible for determining what entries are stored in the CAM-cache.⁹⁰⁸ When CAM's memory is full and entries need to be removed, Wakeman teaches that learning/ag-

⁹⁰⁵ Wakeman, 4:5-13.

⁹⁰⁶ Wakeman, Abstract.

⁹⁰⁷ Wakeman, 5:22-27.

⁹⁰⁸ Wakeman, 4:31-40, Fig. 3.

ing logic 38 removes either: (1) the least frequently access entry or (2) the least recently accessed entry. In doing so, Wakeman's learning/aging logic ensures that the CAM-cache contains entries that are likely to be accessed, such frequently-accessed entries or most-recently-accessed entries.⁹⁰⁹

714. As such, a POSITA would have understood that the previously described inclusion of CAM-cache into Riddle's monitor, based on Wakeman's teachings, provides fast access and would also be for a set of likely-to-be-accessed flow-entries.

h. '646 Claim Element 7.7: "a flow insertion engine coupled to the flow-entry memory and to the lookup engine and configured to create a flow-entry in the flow-entry database, the flow-entry including identifying information for future packets to be identified with the new flow-entry"

715. Riddle discloses this claim element. As discussed regarding '646 claim element 7.4, it would have been obvious to store Riddle's flow-entry lists in a flow-entry database.⁹¹⁰ For example, as discussed regarding '646 claim element 7.3, Riddle teaches parsing traffic to extract identifying information (step 402), comparing that information to traffic specifications (step 404), determining whether the traffic matches one of the classes being classified (step 406), and entering the identifying characteristics into a saved list (step 408).⁹¹¹

⁹⁰⁹ Wakeman, 4:31-40.

⁹¹⁰ Riddle, 12:37-38, Fig. 3.

⁹¹¹ Riddle, 12:42-59, Fig. 4A.

716. Riddle's step 408 shows that its monitor creates a flow-entry in the flow-entry database with identifying information. Further, Riddle's step 410 ("suppress duplicates") shows that monitor uses the identifying information for future packets to be identified with the flow-entry because duplicates can only be suppressed if they are identified.

717. A POSITA would have understood that Riddle's flow insertion engine corresponds to Riddle's processor and the corresponding code performing the functions discussed above: "The method for automatically classifying heterogeneous packets in a packet telecommunications environment of the present invention is implemented in the C programming language and is operational on a computer system such as shown in FIG. 1A."⁹¹² Further, a POSITA would have understood that Riddle's flow insertion engine, lookup engine, and memory work together to achieve the goal of classifying traffic. Thus, a POSITA would have understood that Riddle's flow insertion engine is coupled to the flow-entry memory and to the lookup engine.

718. To the extent Patentee asserts that the "parser subsystem," "lookup engine," and "flow insertion engine" of '646 claim element 7.3, 7.5, and 7.7-7.10 require separate pieces of hardware, it would have been obvious to a POSITA to modify Riddle's processor and programming code to be separate hardware components.

⁹¹² Riddle, 5:53-57.

This is because using dedicated hardware for various functions, especially functions as common as parsing, data lookup, and protocol/state identification, would have been readily understood by a POSITA. The Challenged Patents acknowledge that such a modification would have been obvious to a POSITA:

Each of the individual hardware elements through which the data flows in the system are now described with reference to FIGS. 10 and 11. Note that while we are describing a particular hardware implementation of the invention embodiment of FIG. 3, *it would be clear to one skilled in the art that the flow of FIG. 3 may alternatively be implemented in software running on one or more general-purpose processors, or only partly implemented in hardware.* An implementation of the invention that can operate in software is shown in FIG. 14. The hardware embodiment (FIGS. 10 and 11) can operate at over a million packets per second, while the software system of FIG. 14 may be suitable for slower networks. *To one skilled in the art it would be clear that more and more of the system may be implemented in software as processors become faster.*⁹¹³

719. Further, Ferdinand discloses its monitor can include separate hardware components for performing various functions, such real time parser (RTP) 32, database 36, boot/load 22, and memory transport module 34, event manager 38, and control module 42.⁹¹⁴ As provided below, Ferdinand's Figure 5 illustrates its monitor having separate hardware components.

⁹¹³ '646 Patent, 17:6-21.

⁹¹⁴ Ferdinand, 19:5-13.

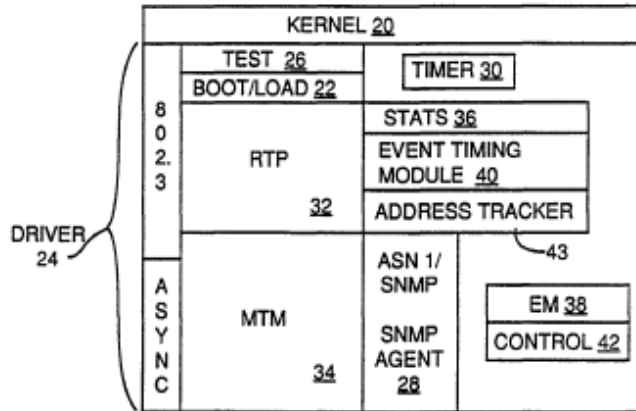


FIG 5

720. Desiring increased performance, a POSITA would have been motivated to utilize dedicated hardware components for parsing, lookups, and flow insertions. On the other hand, a POSITA would have understood that Riddle’s use of a processor for these functions is less expensive and a more extensible solution than using dedicated hardware components.

- i. ’646 Claim Element 7.8: “the lookup engine configured such that if the packet is of an existing flow, the monitor classifies the packet as belonging to the found existing flow; and”

721. Riddle discloses this claim element. As discussed in detail with respect to ’646 claim element 7.5, Riddle teaches a lookup engine and the classification of traffic.

722. If the examined packet is of an existing conversational flow, Riddle discloses that its monitor classifies the packet as belonging to the found existing flow

via the detection and suppression of duplicates:

FIG. 4A depicts a flowchart 401 of processing steps for automatically classifying traffic. In a step 402, a flow specification is parsed from the flow being classified. Then in a step 404, the flow specification parsed from the flow in step 402 is compared with the traffic specifications in each node of the classification tree. Rules are checked starting from most specific to least specific. In a decisional step 406, a determination is made if traffic matches one of the classes being classified. If this is so, then in a step 408, an entry is made in a list of identifying characteristics, such as protocol type (SAP), IP protocol number, server port, traffic type if known, MIME type, a time of occurrence of the traffic. *In an optional step 410, duplicate instances having the same identifying characteristics are suppressed, in favor of keeping a count of the duplicates and a most recent time traffic with these identifying characteristics was encountered.* In an optional step 412, a byte count of traffic of this type has been detected is included.⁹¹⁵

723. Further, Riddle's flowcharts detail the monitor will "parse flow specification from a packet of the flow" (step 402), "compare flow specification with existing classification tree" (step 404), determine if "traffic matches a class?" (step 406), "enter into a saved list characteristics of the traffic" (step 408), "suppress duplicates" (step 410), and determine if "saved traffic belongs to a service aggregate?" (step 426).⁹¹⁶ For example, at Figure 4A's steps 410 and 412, Riddle checks if the

⁹¹⁵ Riddle, 12:42-59.

⁹¹⁶ Riddle, Figures 4A-4B.

flow is a new flow or an existing flow (e.g., suppressing duplicates for existing flows).⁹¹⁷ If the traffic does belong to a service aggregate, then in a step 428, a “traffic class is created which will match all components of the service aggregate.”⁹¹⁸

- j. *'646 Claim Element 7.9: “if the packet is of a new flow, the flow insertion engine stores a new flow-entry for the new flow in the flow-entry database, including identifying information for future packets to be identified with the new flow-entry”*

724. Riddle discloses this claim element. As discussed in detail with respect to '646 claim element 7.7, which is incorporated by reference as if fully set forth herein, Riddle teaches its flow insertion engine is coupled to the lookup engine for creating new flow-entries.

725. As discussed above regarding '646 claim element 1.5, Riddle describes identifying packet flows using a classification tree. Riddle's claims 1-3, 8, and 11 detail creating new tree nodes if a packet is of a new conversational flow for identifying future packets with the new conversational flows. For example, Riddle's claim 1 recites:

A method for automatically classifying traffic in a packet communications network, said network having any number of flows, including

⁹¹⁷ Riddle, 12:53-60, Fig. 4A.

⁹¹⁸ Riddle, 13:52-61, Fig. 4B.

zero, comprising the steps of: parsing a packet into a first flow specification ... thereupon, matching the first flow specification of the parsing step to a plurality of classes represented by a plurality nodes of a classification tree type, each said classification tree type node having a traffic specification; thereupon, *if a matching classification tree type node was not found in the matching step, associating said first flow specification with one or more newly-created classification tree type nodes; thereupon, incorporating said newly-created classification tree type nodes into said plurality of classification tree type nodes.*

726. As shown in Figure 4A's flowchart, Riddle teaches that flow-entries include information to identify future packets with stored flow-entries:

In a decisional step 406, a determination is made if traffic matches one of the classes being classified. *If this is so, then in a step 408, an entry is made in a list of identifying characteristics, such as protocol type (SAP), IP protocol number, server port, traffic type if known, MIME type, a time of occurrence of the traffic.* In an optional step 410, duplicate instances having the same identifying characteristics are suppressed, in favor of keeping a count of the duplicates and a most recent time traffic with these identifying characteristics was encountered. In an optional step 412, a byte count of traffic of this type has been detected is included.⁹¹⁹

727. Riddle's flowchart details the monitor will "parse flow specification from a packet of the flow" (step 402), "compare flow specification with existing classification tree" (step 404), determine if "traffic matches a class?" (step 406), "enter

⁹¹⁹ Riddle, 12:42-59.

into a saved list characteristics of the traffic” (step 408), “suppress duplicates” (step 410), and “determine byte count for traffic and include with traffic specification in saved list” (step 412).⁹²⁰ Riddle’s step 408 shows checking saved flow-entry list to determine whether a flow corresponds to a saved flow is a new flow. And Riddle’s step 410 shows using identification information for future packets to be identified with the stored flow-entries because step 410’s duplicates can only be suppressed if they are identified.

728. Regarding identifying service aggregate flows in Figure 4B (annotated below), Riddle details storing a new flow-entry for a new conversational flow includes information to identify future packets with the new flow-entry:

In decisional step 426, the instance of saved traffic is examined to determine whether it belongs to a service aggregate. For example, an FTP session has one flow that is used to exchange commands and responses and a second flow that is used to transport data files. If the traffic does belong to a service aggregate, then in a step 428, *a traffic class is created which will match all components of the service aggregate. In a further step 425, a new traffic class is created to match the instance of saved traffic.*⁹²¹

⁹²⁰ Riddle, Figure 4A.

⁹²¹ Riddle, 13:53-61, 15:16-27.

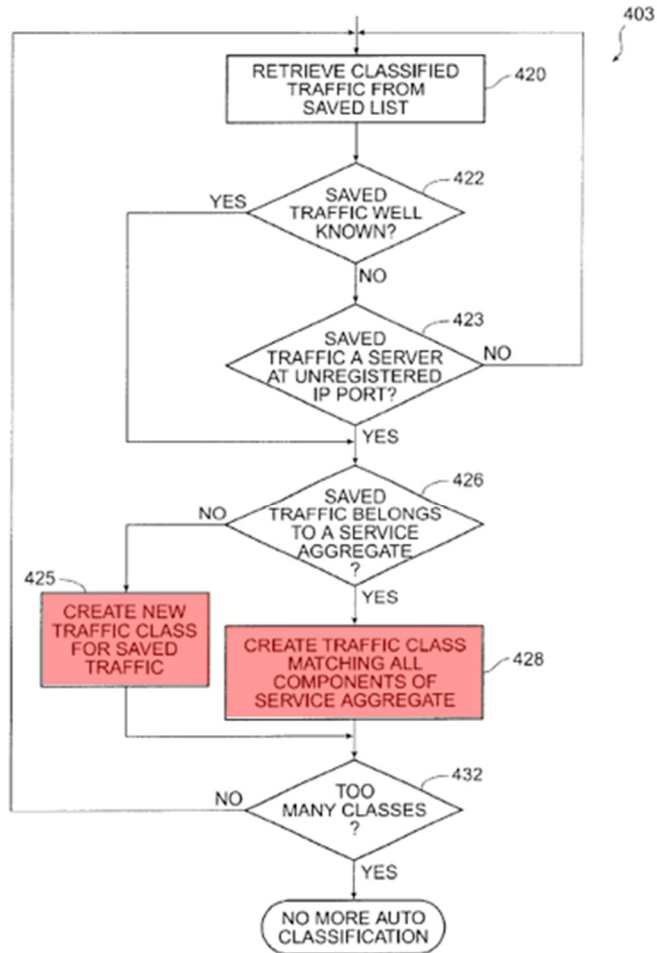


FIG. 4B

- k. '646 Claim Element 7.10: “wherein the operation of the parser subsystem depends on one or more of the protocols to which the packet conforms”

729. Riddle discloses this claim element. Riddle specifies that its parser subsystem operates to apply “individual instances of traffic classification paradigms to packet network flows based on selectable information obtained from a plurality of layers of a multi-layered communication protocol in order to define a characteristic

class, then mapping the flow to the defined traffic class.”⁹²² As part of this operation, Riddle discloses a relational database (knowledge base 306) to store heuristics (i.e., operations) for determining traffic classes based on parsing/extracting portions of a packet and matching those portion’s attributes to a traffic class.⁹²³ For example, Riddle discloses detecting “Marimba and pointcast” traffic by “looking into the data for a signature content header in the get request” and Real Time Protocol (RTP) traffic by determining “the identity of the creator of the connection.”⁹²⁴

730. As a result, a POSITA would have understood that the operation of Riddle’s parser subsystem depends on one or more of the protocols to which the examined packet conforms.

731. Riddle’s parsing varies based upon the protocol since Riddle discloses classifying traffic at any level:

Traffic classes may be defined at any level of the IP protocol as well as for other non-IP protocols. For example, at the IP level, traffic may be defined as only those flows between a specified [sic] set of inside and outside IP addresses or domain names. An example of such a low level traffic class definition would be all traffic between my network and other corporate offices throughout the Internet. *At the application level, traffic classes may be defined for specific URIs within a web server. Traffic*

⁹²² Riddle, Abstract, 4:10-15.

⁹²³ Riddle, 12:26-41, 9:28-42, 9:48-49.

⁹²⁴ Riddle, 11:48-67, 12:3-12.

classes may be defined having “Web aware” class attributes. For example, a traffic class could be created such as all URIs matching “*.html” for all servers, or all URI patterns matching “*.gif” for server X, or for access to server Y with URI pattern “/sales/*” from client Z, wherein ‘*’ is a wildcard character, i.e., a character which matches all other character combinations. Traffic class attributes left unspecified will simply match any value for that attribute. For example, a traffic class that accesses data objects within a certain directory path of a web server is specified by a URI pattern of the directory path to be managed, e.g. “/sales/*” . . .

The present invention provides a method for classifying traffic according to a definable set of classification attributes selectable by the manager, including selecting a subset of traffic of interest to be classified. The invention provides the ability to classify and search traffic based upon multiple orthogonal classification attributes.

Traffic class membership may be hierarchical. Thus, a flow may be classified by a series of steps through a traffic class tree, with the last step (i.e., at the leaves on the classification tree) mapping the flow to a policy. The policy is a rule of assignment for flows. *Web traffic may also be classified by HTTP header types such as Content-Type (MIME type) or User-Agent.*⁹²⁵

732. Riddle’s Table 2 provides examples of information used to build traffic classes.⁹²⁶ Based on Table 2, an exemplary traffic class may be defined depending on one or more protocols to which the packet conforms.

⁹²⁵ Riddle, 8:47-9:27.

⁹²⁶ Riddle, 9:64-65.

TABLE 2

<u>Components of a Traffic Class Specifier</u>		
Inside (Client or Server)	Global	Outside (Server or Client)
IP Address/Domain Name	TCP or UDP Service	IP Address/Domain Name
Port Number	e.g., WWW,	Port Number
MAC Address	FTP, RealAudio, etc. URI pattern for Web Service, MIME type for Web Service IPX Service SNA Service LAT Service IP precedence	MAC Address

733. Thus, in order to determine if traffic matches a class at any level in the IP protocol stack, a POSITA would have understood that the operation of Riddle's parsing subsystem depends on the one or more of the protocols of the received packets.

734. For the above reasons, it is my opinion that Riddle in view of Ferdinand renders obvious claim 7 of the '646 Patent.

6. Independent '646 Claim 16

735. It is my opinion that independent claim 16 of the '646 Patent is obvious in light of Riddle in view of Ferdinand and in light of Riddle in view of Ferdinand and Wakeman.

- a. *'646 Claim 16's Preamble: "A method of examining packets passing through a connection point on a computer network, each packets [sic] conforming to one or more protocols, the method comprising"*

736. Riddle discloses all elements of this preamble. This preamble is similar to the preambles of '646 claims 1 and 7, and is disclosed by the prior art for the same reasons. I incorporate by reference my discussions regarding the preambles of '646 claims 1 and 7 as if fully set forth herein. The prior art's substantive disclosures are located above in my discussion regarding the preamble of '646 claim 1.

- b. *'646 Claim Element 16.1: "(a) receiving a packet from a packet acquisition device"*

737. Riddle discloses this claim element. This claim element is similar to '646 claim elements 1.1 and 7.1, and is disclosed by the prior art for the same reasons. I incorporate by reference my discussions regarding '646 claim elements 1.1 and 7.1 as if fully set forth herein. The prior art's substantive disclosures are located above in my discussion regarding '646 claim element 1.1.

- c. *'646 Claim Element 16.2: "(b) performing one or more parsing/extraction operations on the packet to create a parser record comprising a function of selected portions of the packet"*

738. Riddle discloses this claim element. This claim element is similar to '646 claim element 7.3, and is disclosed by the prior art for the same reasons. I incorporate by reference my discussions regarding '646 claim element 7.3 as if fully set

forth herein. The prior art's substantive disclosures are located above in my discussion regarding '646 claim element 7.3. For example, Riddle teaches performing parsing/extraction operations on packets that create a parser record that includes a flow specification of the packet. Riddle's flow specification is a function of the selected portions of the examined packet.

- d. '646 Claim Element 16.3: "(c) looking up a flow-entry database comprising none or more flow-entries for previously encountered conversational flows, the looking up using at least some of the selected packet portions and determining if the packet is of an existing flow, the lookup being via a cache"*

739. Riddle renders obvious this element alone and/or in view of Ferdinand. This claim element is substantially similar to '646 claim elements 1.2, 1.3, and 1.4, and is disclosed or rendered obvious by the prior art for the same reasons. I incorporate by reference my discussions regarding '646 claim elements 1.2 to 1.4 as if fully set forth herein. The prior art's substantive disclosures are located above in my discussion regarding '646 claim elements 1.2 to 1.4. For example, as discussed above, Riddle alone or in view of Ferdinand's teachings would have motivated a POSITA to store Riddle's lists in a flow-entry database that includes flow-entries for previously encountered flows.

740. As discussed regarding '646 claim element 1.2, Riddle discloses identifying conversational flows in the form of identifying service aggregate flows and PointCast flows.

741. As discussed regarding '646 claim element 1.3, 1.4, and 7.6, from the perspective of a POSITA, Riddle in view of Ferdinand and/or Wakeman would have rendered obvious to modify the lookup engine with a cache such that the lookup is via the cache and using at least some of the selected packet portions and determining if the packet is of an existing flow. As discussed regarding '646 claim elements 1.4 and 7.5, Riddle teaches that its monitor determines whether each received packet is of an existing flow by using selected packet portions, i.e., identifying characteristics. In doing so, Riddle describes determining whether a flow belongs to a service aggregate, and tracking packet duplicates and the most recent time traffic with the same identifying characteristics previously encountered. According to Riddle, this allows for determining if the received packet is of an existing conversational flow.

- e. *'646 Claim Element 16.4: “(d) if the packet is of an existing flow, classifying the packet as belonging to the found existing flow; and”*

742. Riddle discloses this claim element. This claim element is substantially similar to '646 claim element 7.8, and is disclosed by the prior art for the same reasons. I incorporate by reference my discussions regarding '646 claim element 7.8 as if fully set forth herein. The prior art's substantive disclosures are located above in my discussion regarding '646 claim element 7.8. For example, Riddle discloses

that its monitor classifies the packet as belonging to the found existing conversational flow via the detecting whether the flow belong to a service aggregate and/or suppressing of duplicates.⁹²⁷

- f. *'646 Claim Element 16.5: “(e) if the packet is of a new flow, storing a new flow-entry for the new flow in the flow- entry database, including identifying information for future packets to be identified with the new flow- entry”*

743. Riddle discloses this claim element or renders it obvious in view of Ferdinand. This claim element is substantially similar to '646 claim element 7.9, and is disclosed by the prior art for the same reasons. I incorporate by reference my discussions regarding '646 claim element 7.9 as if fully set forth herein. The prior art's substantive disclosures are located above in my discussion regarding '646 claim element 7.9. For example, Riddle discloses creating a flow-entry in the flow-entry database for each new conversational flow that includes identifying information.⁹²⁸

- g. *'646 Claim Element 16.6: “wherein the parsing/extraction operations depend on one or more of the protocols to which the packet conforms”*

744. Riddle discloses this claim element. This claim element is substantially simi-

⁹²⁷ Riddle, 12:42-59, 13:36-62, Figs. 4A-4B.

⁹²⁸ E.g., Riddle, 12:42-59, 13:36-62, Figs. 4A-4B.

lar to '646 claim element 7.10, and is disclosed by the prior art for the same reasons. I incorporate by reference my discussions regarding '646 claim element 7.10 as if fully set forth herein. The prior art's substantive disclosures are located above in my discussion regarding '646 claim element 7.10. For example, Riddle discloses that its parsing/extraction operations use a relational database 306 to store heuristics based on information obtained a multi-layered communication protocol to define traffic classes for parsed packet portions.⁹²⁹

745. For the above reasons, it is my opinion that Riddle in view of Ferdinand renders obvious independent claim 16 of the '646 Patent.

7. Dependent '646 Claim 18

746. Riddle discloses all the limitations of this claim. Claim 18 depends from independent claim 16 and recites:

A method according to claim 16, wherein the function of the selected portions of the packet forms a signature that includes the selected packet portions and that can identify future packets, wherein the lookup operation uses the signature and wherein the identifying information stored in the new or updated flow-entry is a signature for identifying future packets.

747. Riddle describes the storage of identifying characteristics extracted from each parsed packet portion that can identify future packets:

⁹²⁹ Riddle, 4:10-15, 12:26-41, 8:47-9:49.

FIG. 4A depicts a flowchart 401 of processing steps for automatically classifying traffic. In a step 402, a flow specification is parsed from the flow being classified. Then in a step 404, the flow specification parsed from the flow in step 402 is compared with the traffic specifications in each node of the classification tree. Rules are checked starting from most specific to least specific. In a decisional step 406, a determination is made if traffic matches one of the classes being classified. *If this is so, then in a step 408, an entry is made in a list of identifying characteristics, such as protocol type (SAP), IP protocol number, server port, traffic type if known, MIME type, a time of occurrence of the traffic.* In an optional step 410, duplicate instances having the same identifying characteristics are suppressed, in favor of keeping a count of the duplicates and a most recent time traffic with these identifying characteristics was encountered. In an optional step 412, a byte count of traffic of this type has been detected is included.⁹³⁰

748. And Riddle's flowchart details that the monitor will "parse flow specification from a packet of the flow" (step 402), "compare flow specification with existing classification tree" (step 404), determine if "traffic matches a class?" (step 406), and "enter into a saved list characteristics of the traffic" (step 408), and "suppress duplicates" (step 410).⁹³¹ Riddle's step 408 describes using identifying information, i.e., a signature, to create a flow-entry in the flow-entry database for new

⁹³⁰ Riddle, 12:42-59.

⁹³¹ Riddle, Fig. 4A.

flows. Riddle's identifying information includes selected portions of the packet, such as protocol type, IP protocol number, server port, traffic type, MIME type, and/or a time of occurrence of the traffic.⁹³² And Riddle's step 410 describes using the stored packet portion's signature to identify future packets to suppress duplicates of previously identified packet flows. With respect to Figure 4B, Riddle details creating a new traffic class for a new service aggregate (i.e., conversational flow).⁹³³

749. To the extent Patentee argues that forming a packet portion signature requires hashing a signature for the packet, such a modification would have been obvious to a POSITA based upon a POSITA's own knowledge and Riddle's disclosures. The '646 Patent states that hashing signatures and the benefits of doing so were well known to a POSITA.⁹³⁴ And like the '646 Patent, Riddle demonstrates hashing signatures. As discussed above in Section IV.A.1, Riddle incorporates-by-reference Packer as though fully set forth in Riddle. Packer teaches using hash tables 402 and 408 to index flows for TCP connections.⁹³⁵ A POSITA would have known that a hash table of TCP flows, like Packer's, uses a hash function for identifying characteristics of the TCP connection (such as source address, destination

⁹³² Riddle, 12:42-59.

⁹³³ Riddle, 11:11-23, 13:36-62 (Fig. 4B's steps 425, 428).

⁹³⁴ '646 Patent, 9:22-28.

⁹³⁵ Ex. 1031 (Packer), 15:43-51, Fig. 4A.

address, and/or ports). And a POSITA would have known one benefit of using a hash function is to decrease lookup times. Thus, the addition of a hash function to Riddle's signature is nothing more than the use of a known technique to improve similar devices in the same way.

750. As such, it is my opinion that Riddle discloses all the limitations of dependent claim 16 of the '646 Patent. And for all the reasons set forth above, it is my opinion that Riddle in view of Ferdinand and Wakeman renders obvious claims 1-3, 7, 16, and 18 of the '646 Patent.

B. For the '646 Patent, Riddle in View of Ferdinand and Wakeman and Further in View of Yu Renders Obvious Claims 1-3, 7, 16, and 18.

751. It is my opinion that a POSITA would have recognized that each and every limitation of the '646 claims 1-3, 7, 16, and 18 is disclosed or rendered obvious in light of Riddle in view of Ferdinand and Wakeman and further in view of Yu. Specifically, my opinions regarding Riddle in view of Ferdinand and Wakeman are exactly the same as those above in Section IX.A, but further include the teachings of Yu. Thus, as if fully set forth here, I incorporate the discussion from Section IX.A regarding the obviousness of '646 Claims 1-3, 7, 16, and 18 over Riddle in view of Ferdinand and Wakeman.

752. As discussed above, all of the Challenged Claims require “conversational

flows.” For example, ’646 claim element 1.2 recites “previously encountered conversational flows to which a received packet may belong.” While Riddle itself teaches identifying conversational flows, Yu further demonstrates identifying conversational flows through its “flow classification.”⁹³⁶ Further, Yu teaches flow classification logic that “keeps track of the flow’s state until matching criteria is met” when identifying whether a packet belongs to a conversational flow.⁹³⁷

753. As discussed with respect to the obviousness of ’099 claims 1 and 2 over Riddle in view of Ferdinand and further in view of Yu in Section VII.C, Yu teaches identifying the claimed “conversational flow” by Yu’s flow classifier linking multiple “streams” into a “flow” based on application or application data.⁹³⁸

And as discussed in Section VII.C, Yu teaches state tracking that binds policy decisions to each stream of a flow so that actions can be taken on future packets without intervention from the “host” application.⁹³⁹ Moreover, as discussed in Section VII.C, Yu specifies using hash values to find corresponding policies to reduce further complicated pattern-matching.⁹⁴⁰ I incorporate by reference that discussion as if fully set forth herein.

⁹³⁶ Yu, 1:56-60, 3:32-36, 3:47-49; 4:1-8.

⁹³⁷ Yu, 4:62-64.

⁹³⁸ Yu, 1:56-60, 3:32-49; 4:1-8.

⁹³⁹ Yu, 4:57-5:13.

⁹⁴⁰ Yu, 4:23-29.

754. As discussed with respect the obviousness of '099 claims 1 and 2 over Riddle in view of Ferdinand and further in view of Yu in Section VII.C, I explain how a POSITA would have been motivated to combine Yu's teachings into Riddle's monitor. I incorporate by reference that discussion as if fully set forth herein.

755. For the same reasons, it is my opinion that combining the teachings of Riddle, Ferdinand, Wakeman, and Yu renders obvious all the claim elements relating to "conversational flows" as well as carrying out state operations with the state processor progressing through a series of states and state operations.

756. As set forth in my analysis of the '646 Patent in Sections IX.A.2 through IX.A.7 above, Riddle, Ferdinand, and Wakeman disclose or render obvious all the remaining elements of '646 claims 1-3, 7, 16, and 18. Thus, it is my opinion that Riddle in view of Ferdinand and Wakeman and further in view of Yu renders obvious '646 claims 1-3, 7, 16, and 18.

C. For the '646 Patent, Riddle in View of Ferdinand and Wakeman and Further in View of RFC1945 Renders Obvious Claims 1-3, 7, 16, and 18.

757. It is my opinion that a POSITA would have recognized that each and every limitation of the '646 claims 1-3, 7, 16, and 18 is disclosed or rendered obvious in light of Riddle in view of Ferdinand and Wakeman, and further in view of RFC1945. Specifically, my opinions regarding Riddle in view of Ferdinand and Wakeman are exactly the same as those above in Section IX.A, but further include

the teachings of RFC1945. Thus, as if fully set forth here, I incorporate the discussion from Section IX.A regarding the obviousness of '646 Claims 1-3, 7, 16, and 18 over Riddle in view of Ferdinand and Wakeman.

758. As discussed above, all of the Challenged Claims require “conversational flows.” For example, '646 claim element 1.2 recites “previously encountered conversational flows to which a received packet may belong.” While Riddle itself teaches identifying conversational flows, RFC1945 further demonstrates identifying conversational flows through the additional example of the use of HTTP header fields.

759. As discussed with respect to the obviousness of '099 claims 1 and 2 in view Riddle, Ferdinand, and RFC1945 in Section VII.E, I understand that Patentee and its technical expert have taken the position that the HTTP Referrer field can be used to create a conversational flow. I incorporate by reference that discussion as if fully set forth herein.

760. As discussed with respect the obviousness of '099 claims 1 and 2 over Riddle in view of Ferdinand and further in view of RFC1945 in Section VII.E, I explain how a POSITA would have been motivated to combine RFC1945's teachings into Riddle's monitor. I incorporate by reference that discussion as if fully set forth herein. For the same reasons, it is my opinion that combining the teachings of Rid-

dle, Ferdinand, Wakeman, and RFC1945 renders obvious all the claim elements relating to “conversational flows,” at least under Patentee’s interpretation for that term.

761. As set forth in my analysis of the ’646 Patent in Sections IX.A.2 through IX.A.7 above, Riddle, Ferdinand, and Wakeman disclose or render obvious all the remaining elements of ’646 claims 1-3, 7, 16, and 18. Thus, it is my opinion Riddle in view of Ferdinand and Wakeman and further in view of RFC1945 renders obvious ’646 claims 1-3, 7, 16, and 18 at least under Patentee’s interpretation of “conversational flow.”

X. THE CLAIMS OF THE ’751 PATENT ARE UNPATENTABLE

762. For the ’751 Patent, the challenged claims include independent claims 1 and 17 as well as dependent claims 2, 5, 10, 14, and 15. As I detail below, it is my opinion that a POSITA would have recognized that each and every limitation of those claims is disclosed or rendered obvious by the prior art. Specifically, it is my opinion that Riddle in view of Ferdinand renders obvious all those claims. It is also my opinion that Riddle in view of Ferdinand and further in view of Yu renders obvious ’751 claims 1, 2, 5, 10, 14, 15, and 17. Moreover, it is my opinion that Riddle in view of Ferdinand and further in view of RFC1945 renders obvious ’751 claims 1, 2, 5, 10, 14, 15, and 17.

A. For the '751 Patent, Riddle in View of Ferdinand Renders Obvious Claims 1, 2, 5, 10, 14, 15, and 17.

763. It is my opinion that a POSITA would have recognized that each and every limitation of '751 claims 1, 2, 5, 10, 14, 15, and 17 is disclosed or rendered obvious by the prior art. Specifically, it is my opinion that '751 claims 1, 2, 5, 10, 14, 16, and 17 are obvious over Riddle in view of Ferdinand.

1. Reasons to Modify Riddle in View of Ferdinand

764. As described above with respect to the '099 Patent in Section VII.A.1, a POSITA would have been motivated and found it obvious to combine the teachings of Riddle and Ferdinand.

2. Independent '751 Claim 1

a. '751 Claim 1's Preamble: "A method of analyzing a flow of packets passing through a connection point on a computer network, the method comprising"

765. Riddle discloses all elements of this preamble. This preamble is similar to the preambles of '646 claim 1 and '099 claim 1, except that they are apparatus claims.⁹⁴¹ '751 claim 1's preamble is disclosed in the prior art for the same reasons, and I incorporate my discussions of the preambles of '646 claim 1 and '099 claim 1 as if fully set forth herein.

⁹⁴¹ '646 claim 1 and '099 claim 1 both recite "[a] packet monitor for examining packet[s] passing through a connection point on a computer network"

- b. *'751 Claim Element 1.1: “(a) receiving a packet from a packet acquisition device coupled to the connection point”*

766. Riddle discloses this claim element, which is similar to '725 element 10.1 and '646 claim element 1.1 and disclosed in the prior art for the same reasons.⁹⁴² I incorporate by reference my discussions of those elements as if fully set forth herein.

- c. *'751 Claim Element 1.2: “(b) for each received packet, looking up a flow-entry database for containing one or more flow-entries for previously encountered conversational flows, the looking up to determine if the received packet is of an existing flow”*

767. Riddle renders obvious this element alone and/or in view of Ferdinand. As I explain in my discussion of '099 claim element 1.5, which I incorporate by reference as if fully set forth herein, Riddle teaches flow-entry lists that stores flow-entries of previously-encountered flows.⁹⁴³ For example, Riddle's Figure 4A shows the process of parsing a flow specification from a packet and then storing the flow

⁹⁴² '725 claim 10's preamble recites a “method of performing protocol specific operations on a packet passing through a connection point on a computer network,” the first step of which—i.e., element 10.1—is “receiving the packet.” '646 claim element 1.1 recites “a packet acquisition device coupled to the connection point and configured to receive packets passing through the connection point.”

⁹⁴³ Riddle, 12:37-59, Fig. 3.

specifications in the saved list.⁹⁴⁴ And Riddle's Figure 4B shows accessing previously-encountered flow-entries stored in the database to determine if the received packet is of an existing flow.⁹⁴⁵

768. As discussed with respect to '099 claim element 1.5 and '646 claim element 1.2, a POSITA would have been motivated and found it obvious to store Riddle's flow-entries in a database based upon a POSITA's own knowledge of network devices or the disclosures in Ferdinand.

769. Further, as discussed with respect to '099 claim element 1.4 and '646 claim element 1.2, Riddle's identification of flows as "service aggregates" and/or classification of separate flows as PointCast traffic teaches the claimed flow-entries for previously encountered "conversational flows."

d. '751 Claim Element 1.3: "a conversational flow including an exchange of a sequence of one or more packets in any direction between two network entities as a result of a particular activity using a particular layered set of one or more network protocols, a conversational flow further having a set of one or more states, including an initial state"

770. Riddle discloses this claim element. In the preceding section for '751 claim element 1.2, I noted at least two ways in which Riddle teaches flow-entries for previously-encountered "conversational flows." For similar reasons, Riddle further

⁹⁴⁴ Riddle, 12:42-59, Fig. 4A.

⁹⁴⁵ Riddle, 13:35-62, Fig. 4B.

discloses the claimed “conversational flow including an exchange of a sequence of one or more packets in any direction between two network entities as a result of a particular activity.” For example, Riddle describes (i) conversational flows for FTP activities between two network entities and (ii) conversational flows for PointCast activities between two network entities.

771. Further, Riddle describes network activity using the claimed “particular layered set of one or more network protocols.” For example, Riddle details using service aggregates (claimed “conversational flows”) for identifying a packet’s set of network protocols:

A service aggregate is provided for certain applications that use more than one connection in a particular conversation between a client and a server. For example, *an FTP client in conversation with an FTP server employs a command channel and a transfer channel, which are distinct TCP sessions on two different ports. In cases where two or three TCP or UDP sessions exist for each conversation between one client and one server, it is useful to provide a common traffic class i.e., the service aggregate, containing the separate conversations.* In practice, these types of conversations are between the same two hosts, but use different ports. According to the invention, a class is created with a plurality of traffic specifications, each matching various component conversations.⁹⁴⁶

⁹⁴⁶ Riddle, 11:10-23; 13:54-61; ’864 Provisional, 69.

As the above passage illustrates, Riddle teaches correlating transport-layer connections (e.g., TCP or UDP connections) to identify an application-layer conversation (e.g., an FTP application). Similarly, Riddle identifies a conversational flow resulting from PointCast activity by relating multiple application-layer HTTP connections with URLs that begin with “/FIDO-1/.”⁹⁴⁷ As was known in the art before the priority date of the Challenged Patents, tag FIDO-1 was used to fetch concatenated connection flows.⁹⁴⁸

772. Each of Riddle’s described protocols (e.g., FTP, HTTP, TCP, UDP) and layers (e.g., application, transport) were well known in the prior art. For example, Riddle’s Figure 1D confirms that the OSI model shown below is “PRIOR ART.”⁹⁴⁹

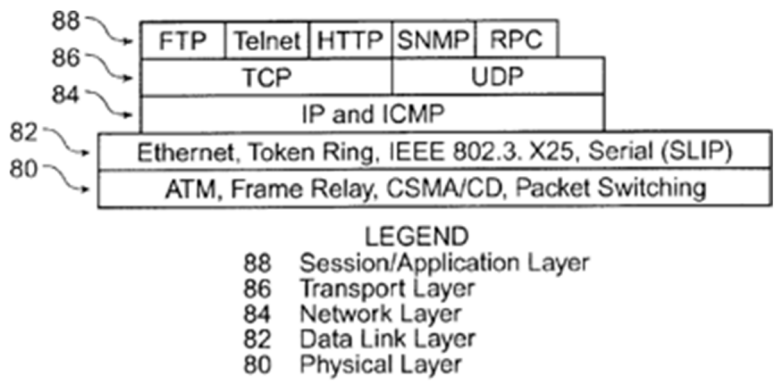


FIG. 1D
(PRIOR ART)

⁹⁴⁷ Riddle, 11:47-67.

⁹⁴⁸ Ex. 1036 (U.S. Patent No. 6,807,558), 30:62-31:17, 33:28-44, 39:14-40:21.

⁹⁴⁹ Riddle, 7:35-8:46, 1D.

773. Additionally, Riddle specifies its conversational flows include the claimed “set of one or more states, including an initial state.” For example, Riddle describes determining the state of the packet’s flow to identify potential duplicates stored in its flow-entry lists:

In an optional step 410, duplicate instances having the same identifying characteristics are suppressed, in favor of keeping a count of the duplicates and a most recent time traffic with these identifying characteristics was encountered. In an optional step 412, a byte count of traffic of this type has been detected is included.⁹⁵⁰

In Figure 4A, Riddle’s flowchart details the monitor will “parse flow specification from a packet of the flow” (step 402), “compare flow specification with existing classification tree” (step 404), determine if “traffic matches a class?” (step 406), “enter into a saved list characteristics of the traffic” (step 408), “suppress duplicates” (step 410), and “determine byte count for traffic and include with traffic specification in saved list” (step 412).

774. The Challenged Patents’ claims show that updating a flow entry is a state operation.⁹⁵¹ When suppressing duplicates, Riddle teaches updating the flow entry

⁹⁵⁰ Riddle, 12:53-59.

⁹⁵¹ E.g., ’646 claim 15 (“wherein the state operations include updating the flow-entry, including identifying information for future packets to be identified with the flow-entry”); ’751 claim 14; ’789 claims 15, 30, 45.

with “a count of the duplicates and a most recent time traffic with these identifying characteristics was encountered.”⁹⁵²

775. Further, the Challenged Patents’ claims show that determining metrics is a state operation.⁹⁵³ Riddle similarly discloses determining metrics, such as a count of the duplicates, a most recent time traffic with these identifying characteristics was encountered, and a byte count of traffic of this type that has been detected.⁹⁵⁴

776. As such, Riddle discloses the claimed “conversational flow further having a set of one or more state, including an initial state.” For the three states discussed above, a POSITA would have understood Riddle’s initial state to be a count of zero duplicates, the time when the initial traffic with these identifying characteristics was received, and the byte count of the initial traffic.

777. Further, as I discuss regarding ’099 claim element 1.7, which I incorporate by reference as if fully set forth herein, Riddle provides an example state transition pattern (e.g., Client IP/Subnet B/FTP/FTP-cmd/FTP service aggregate) for flow belongs to a service aggregate with flows involving FTP applications.⁹⁵⁵ As exemplified in Figure 4B, Riddle’s classification tree includes determining the state of a

⁹⁵² Riddle, 12:53-59, Fig. 4A.

⁹⁵³ E.g., ’751 claim 16 (“wherein one or more metrics related to the state of the flow are determined as part of the state operations specified for the state of the flow”).

⁹⁵⁴ Riddle, 12:53-59, Fig. 4A.

⁹⁵⁵ Riddle, 11:10-23; 13:52-57; Fig. 4B.

service aggregate flow (i.e., “conversational flow”) at Step 426, and determines the initial state of a new service aggregate flow at Step 428.⁹⁵⁶ And Packer, which Riddle incorporates-by-reference, describes a traffic-classification system illustrated in Figure 5F, below.⁹⁵⁷ Packer further describes traversing a set of predefined states by recursively processing though matching child-class definitions.⁹⁵⁸

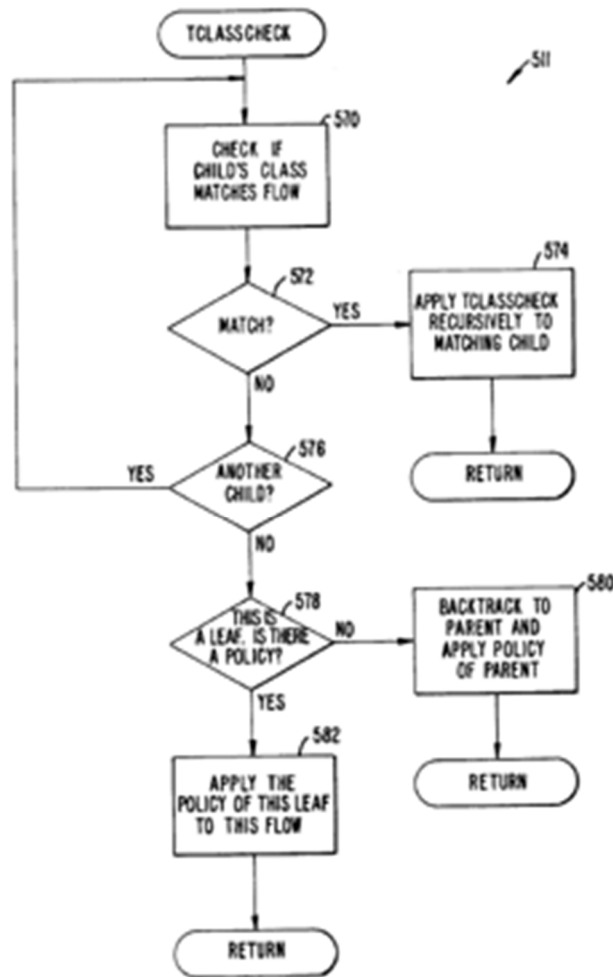


FIG. 5F.

⁹⁵⁶ Riddle, 11:10-23, 13:36-62, Fig. 4B.

⁹⁵⁷ Riddle, 1:38-44.

⁹⁵⁸ Ex. 1031 (Packer), 18:1-26.

- e. *'751 Claim Element 1.4: “(c) if the packet is of an existing flow, identifying the last encountered state of the flow, performing any state operations specified for the state of the flow, and updating the flow-entry of the existing flow including storing one or more statistical measures kept in the flow-entry; and”*

778. Riddle discloses this claim element, which is similar to '646 claim elements 7.8 and 16.4 and disclosed in the prior art for the same reasons. I incorporate by reference my discussions of those elements as if fully set forth herein.

779. As discussed regarding '751 claim element 1.3, Riddle teaches performing the state operation of determining whether flow matches a traffic class in relation to classifying a service aggregate , and determining statistical metrics such as (i) a count of the duplicates, (ii) the most recent time traffic with the same identifying characteristics was encountered, and (iii) a byte count of the detected traffic.⁹⁵⁹ For existing flows, Riddle specifies storing “a count of the duplicates and a most recent time traffic with these identifying characteristics was encountered.”⁹⁶⁰

780. As described in the Challenged Patents, the packet monitor determines the state of the flow based on the relationship of packets and the entire flow.⁹⁶¹ And the Challenged Patents state that the state of the flow includes “parameters such as

⁹⁵⁹ Riddle, 12:42-13:8, 14:1-5, Fig. 4A.

⁹⁶⁰ Riddle, 12:56-57, Fig. 4A.

⁹⁶¹ E.g., '099 Patent, 5:27-34.

the time, length of the conversational flow, data rate, etc.”⁹⁶² Moreover, the Challenged Patents’ claims show that updating a flow entry is a state operation⁹⁶³ and determining metrics is a state operation.⁹⁶⁴

781. Further, Riddle describes displaying a hierarchical classification tree that shows state operations related to FTP. For example, Riddle presents classes “to host 1,” “tcp,” and “FTP.”⁹⁶⁵ As previously discussed, a POSITA would have understood that transitioning from one class to the next in this hierarchy involves state operations.

782. In connection with Figure 4A, Riddle describes analyzing information identifying the characteristics of the traffic as the classifier parses packets of a flow and matches the parsed packets to a class.⁹⁶⁶ Riddle describes that after an initial classification, “sub-classification” proceeds in a sequential manner by performing finer grade matchings as characteristics such as the hosts and services are identified, leading to matching of a flow with operation of a particular application program.⁹⁶⁷

⁹⁶² E.g., ’099 Patent, 5:27-34.

⁹⁶³ E.g., ’646 claim 15 (“wherein the state operations include updating the flow-entry, including identifying information for future packets to be identified with the flow-entry”); ’751 claim 14; ’789 claims 15, 30, 45.

⁹⁶⁴ ’751 claim 16 (“wherein one or more metrics related to the state of the flow are determined as part of the state operations specified for the state of the flow”).

⁹⁶⁵ Riddle, 13:11-22.

⁹⁶⁶ Riddle, 12:42-48.

⁹⁶⁷ Riddle, 11:25-31, 13:11-22.

In accordance with the classification tree, Riddle's classifier advances through the sequence of packets in a particular traffic flow to parse and classify those packets. This results in Riddle's classifier performing a corresponding state operation at each node to update the identifying characteristics of the flow.

783. Accordingly, Riddle teaches this claim element because Riddle's monitor can identify the last encountered state of the flow and perform any state operations specified for the state of the flow. And Riddle teaches updating the flow-entry of the existing flow, including determining whether flow matches a traffic class in relation to classifying a service aggregate, and storing one or more statistical measures kept in the flow-entry by determining and updating metrics such as (i) a count of the duplicates, (ii) the most recent time traffic with the same identifying characteristics was encountered, and (iii) a byte count of the detected traffic.⁹⁶⁸

⁹⁶⁸ Riddle, 12:53-13:8, 14:1-5, Fig. 4A (Step 410 "suppress duplicates" and Step 412 "determine byte count for traffic and include with traffic specification in saved list").

f. '751 Claim Element 1.5: “(d) if the packet is of a new flow, performing any state operations required for the initial state of the new flow and storing a new flow-entry for the new flow in the flow-entry database, including storing one or more statistical measures kept in the flow-entry”

784. Riddle discloses this claim element, which is similar to '646 claim elements 7.9 and 16.5 and disclosed in the prior art for the same reasons. I incorporate by reference my discussions of those elements as if fully set forth herein.

785. As discussed above regarding '751 claim element 1.2, Riddle alone and/or in view of Ferdinand renders obvious a flow-entry database for storing flow-entries.

For example, Riddle describes storing flow-entries in lists 308:

A traffic classifier 304 detects services for incoming traffic. ... A plurality of saved lists 308 stores classified traffic pending incorporation into traffic tree 302. In select embodiments, entries for each instance of traffic may be kept. In alternate embodiments, a copy of an entry and a count of duplicate copies for the entry is maintained.

FIG. 4A depicts a flowchart 401 of processing steps for automatically classifying traffic.... In a decisional step 406, a determination is made if traffic matches one of the classes being classified. *If this is so, then in a step 408, an entry is made in a list of identifying characteristics, such as protocol type (SAP), IP protocol number, server port, traffic type if known, MIME type, a time of occurrence of the traffic.* In an optional step 410, duplicate instances having the same identifying characteristics are suppressed, in favor of keeping a count of the duplicates and a most recent time traffic with these identifying characteristics was encountered. In an optional step 412,

a byte count of traffic of this type has been detected is included....

In an optional step 413 (not show), after the processing of flowchart 401 completes or at periodic intervals or on demand, ***a list of traffic classes produced in steps 402 through 412 are displayed to a network manager. The list may be sorted by any well-known criteria*** such as: 1) most “hits” during a recent interval, 2) most recently-seen (most recent time first), 3) most data transferred (bytes/second) during some interval, or a moving average.⁹⁶⁹

786. For new flow entries, Riddle discloses the state operation of creating a new flow entry such as Figure 4A’s step 408 and a new class in Figure 4B’s step 425 for service aggregate classes. A POSITA would have understood that Riddle’s creation of a new flow entry and a new class are examples of state operations for the initial state of the new flow. Indeed, the Challenged Patents confirm that creating a new flow entry with identifying information is a state operation:

A method ... wherein one of the state operations specified for at least one of the states includes ***creating a new flow-entry*** for future packets to be identified with the flow, the new flow-entry including identifying information for future packets to be identified with the flow-entry.⁹⁷⁰

787. Further, as shown above, Riddle discloses the state operations of updating flow-entries and determining metrics such as (a) determining whether flow matches

⁹⁶⁹ Riddle, 12:27-13:5.

⁹⁷⁰ ’789 Claim 47.

a traffic class in relation to classifying a service aggregate, (b) a count of the duplicates (which would be zero for a new flow), (c) a most recent time traffic with these identifying characteristics was encountered, and (d) a byte count of traffic of this type that has been detected. Riddle teaches these updates and metrics are stored in the saved list.⁹⁷¹ The '751 Patent acknowledges that determining metrics is a state operation.⁹⁷² As such, a POSITA would have understood that each of Riddle's state operations is an example of state operations for the initial state of the new flow.

788. Thus, Riddle discloses that, if the packet is of a new flow, the monitor performs any state operations required for the initial state of the new flow and stores a new flow-entry for the new flow in the flow-entry database, including creating a new traffic class for newly-encountered service aggregate and storing one or more statistical measures kept in the flow-entry.

- g. '751 Claim Element 1.6: "wherein every packet passing through the connection point is received by the packet acquisition device, and"*

789. Riddle discloses this claim element, which is similar to '099 claim element 1.1 and '646 claim elements 1.1 and 7.1 and disclosed in the prior art for the same reasons. I incorporate by reference my discussions of those elements as if fully set forth herein.

⁹⁷¹ Riddle, 12:30-59, 13:36-62.

⁹⁷² '751 Claim 16.

790. As discussed above regarding the preamble of '099 claim 1, Riddle's Figure 1A shows a packet monitor and a packet acquisition device (e.g., network interfaces 40) connected to the connection point.⁹⁷³ And as shown in Figure 1C, Riddle teaches that router 75 contains a network interface and system gateway, which contains a packet acquisition device coupled to the connection point.⁹⁷⁴ As illustrated in the '751 Patent's Figure 1, a POSITA would have understood that the connection point is where the packet monitor connects to the network.⁹⁷⁵

791. Riddle describes creating flow-entries for "each instance of traffic," which indicates all traffic is received by the acquisition device:

A traffic classifier 304 detects services for incoming traffic.... A plurality of saved lists 308 stores classified traffic pending incorporation into traffic tree 302. *In select embodiments, entries for each instance of traffic may be kept. In alternate embodiments, a copy of an entry and a count of duplicate copies for the entry is maintained.*⁹⁷⁶

792. As discussed above in Section IV.A.1, Riddle incorporates-by-reference the teachings of U.S. Patent 6,046,980 ("Packer") as though fully set forth in Riddle.

⁹⁷³ Riddle, 6:9-15, Fig. 1A.

⁹⁷⁴ Riddle, 7:21-34, claim 8, Fig. 1C.

⁹⁷⁵ '751 Patent, 6:3-19, Fig. 1 (showing connection points 121, 123, 125).

⁹⁷⁶ Riddle, 12:30-41.

Like Riddle, Packer describes a traffic classification system and specifies that traffic flows are examined “continuously and automatically”⁹⁷⁷ and that “a determination is made whether the total rate of all flows is greater than a maximum limit.”⁹⁷⁸ This teaching further shows that Riddle discloses examining every packet passing through the connection point.

793. One of the goals of Riddle is “analyzing real traffic in a customer’s network and *automatically* producing a list of the ‘found traffic.’”⁹⁷⁹ To accomplish this goal, a POSITA would have understood that Riddle teaches examining every packet that passes through the connection point and is received by the packet acquisition device before being sent to traffic classifier 304.

794. Similarly, Riddle describes classifying a “complete enumeration of the possible traffic” and providing a default class for “all traffic” that does not match a user-specified class, which creates a new class to match an instance of otherwise non-matching traffic.⁹⁸⁰ By creating a class for otherwise non-matching traffic, along with keeping detailed metrics of traffic and checking for duplicates, a

⁹⁷⁷ Ex. 1031 (Packer), 4:12-16.

⁹⁷⁸ Ex. 1031 (Packer), 18:58-60; 18:61-62.

⁹⁷⁹ Riddle, 4:1-2, Abstract, 10:57-11:9.

⁹⁸⁰ Riddle, 4:16-17, 10:52-56.

POSITA would have understood that Riddle receives every packet passing through the connection point.⁹⁸¹

- h. '751 Claim Element 1.7: "wherein at least one step of the set consisting of of [sic] step (a) and step (b) includes identifying the protocol being used in the packet from a plurality of protocols at a plurality of protocol layer levels"*

795. Riddle discloses this claim element. Riddle specifies that each examined packet conforms to one or more protocols. For example, Riddle states its system maintains "network bandwidth based on information ascertainable from multiple layers of OSI network model."⁹⁸² And Riddle seeks to classify packet traffic "based upon information gathered from ... multiple layers in a multi-layer protocol network."⁹⁸³

796. Riddle's Figure 1D shows the relationship diagram of the well-known prior art layers of the TCP/IP protocol suite. This includes the application layer 88, the transport layer 86, the network layer 84, the data link layer 82, and the physical layer 80, as shown below in Riddle's Figure 1D.

⁹⁸¹ Riddle, 10:52-56, 12:50-59, 13:1-31, 13:59-60.

⁹⁸² Riddle, 1:54-57.

⁹⁸³ Riddle, 3:36-39.

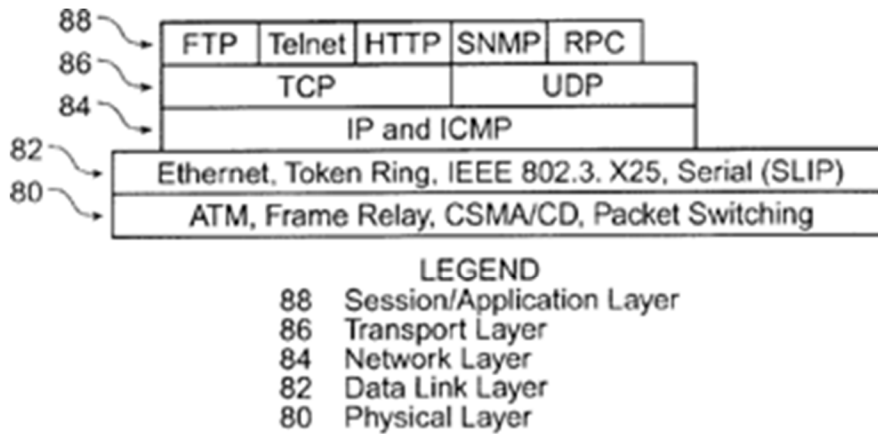


FIG. 1D
(PRIOR ART)

797. Riddle describes these prior art layers in detail:

FIG. 1D is illustrative of the constituents of the Transmission Control Protocol/Internet Protocol (TCP/IP) protocol suite. *The base layer of the TCP/IP protocol suite is the physical layer 80*

Overlying the physical layer is the data link layer 82. The data link layer provides the function and protocols to transfer data between network resources and to detect errors that may occur at the physical layer. ...

Network layer protocols 84 overlay the datalink layer and provide the means for establishing connections between networks. ...

The transport layer protocols 86 provide end-to-end transport services across multiple heterogenous networks....

The session, or application layer 88 provides a list of network applications and utilities, a few of which are illustrated here. For example, File Transfer Protocol (FTP) is a standard TCP/IP protocol for transferring files from one machine to another.... The Hypertext Transfer Protocol is a simple protocol built on top of Transmission Control Protocol (TCP). It is the

mechanism which underlies the function of the World Wide Web.⁹⁸⁴

798. To classify packets, Riddle teaches assigning service levels to traffic classes.⁹⁸⁵ This includes “applying individual instances of traffic classification paradigms to packet network flows based on *selectable information obtained from a plurality of layers of a multi-layered communication protocol* in order to define a characteristic class, then mapping the flow to the defined traffic class.”⁹⁸⁶

799. Riddle details that its packet classification includes identifying the protocol being used in the packet at a plurality of protocol layer levels:

Traffic classes may be defined at any level of the IP protocol as well as for other non-IP protocols.... At the application level, traffic classes may be defined for specific URIs within a web server. Traffic classes may be defined having “Web aware” class attributes. For example, a traffic class could be created such as all URIs matching “*.html” for all servers, or all URI patterns matching “*.gif” for server X, or for access to server Y with URI pattern “/sales/*” from client Z, wherein ‘*’ is a wildcard character, i.e., a character which matches all other character combinations....

The present invention provides a method for classifying traffic according to *a definable set of classification attributes selectable by the manager*, including selecting a subset of traffic of interest to be classified. The invention provides the ability to classify and search traffic based upon multiple orthogonal classification attributes.

⁹⁸⁴ Riddle, 7:35-8:46.

⁹⁸⁵ Riddle, Abstract 4:7-10, 4:60-66.

⁹⁸⁶ Riddle, 4:10-15.

Traffic class membership may be hierarchical. Thus, a flow may be classified by a series of steps through a traffic class tree, with the last step (i.e., at the leaves on the classification tree) mapping the flow to a policy. The policy is a rule of assignment for flows. *Web traffic may also be classified by HTTP header types such as Content-Type (MIME type) or User-Agent.*⁹⁸⁷

800. Further, Riddle describes classifying traffic based on packet characteristics beyond protocol-layer categories:

Network traffic is automatically classified under existing classes, beginning with the broadest classes, an inbound traffic class and an outbound traffic class, *in protocol layer independent categories. For example, a particular instance of traffic may be classified according to its transport layer characteristics, e.g., Internet Protocol port number, as well as its application layer information, e.g., SMTP.* Characteristics such as MIME types may also be automatically identified. Standard protocols, such as, IPX, SNA, and services, such as, SMTP and FTP are recognized for automatic classification. *Classification is performed to the most specific level determinable. For example, in select embodiments, non-IP traffic, such as SNA, may be classified only by protocol, whereas Internet Protocol traffic may be classified to the /etc/services level.* Classification beyond a terminal classification level is detected and prevented. For example, in a select embodiment, a class matching “ipx” or “nntp” will not be further automatically classified.⁹⁸⁸

⁹⁸⁷ Riddle, 8:57-9:27.

⁹⁸⁸ Riddle, 10:57-11:9.

801. As shown below, Riddle’s Table 2 includes exemplary information from which traffic classes may be built.⁹⁸⁹ For example, a traffic class may be a service aggregate defined for an FTP application (using application-layer protocol) with TCP connections (using transport-layer protocol).

Components of a Traffic Class Specifier

Inside (Client or Server)	Global	Outside (Server or Client)
IP Address/Domain Name	TCP or UDP Service	IP Address/Domain Name
Port Number	e.g., WWW, FTP, RealAudio, etc.	Port Number
MAC Address	URI pattern for Web Service, MIME type for Web Service IPX Service SNA Service LAT Service IP precedence	MAC Address

802. Moreover, Riddle describes displaying classified traffic to a network manager.⁹⁹⁰ Riddle includes an example display showing a hierarchical arrangement of FTP, TCP, and HTTP protocols used in the packet that were identified from a plurality of protocols at a plurality of protocol layer levels:

In an optional step 413 (not show), after the processing of flowchart 401 completes or at periodic intervals or on demand, a list of traffic classes produced in steps 402 through 412 are displayed to a network manager. ... The display can be hierarchical, as depicted in lines (3) below:

FTP (3)

⁹⁸⁹ Riddle, 9:64-65.

⁹⁹⁰ Riddle, 12:64-13:23.

```
FTP-cmd
FTP-data
to host1
tcp
  FTP
    FTP-cmd
    FTP-data
  HTTP
    images
    java
    text
port 9999 ....991
```

- i. '751 Claim Element 1.8: "such that the flow-entry database is to store flow entries for a plurality of conversational flows using a plurality of protocols, at a plurality of layer levels, including levels above the network layer"*

803. Riddle alone and/or in view of Ferdinand renders obvious this claim element. As shown above regarding '751 claim elements 1.2-1.3, Riddle discloses storing flow-entries for a plurality of conversational flows using protocol layer levels above the network layer. And as discussed above regarding '751 claim element 1.2, Riddle alone and/or in view of Ferdinand renders obvious storing flow-entries

⁹⁹¹ Riddle, 12:64-13:23. As shown in Ex. 1024 ('864 Provisional) (see page 24), the first line "FTP" of Riddle's exemplary classification tree should be directly above the subclassifications "FTP-cmd" and "FTP-data."

in a database. I incorporate by reference my discussion for '751 claims elements 1.2 and 1.3 as if fully set forth herein.

804. For example, as discussed regarding '751 claim elements 1.2-1.3, Riddle discloses storing service aggregates (claimed “conversational flows”) for a plurality of TCP or UDP connections (transport layer) to determine an FTP application (application layer). As another example, Riddle teaches identifying conversational flows for PointCast connections by relating multiple HTTP connections (application layer) with URLs that begin with “/FIDO-1/.”⁹⁹² Both of these examples meet the claimed “conversational flows using a plurality of protocols, at a plurality of layer levels, including levels above the network layer.”

805. For the above reasons, it is my opinion that Riddle in view of Ferdinand renders obvious claim 1 of the '751 Patent.

3. Dependent '751 Claim 2

806. Riddle discloses all the limitations of claim 2, which depends from independent claim 1.

a. '751 Claim Element 2.1: “wherein step (b) includes extracting identifying portions from the packet”

807. Riddle discloses this claim element. As shown in Figure 4A, Riddle’s steps 402 to 408 disclose parsing and extracting information from a packet:

- Step 402 - “Parse flow specification from a packet of the flow”;

⁹⁹² Riddle, 11:47-67.

- Step 406 - “Traffic matches a class?”; and
- Step 408 - “Enter into saved list characteristics of the traffic.”⁹⁹³

If the parsed packet matches a traffic class, Riddle teaches that an entry is made into a saved list with the extracted identifying information such as “protocol type (SAP), IP protocol number, server port, traffic type if known, MIME type, a time of occurrence of the traffic.”⁹⁹⁴

808. Further, Riddle’s claims 1, 8, and 11 provide examples of identification information parsed and extracted from a packet:

[Parse/parsing] a packet into a first flow specification, wherein said first flow specification contains at least one instance of any one of the following: a protocol family designation,
a direction of packet flow designation,
a protocol type designation,
a pair of hosts,
a pair of ports,
in HTTP protocol packets, a pointer to a MIME type.

As discussed above, a POSITA would have understood that Riddle’s “pair of hosts” refers to the network-layer source and destination addresses, and Riddle’s “pair of ports” refers to the transport-layer source and destination port numbers.

⁹⁹³ Riddle, 12:42-53.

⁹⁹⁴ Riddle, Fig. 4A, 12:50-53.

- b. '751 Claim Element 2.2: “wherein the extracting at any layer level is a function of the protocol being used at the layer level, and”

809. Riddle discloses this claim element. Riddle teaches managing “network bandwidth based on information ascertainable from multiple layers of OSI network model.”⁹⁹⁵ And Riddle’s Figure 1D illustrates the well-known prior art relationship between the layers of the TCP/IP protocol suite, including the application, transport, network, data-link, and physical layers.⁹⁹⁶

810. To classify extracted packet portions, Riddle teaches assigning service levels to traffic classes.⁹⁹⁷ This includes “applying individual instances of traffic classification paradigms to packet network flows based on *selectable information obtained from a plurality of layers of a multi-layered communication protocol in order to define a characteristic class*, then mapping the flow to the defined traffic class.”⁹⁹⁸ Riddle specifies the “automatic classification is sufficiently robust to classify a complete enumeration of the possible traffic.”⁹⁹⁹

811. Riddle details that its packet classification may be defined at any level of the OSI model:

Traffic classes may be defined at any level of the IP protocol as well as

⁹⁹⁵ Riddle, 1:54-57.

⁹⁹⁶ Riddle, 7:35-8:46, Fig. 1D.

⁹⁹⁷ Riddle, Abstract 4:7-10, 4:60-66.

⁹⁹⁸ Riddle, 4:10-15.

⁹⁹⁹ Riddle, Abstract, 4:15-17.

for other non-IP protocols.... At the application level, traffic classes may be defined for specific URIs within a web server. Traffic classes may be defined having “Web aware” class attributes. For example, a traffic class could be created such as all URIs matching “*.html” for all servers, or all URI patterns matching “*.gif” for server X, or for access to server Y with URI pattern “/sales/*” from client Z, wherein ‘*’ is a wildcard character, i.e., a character which matches all other character combinations....

The present invention provides a method for classifying traffic according to *a definable set of classification attributes selectable by the manager*, including selecting a subset of traffic of interest to be classified. The invention provides the ability to classify and search traffic based upon multiple orthogonal classification attributes.

Traffic class membership may be hierarchical. Thus, a flow may be classified by a series of steps through a traffic class tree, with the last step (i.e., at the leaves on the classification tree) mapping the flow to a policy. The policy is a rule of assignment for flows. *Web traffic may also be classified by HTTP header types such as Content-Type (MIME type) or User-Agent.*¹⁰⁰⁰

812. As shown below, Riddle’s Table 2 includes exemplary information extracted from a packet from which traffic classes may be built.¹⁰⁰¹ For example, a traffic class may be a service aggregate defined for an FTP application using a client IP address and a server IP address.

¹⁰⁰⁰ Riddle, 8:57-9:27.

¹⁰⁰¹ Riddle, 9:64-65.

Components of a Traffic Class Specifier

Inside (Client or Server)	Global	Outside (Server or Client)
IP Address/Domain Name	TCP or UDP Service	IP Address/Domain Name
Port Number	e.g., WWW,	Port Number
MAC Address	FTP, RealAudio, etc. URI pattern for Web Service, MIME type for Web Service IPX Service SNA Service LAT Service IP precedence	MAC Address

813. Based on these teachings, a POSITA would have understood that Riddle’s packet information extraction is a function of the layer protocols being used in traffic classes created by the network manager.

c. ’751 Claim Element 2.3: “wherein the looking up uses a function of the identifying portions”

814. Riddle discloses this claim element, which relates to the looking up step of ’751 claim element 1.2 and the extracting of the identifying portions of ’751 claim element 2.1.

815. As illustrated in Figure 4A, Riddle discloses using the packet’s identifying portions to look up flow-entries:

FIG. 4A depicts a flowchart 401 of processing steps for automatically classifying traffic. In a step 402, a flow specification is parsed from the flow being classified. *Then in a step 404, the flow specification parsed from the flow in step 402 is compared with the traffic specifications in each*

node of the classification tree. Rules are checked starting from most specific to least specific. *In a decisional step 406, a determination is made if traffic matches one of the classes being classified. If this is so, then in a step 408, an entry is made in a list of identifying characteristics, such as protocol type (SAP), IP protocol number, server port, traffic type if known, MIME type, a time of occurrence of the traffic.* In an optional step 410, duplicate instances having the same identifying characteristics are suppressed, in favor of keeping a count of the duplicates and a most recent time traffic with these identifying characteristics was encountered. In an optional step 412, a byte count of traffic of this type has been detected is included.¹⁰⁰²

816. Based on these teachings, a POSITA would have understood that Riddle's monitor uses the identifying characteristics (such as protocol type, IP protocol number, server port, traffic type, MIME type, a time of occurrence of the traffic) to determine whether the parsed packet matches a traffic class (e.g., service aggregate) and/or is a duplicate. Thus, Riddle teaches its flow-entry lookup uses a function of the parsed packet's identifying portions.

4. Dependent '751 Claim 5

817. Riddle discloses all limitations of claim 5, which recites: "A method according to claim 1, further including reporting one or more metrics related to the flow of a flow-entry from one or more of the statistical measures in the flow-entry."

¹⁰⁰² Riddle, 12:42-59; see also step 402 to 412 of Riddle's Fig. 4.

818. After progressing through the steps of Figure 4A's flowchart 401, Riddle discloses reporting one or metrics by displaying the results to a network manager:

In an optional step 413 (not show), after the processing of flowchart 401 completes or at periodic intervals or on demand, a list of traffic classes produced in steps 402 through 412 are displayed to a network manager.

The list may be sorted by any well-known criteria such as: 1) most "hits" during a recent interval, 2) most recently-seen (most recent time first), 3) most data transferred (bytes/second) during some interval, or a moving average. The user may choose an interval length or display cutoff point (how many items, how recent, at least B bytes per second, or other thresholds).¹⁰⁰³

Thus, Riddle discloses displaying the received traffic sorted by metrics, such as most "hits" (i.e., the duplicate count), most recently seen, most data transferred, or moving average.

5. Dependent '751 Claim 10

819. Riddle discloses all the limitations of claim 10, which depends from independent claim 1.

¹⁰⁰³ Riddle, 12:64-13:8, 14:1-5.

- a. *'751 Claim Element 10.1: "wherein step (c) includes if the packet is of an existing flow, identifying the last encountered state of the flow and performing any state operations specified for the state of the flow starting from the last encountered state of the flow; and"*

820. Riddle discloses this claim element. As explained in my discussion of '751 claim element 1.4, which I incorporate by reference, Riddle teaches, if the packet is of an existing flow, identifying the last encountered state of the flow and performing any state operations specified for the state of the flow.¹⁰⁰⁴ The only additional requirement of '751 claim element 10.1 is that the state operations start from the last encountered state of the flow.

821. As shown regarding '751 claim element 1.4, Riddle discloses performing the state operations for the state of the flow by updating flow entries, including determining whether a flow belongs to a stored service aggregate flow (i.e., conversational flow), and determining statistical metrics for (i) a count of the duplicates, (ii) the most recent time traffic with the same identifying characteristics was encountered, and (iii) the byte count of the detected traffic.¹⁰⁰⁵

822. A POSITA would have understood that each of these state operations start from the last encountered state. For example, Riddle's service aggregate flow

¹⁰⁰⁴ '751 claim element 1.4 recites "(c) if the packet is of an existing flow, identifying the last encountered state of the flow, performing any state operations specified for the state of the flow"

¹⁰⁰⁵ Riddle, 12:42-59, 13:1-8, 13:36-14:5, Figs. 4A-4B.

check starts from previous encounter.¹⁰⁰⁶ Similarly, Riddle's count of duplicates starts from the previous count.¹⁰⁰⁷ And Riddle's most recent time traffic was encountered starts with the previous time.¹⁰⁰⁸ Further, Riddle's byte count starts from the previous byte count.¹⁰⁰⁹

823. As I discuss above with respect to '751 claim element 1.4, Riddle describes displaying a hierarchical classification tree that shows state operations related to FTP. For example, Riddle presents classes "to host 1," "tcp," and "FTP."¹⁰¹⁰ A POSITA would have understood that transitioning from one class to the next in this hierarchy involves performing state operations specified for the state of the flow starting from the last encountered state of the flow. Examples of these state operations include saving TCP-session information from classified TCP packets exchanged with the host, and aggregating saved TCP-sessions belonging to the instance of FTP application being classified.

824. As I also discussed with respect to '751 claim element 1.4, in connection with Figure 4A, Riddle describes analyzing information identifying the characteristics of the traffic as the classifier parses packets of a flow and matches the parsed

¹⁰⁰⁶ Riddle, 13:36-62, Fig. 4B.

¹⁰⁰⁷ Riddle, 12:42-59, Fig. 4A.

¹⁰⁰⁸ Riddle, 12:42-59, 13:1-8, 14:1-5, Fig. 4A.

¹⁰⁰⁹ Riddle, 12:42-59, 13:1-8, 14:1-5, Fig. 4A.

¹⁰¹⁰ Riddle, 13:11-22.

packets to a class.¹⁰¹¹ Riddle describes that after an initial classification, “sub-classification” proceeds in a sequential manner by performing finer grade matchings as characteristics such as the hosts and services are identified, leading to matching of a flow with operation of a particular application program.¹⁰¹² In accordance with the classification tree, Riddle’s classifier advances through the sequence of packets in a particular traffic flow to parse and classify those packets. This results in Riddle’s classifier performing a corresponding state operation at each node to update the identifying characteristics of the flow.

b. ’751 Claim Element 10.2: “wherein step (d) includes if the packet is of a new flow, performing any state operations required for the initial state of the new flow”

825. Riddle discloses this claim element, which is similar to ’751 claim element 1.5 and disclosed in the prior art for the same reasons.¹⁰¹³ I incorporate by reference my discussions of ’751 claim element 1.5 as if fully set forth herein.

6. Dependent ’751 Claim 14

826. Riddle discloses all limitations of claim 14, which recites: “A method according to claim 10, wherein the state operations include updating the flow-entry, including storing identifying information for future packets to be identified with

¹⁰¹¹ Riddle, 12:42-48.

¹⁰¹² Riddle, 11:25-31, 13:11-22.

¹⁰¹³ The only difference is that ’751 claim element 1.5 also recites “storing a new flow-entry for the new flow in the flow-entry database, including storing one or more statistical measures kept in the flow-entry.”

the flow-entry.” The claim recites elements similar to ’646 claim elements 7.9 and 16.5, and is disclosed in the prior art for the same reasons. I incorporate by reference my discussions of those elements as if fully set forth herein.

827. As explained in my discussion of ’751 claim 10, which I incorporate by reference, Riddle teaches updating flow entries with (a) whether flow belongs to previously-encountered service aggregate flow, (b) the count of the duplicates, (c) the most recent time traffic with the same identifying characteristics was encountered, and (d) the byte count of the detected traffic.¹⁰¹⁴

828. Further, Riddle’s claims 1-3, 8, and 11 detail creating new tree nodes if a packet is of a new flow so future packets can be identified with the new flows. For example, Riddle’s claim 1 recites:

A method for automatically classifying traffic in a packet communications network, said network having any number of flows, including zero, comprising the steps of: parsing a packet into a first flow specification ... thereupon, matching the first flow specification of the parsing step to a plurality of classes represented by a plurality nodes of a classification tree type, each said classification tree type node having a traffic specification; thereupon, *if a matching classification tree type node was not found in the matching step, associating said first flow specification with one or more newly-created classification tree type nodes; thereupon, incorporating said newly-created classification tree type*

¹⁰¹⁴ Riddle, 12:53-58, 13:36-62, Figs. 4A-4B.

nodes into said plurality of classification tree type nodes.

829. As shown in Figure 4A's flowchart, Riddle teaches that flow-entries include information to identify future packets with stored flow-entries:

In a decisional step 406, a determination is made if traffic matches one of the classes being classified. *If this is so, then in a step 408, an entry is made in a list of identifying characteristics, such as protocol type (SAP), IP protocol number, server port, traffic type if known, MIME type, a time of occurrence of the traffic.* In an optional step 410, duplicate instances having the same identifying characteristics are suppressed, in favor of keeping a count of the duplicates and a most recent time traffic with these identifying characteristics was encountered. In an optional step 412, a byte count of traffic of this type has been detected is included.¹⁰¹⁵

830. Regarding identifying service aggregate flows in Figure 4B, Riddle describes storing a new flow-entry for a new flow. Riddle's new flow-entry includes information to identify future packet with the new flow entry.¹⁰¹⁶

831. As such, a POSITA would have understood that Riddle's identifying information (such as protocol type, IP protocol number, server port, traffic type, MIME type, a time of occurrence of the traffic) is identifying information for future packets to be identified with the flow-entry.

¹⁰¹⁵ Riddle, 12:37-59; see also steps 402 to 412 of Riddle's Fig. 4A.

¹⁰¹⁶ Riddle, 13:53-61, 15:16-27, Fig. 4B.

7. Dependent '751 Claim 15

832. Riddle discloses all the limitations of claim 15, which recites: “A method according to claim 14, further including receiving further packets, wherein the state processing of each received packet of a flow furthers the identifying of the application program of the flow.” The claim recites elements similar to '099 claim elements 1.7 and 1.10, and disclosed in the prior art for the same reasons. I incorporate by reference my discussions of those elements as if fully set forth herein.

833. For example, Riddle discusses classifying packets based upon at least two protocols, a transport-layer protocol and an application-layer protocol:

Network traffic is automatically classified under existing classes, beginning with the broadest classes, an inbound traffic class and an outbound traffic class, in protocol layer independent categories. *For example, a particular instance of traffic may be classified according to its transport layer characteristics, e.g., Internet Protocol port number, as well as its application layer information, e.g., SMTP.*¹⁰¹⁷

Accordingly, as the protocol layers of each packet are examined, a POSITA would have understood that Riddle's monitor identifies the transport session, IP session, and then the particular instance of traffic, such as an SMTP e-mail program. Riddle teaches processing each packet and using the application-layer protocol to identify the flow's application program, such as an SMTP e-mail program.

¹⁰¹⁷ Riddle, 10:59-65.

834. For the transport-layer protocol, Riddle describes an example of classifying using the TCP protocol.¹⁰¹⁸ Riddle discloses that “Internet/Intranet technology is based largely on the TCP/IP protocol suite, where IP, or Internet Protocol, is the network layer protocol and TCP, or Transmission Control Protocol, is the transport layer protocol.”¹⁰¹⁹

835. At the time of the Challenged Patents’ priority date, a POSITA would have known that TCP connections are initialized by a three-way handshake. Thus, a POSITA would have understood that Riddle’s monitor examines each received packet of the handshake to identify the TCP connection. For example, the 1981 RFC793 describes initializing TCP connections as follows:

For a connection to be established or initialized, the two TCPs must synchronize on each other’s initial sequence numbers. This is done in an exchange of connection establishing segments carrying a control bit called “SYN” (for synchronize) and the initial sequence numbers. As a shorthand, segments carrying the SYN bit are also called “SYNs”. Hence, the solution requires a suitable mechanism for picking an initial sequence number and a slightly involved handshake to exchange the ISN’s.

The synchronization requires each side to send it’s own initial sequence number and to receive a confirmation of it in acknowledgment from the other side. Each side must also receive the other side’s initial sequence

¹⁰¹⁸ Riddle, 2:2-29, 10:1-18 (Table 2), Figs. 1B, 1C (TCP at transport layer 86).

¹⁰¹⁹ Riddle, 2:14-17.

number and send a confirming acknowledgment.

- 1) A --> B SYN my sequence number is X
- 2) A <-- B ACK your sequence number is X
- 3) A <-- B SYN my sequence number is Y
- 4) A --> B ACK your sequence number is Y

Because steps 2 and 3 can be combined in a single message this is called the three way (or three message) handshake.

A three way handshake is necessary because sequence numbers are not tied to a global clock in the network, and TCPs may have different mechanisms for picking the ISN's.¹⁰²⁰

836. As the above RFC793 passages illustrate, a POSITA would have understood that Riddle's monitor receives multiple packets and examines each packet in order to identify a TCP protocol connection using the three-way handshake. Moreover, a POSITA would have understood that this three-way handshake is completed before any higher-layer protocol information (such as application-layer information) can be sent or received.

837. Riddle further describes hierarchical classifications such as "to host 1," "tcp," and "FTP."¹⁰²¹ As each packet is received, Riddle's classification proceeds along this hierarchy, leading to matching of a flow with the operation of a particular application program.

¹⁰²⁰ Ex. 1039 (RFC793), 31-32.

¹⁰²¹ Riddle, 13:11-22.

838. As another example, Riddle presents classifying PointCast traffic, which involves a sequence of packets traversing a transition pattern (e.g., HTTP connection, Pointcast get request, inbound traffic) indicating an association with the PointCast application program initiated on the user's computer.

839. Because Riddle discloses identifying packets based on TCP protocol information and application-layer information, Riddle teaches receiving further packets, wherein the state processing of each received packet of a flow furthers the identifying of the application program of the flow.

8. Independent '751 Claim 17

840. It is my opinion that independent claim 17 of the '751 Patent is obvious in light of Riddle in view of Ferdinand.

- a) *'751 Claim 17's Preamble: "A packet monitor for examining packets passing through a connection point on a computer network, each packets [sic] conforming to one or more protocols, the monitor comprising"*

841. Riddle discloses all elements of this preamble. This preamble is substantially identical to the preamble of '646 claim 1 and disclosed by the prior art for the same

reasons.¹⁰²² I incorporate by reference my discussion of '646 claim 1's preamble as if fully set forth herein.¹⁰²³

b. *'751 Claim Element 17.1: "(a) a packet acquisition device coupled to the connection point and configured to receive packets passing through the connection point"*

842. Riddle discloses this claim element, which is identical to '646 claim element 1.1 and disclosed by the prior art for the same reasons. I incorporate by reference my discussion of '646 claim element 1.1 as if fully set forth herein.¹⁰²⁴

c. *'751 Claim Element 17.2: "(b) a memory for storing a database for containing one or more flow-entries for previously encountered conversational flows to which a received packet may belong"*

843. Riddle alone and/or in view of Ferdinand renders obvious this claim element. This element is similar to '099 claim element 1.5 and '646 claim element 1.2, and is disclosed by the prior art for the same reasons. I incorporate by reference my discussions of those elements as if fully set forth herein.¹⁰²⁵

¹⁰²² The only difference is the correction of typographical errors in '646 claim 1. The '646 Patent, Aug. 3, 2004 Certificate of Correction.

¹⁰²³ '751 claim 17's preamble is also identical to the preambles of '646 claim 7 (except for a corrected typographical error) and '789 claim 19, my discussions of which likewise incorporate my discussion of '646 claim 1's preamble.

¹⁰²⁴ '751 claim element 17.1 is also identical to '646 claim element 7.1 and '789 claim element 19.1, my discussions of which likewise incorporate my discussion of '646 claim element 1.1.

¹⁰²⁵ As explained in those discussions, Riddle discloses identifying conversational flows in the form of identifying service aggregate flows and PointCast flows, and Riddle in view of Ferdinand discloses or renders obvious a memory for storing a flow-entry database.

- d. *'751 Claim Element 17.3: "a conversational flow including an exchange of a sequence of one or more packets in any direction between two network entities as a result of a particular activity using a particular layered set of one or more network protocols, a conversational flow further having a set of one or more states, including an initial state; and"*

844. Riddle discloses this claim element, which is identical to '751 claim element 1.3 and disclosed by the prior art for the same reasons. I incorporate by reference my discussion of '751 claim element 1.3 as if fully set forth herein.

- e. *'751 Claim Element 17.4: "(c) an analyzer subsystem coupled to the packet acquisition device configured to lookup for each received packet whether a received packet belongs to a flow-entry in the flow-entry database"*

845. Riddle discloses this claim element, which is similar to '099 claim element 1.6 and '646 claim element 1.4 and disclosed by the prior art for the same reasons. I incorporate by reference my discussion of '099 claim element 1.6 and '646 claim element 1.4 as if fully set forth herein.¹⁰²⁶

¹⁰²⁶ As explained in those discussions, Riddle packet acquisition device is coupled to the analyzing engine's processor discloses an engine configured to lookup whether a received packet belongs to a flow-entry in the flow-entry database. Riddle, 12:37-49, 13:36-62, claim 8, Figs. 3, 4A-4B. Riddle details this engine includes a processor and corresponding code to perform the function of looking up whether a received packet belongs to stored flow-entry. Riddle, 5:53-57, 12:37-49, 13:36-62, claim 8, Figs. 3, 4A-4B. And a POSITA would have appreciated that Riddle discloses an "analyzer subsystem" because the processor and corresponding code is a system in Riddle's monitor that analyzes parsed packet information. Additionally, as discussed above regarding '646 claim element 1.4, Riddle discloses that its packet acquisition device is coupled to the analyzing engine's processor.

- f. '751 Claim Element 17.5: an analyzer subsystem configured "to update the flow-entry of the existing flow including storing one or more statistical measures kept in the flow-entry in the case that the packet is of an existing flow, and"*

846. Riddle discloses this claim element, which is similar to '751 claim element 1.4 and disclosed by the prior art for the same reasons. I incorporate by reference my discussion of '751 claim element 1.4 as if fully set forth herein.

- g. '751 Claim Element 17.6: an analyzer subsystem configured "to store a new flow-entry for the new flow in the flow-entry database, including storing one or more statistical measures kept in the flow-entry if the packet is of a new flow"*

847. Riddle alone and/or in view of Ferdinand renders obvious this claim element, which is similar to '751 claim element 1.5 and disclosed by the prior art for the same reasons. I incorporate by reference my discussion of '751 claim element 1.5 as if fully set forth herein.

- h. '751 Claim Element 17.7: "wherein the analyzer subsystem is further configured to identify the protocol being used in the packet from a plurality of protocols at a plurality of protocol layer levels, and"*

848. Riddle discloses this claim element, which is similar to '751 claim element 1.7 and disclosed by the prior art for the same reasons. I incorporate by reference my discussion of '751 claim element 1.7 as if fully set forth herein.

- i. '751 Claim Element 17.8: “wherein the database is to store flow entries for a plurality of conversational flows using a plurality of protocols, at a plurality of layer levels, including levels above the network layer”*

849. Riddle alone and/or in view of Ferdinand renders obvious this claim element, which is substantially similar to '751 claim element 1.8 and disclosed by the prior art for the same reasons. I incorporate by reference my discussion of '751 claim element 1.8 as if fully set forth herein.

850. Thus, for all the above reasons, it is my opinion that Riddle in view of Ferdinand renders obvious claims 1, 2, 5, 10, 14, 15, and 17 of the '751 Patent.

B. For the '751 Patent, Riddle in View of Ferdinand and Further in View of Yu Renders Obvious Claims 1, 2, 5, 10, 14, 15, and 17.

851. It further is my opinion that a POSITA would have recognized that each and every limitation of '751 claims 1, 2, 5, 10, 14, 15, and 17 is disclosed or rendered obvious by Riddle in view of Ferdinand and further in view of Yu. Specifically, my opinions regarding Riddle in view of Ferdinand are exactly the same as those set forth above in Section X.A, but further include the teachings of Yu. Thus, as if fully set forth here, I incorporate the discussion from Section X.A regarding the obviousness of '751 claims 1, 2, 5, 10, 14, 15, and 17 over Riddle in view of Ferdinand.

852. As noted, all the Challenged Claims require “conversational flows.” For ex-

ample, '751 claim element 1.2 recites “looking up a flow-entry database for containing one or more flow-entries for previously encountered conversational flows.” While Riddle itself teaches identifying conversational flows, Yu further demonstrates identifying conversational flows through its “flow classification.”¹⁰²⁷ Further, Yu teaches flow classification logic that “keeps track of the flow’s state until matching criteria is met” when identifying whether a packet belongs to a conversational flow.¹⁰²⁸

853. As discussed with respect to the obviousness of '099 claims 1 and 2 over Riddle in view of Ferdinand and further in view of Yu in Section VII.C, Yu teaches identifying the claimed “conversational flow” by Yu’s flow classifier linking multiple “streams” into a “flow” based on application or application data.¹⁰²⁹

And as discussed in Section VII.C, Yu teaches state tracking that binds policy decisions to each stream of a flow so that actions can be taken on future packets without intervention from the “host” application.¹⁰³⁰ Moreover, as discussed in Section VII.C, Yu specifies using hash values to find corresponding policies to reduce further complicated pattern-matching.¹⁰³¹ I incorporate by reference that discussion as if fully set forth herein.

¹⁰²⁷ Yu, 1:56-60, 3:32-36, 3:47-49; 4:1-8.

¹⁰²⁸ Yu, 4:62-64.

¹⁰²⁹ Yu, 1:56-60, 3:32-49; 4:1-8.

¹⁰³⁰ Yu, 4:57-5:13.

¹⁰³¹ Yu, 4:23-29.

854. As discussed with respect the obviousness of '099 claims 1 and 2 over Riddle in view of Ferdinand and further in view of Yu in Section VII.C, I explain how a POSITA would have been motivated to combine Yu's teachings into Riddle's monitor. I incorporate by reference that discussion as if fully set forth herein.

855. For the same reasons, it is my opinion that combining the teachings of Riddle, Ferdinand, and Yu renders obvious all the claim elements relating to "conversational flows" as well as performing state operations relating to the flow's state.

856. As set forth in my analysis of the '751 Patent in Sections X.A.2 through X.A.8 above, Riddle and Ferdinand disclose or render obvious all the remaining elements of '751 claims 1, 2, 5, 10, 14, 15, and 17. Thus, it is my opinion that Riddle in view of Ferdinand and further in view of Yu renders obvious '751 claims 1, 2, 5, 10, 14, 15, and 17.

C. For the '751 Patent, Riddle in View of Ferdinand and Further in View of RFC1945 Renders Obvious Claims 1, 2, 5, 10, 14, 15, and 17.

857. It further is my opinion that a POSITA would have recognized that each and every limitation of '751 claims 1, 2, 5, 10, 14, 15, and 17 is disclosed or rendered obvious by Riddle in view of Ferdinand and further in view of RFC1945. Specifically, my opinions regarding Riddle in view of Ferdinand are exactly the same as those set forth above in Section X.A, but further include the teachings of RFC1945. Thus, as if fully set forth here, I incorporate the discussion from Section

X.A regarding the obviousness of '751 claims 1, 2, 5, 10, 14, 15, and 17 over Riddle in view of Ferdinand.

858. As noted, all the Challenged Claims require “conversational flows.” For example, '751 claim element 1.2 recites “looking up a flow-entry database for containing one or more flow-entries for previously encountered conversational flows.” While Riddle itself teaches identifying conversational flows, RFC1945 further demonstrates identifying conversational flows through the additional example of the use of HTTP header fields.

859. As discussed with respect to the obviousness of '099 claims 1 and 2 over Riddle in view of Ferdinand and further in view of RFC1945 in Section VII.E, I understand that Patentee and its technical expert have taken the position that the HTTP Referrer field can be used to create a conversational flow. I incorporate by reference that discussion as if fully set forth herein.

860. As discussed with respect the obviousness of '099 claims 1 and 2 over Riddle in view of Ferdinand and further in view of RFC1945 in Section VII.E, I explain how a POSITA would have been motivated to combine RFC1945's teachings into Riddle's monitor. I incorporate by reference that discussion as if fully set forth herein.

861. For the same reasons, it is my opinion that combining the teachings of Riddle, Ferdinand, and RFC1945 renders obvious all the claim elements relating to

“conversational flows,” at least under Patentee’s interpretation for that term.

862. As set forth in my analysis of the ’751 Patent in Sections X.A.2 through X.A.8 above, Riddle and Ferdinand disclose or render obvious all the remaining elements of ’751 claims 1, 2, 5, 10, 14, 15, and 17. Thus, it is my opinion that Riddle in view of Ferdinand and further in view of RFC1945 renders obvious ’751 claims 1, 2, 5, 10, 14, 15, and 17 at least under Patentee’s interpretation of “conversational flow.”

XI. THE CLAIMS OF THE ’789 PATENT ARE UNPATENTABLE

863. For the ’789 Patent, the challenged claims include independent claims 1, 19, and 44 as well as dependent claims 2, 13-17, 20, 31, 33, 34, 42, 48, and 49. As I detail below, it is my opinion that a POSITA would have recognized that each and every limitation of those claims is disclosed or rendered obvious by the prior art. Specifically, it is my opinion that Riddle in view of Ferdinand renders obvious ’789 claims 1, 2, 13-17, 19, 20, and 42; Riddle in view of Ferdinand and further in view of Baker renders obvious ’789 claim 31; Riddle in view of Ferdinand and further in view of Wakeman renders obvious ’789 claims 33 and 34; and Riddle in view of Ferdinand and further in view of Hasani renders obvious ’789 claims 44, 48, and 49.

A. For the ’789 Patent, Riddle in View of Ferdinand Renders Obvious Claims 1, 2, 13-17, 19, 20, and 42

864. It is my opinion that a POSITA would have recognized that each and every

limitation of '789 claims 1, 2, 13-17, 19, 20, and 42 is disclosed or rendered obvious by the prior art. Specifically, it is my opinion that '789 claims 1, 2, 13-17, 19, 20, and 42 are obvious over Riddle in view of Ferdinand.

1. Reasons to Modify Riddle in View of Ferdinand

865. As described above with respect to the '099 Patent in Section VII.A.1, a POSITA would have been motivated and found it obvious to combine the teachings of Riddle and Ferdinand.

2. Independent '789 Claim 1

- a. '789 Claim 1's Preamble: "A method of examining packets passing through a connection point on a computer network, each packets [sic] conforming to one or more protocols, the method comprising"*

866. Riddle discloses all elements of this preamble. This preamble is similar to the apparatus-type claim preambles of '646 claim 1 and '099 claim 1, identical to the method-type preamble of '646 claim 16, and is disclosed in the prior art for the same reasons. I incorporate by reference my discussions of those preambles as if fully set forth herein.¹⁰³²

¹⁰³² My discussion of '646 claim 16's preamble, which is identical to '789 claim 1's preamble, likewise incorporates my discussions of the preambles of '646 claim 1 and '099 claim 1.

b. *'789 Claim Element 1.1: "(a) receiving a packet from a packet acquisition device"*

867. This claim element is identical to '646 claim element 16.1, similar to '646 claim element 1.1, and disclosed in the prior art for the same reasons. I incorporate by reference my discussion of '646 claim elements 1.1 and 16.1 as if fully set forth herein.

c. *'789 Claim Element 1.2: "(b) performing one or more parsing/extraction operations on the packet to create a parser record comprising a function of selected portions of the packet"*

868. This claim element is similar to '646 claim element 7.3, and disclosed or rendered obvious by the prior art for the same reasons.¹⁰³³ I incorporate by reference my discussion of '646 claim element 7.3 as if fully set forth herein.¹⁰³⁴

d. *'789 Claim Element 1.3: "(c) looking up a flow-entry database comprising none or more flow-entries for previously encountered conversational flows, the looking up using at least some of the selected packet portions and determining if the packet is of an existing flow"*

869. This claim element is identical to '646 claim element 16.3 (which additionally recites "the lookup being via a cache"), is similar to '646 claim element 1.2,

¹⁰³³ '646 claim element 7.3 recites "a parser subsystem ... configured to extract selected portions of the accepted packet and to output a parser record containing the selected portions."

¹⁰³⁴ My discussion of '646 claim element 16.2, which is identical to '789 claim element 1.2, likewise incorporates my discussion of '646 claim element 7.3.

and disclosed or rendered obvious by the prior art for the same reasons. I incorporate by reference my discussions of '646 claim elements 1.2 and 16.3 as if fully set forth herein.

e. '789 Claim Element 1.4: “(d) if the packet is of an existing flow, classifying the packet as belonging to the found existing flow; and”

870. This claim element is identical to '646 claim element 16.4, similar to '646 claim element 7.8, and disclosed or rendered obvious by the prior art for the same reasons.¹⁰³⁵ I incorporate by reference my discussions of '646 claim elements 7.8 and 16.4 as if fully set forth herein.

f. '789 Claim Element 1.5: “(e) if the packet is of a new flow, storing a new flow-entry for the new flow in the flow-entry database, including identifying information for future packets to be identified with the new flow-entry,”

871. This claim element is similar to '646 claim element 7.9 and disclosed or rendered obvious by the prior art for the same reasons. I incorporate by reference my discussion of '646 claim element 7.9 as if fully set forth herein.¹⁰³⁶

¹⁰³⁵ '646 claim element 7.8 recites that the lookup engine is configured so that the monitor performs this classification.

¹⁰³⁶ My discussion of '646 claim element 16.5, which is identical to '789 claim element 1.5, likewise incorporates my discussion of '646 claim element 7.9.

- g. '789 Claim Element 1.6: “wherein the parsing/extraction operations depend on one or more of the protocols to which the packet conforms”

872. This claim element is similar to '646 claim element 7.10 and disclosed or rendered obvious by the prior art for the same reasons. I incorporate by reference my discussion of '646 claim element 7.10 as if fully set forth herein.¹⁰³⁷

873. For the above reasons, it is my opinion that Riddle in view of Ferdinand renders obvious independent claim 1 of the '789 Patent.

3. Dependent '789 Claim 2

874. Riddle discloses all the limitations of claim 2, which recites: “A method according to claim 1, wherein each packet passing through the connection point is examined in real time.” The claim recites elements similar to the preamble of '099 claim 1, and is disclosed in the prior art for the same reasons. I incorporate by reference my discussions of the preamble of '099 claim 1 as if fully set forth herein.

875. For example, Riddle discloses examining packets in real-time to manage network bandwidth based on flow classification. Riddle emphasizes the need to allocate bandwidth to flows based on information ascertainable from multiple OSI lay-

¹⁰³⁷ My discussion of '646 claim element 16.6, which is identical to '789 claim element 1.6, likewise incorporates my discussion of '646 claim element 7.10.

ers of each flow, which improves data transmission efficiency and avoids unwanted data transfer stoppage.¹⁰³⁸ To improve bandwidth management, Riddle describes “policies to assign available bandwidth from a single logical link to network flows.”¹⁰³⁹ And Riddle teaches examining and classifying traffic flows to allocate bandwidth consistent with those policies.¹⁰⁴⁰ From these disclosures, a POSITA would have understood that Riddle teaches traffic-examination and classification in real-time to assign available bandwidth.

876. Indeed, Riddle seeks to provide “a method for analyzing real traffic in a customer’s network and *automatically* producing a list of the ‘found traffic’”¹⁰⁴¹ Riddle specifies that its monitor can determine a packet’s traffic class, such as Real Time Protocol used for point-to-point telephony.¹⁰⁴² Based on these disclosures, a POSITA would have understood that Riddle teaches examining packets in real-time to optimally assign bandwidth based on flow rates.

877. In addition, Riddle describes classifying a “complete enumeration of the possible traffic,” providing a default class for “all traffic” that does not match a user specified class, creating a new class to match an instance of otherwise non-

¹⁰³⁸ Riddle, 1:54-61, 2:4-13.

¹⁰³⁹ Riddle, 2:66-67.

¹⁰⁴⁰ Riddle, Abstract, 4:6-23, 10:19-51.

¹⁰⁴¹ Riddle, 3:67-4:2, Abstract, 3:32-39, 4:6-9, 10:57-59.

¹⁰⁴² Riddle, 12:3-12.

matching traffic, keeping detailed metrics of traffic, and checking for duplicates, for which purposes a POSITA would have understood Riddle to be teaching to examine each packet passing through the connection point.¹⁰⁴³

878. As discussed above in Section IV.A.1, Riddle incorporates-by-reference the teachings of U.S. Patent 6,046,980 (“Packer”) as though fully set forth in Riddle. Like Riddle, Packer describes a traffic classification system and specifies that traffic flows are examined “continuously and automatically.”¹⁰⁴⁴ This teaching further shows that Riddle discloses examining packets in real-time.

879. Further, as discussed above, a POSITA would have understood that the claimed “connection point” is where the packet monitor is connected to the network.¹⁰⁴⁵ Accordingly, Riddle discloses wherein each packet passing through the connection point is examined in real time.

4. Dependent ’789 Claim 13

880. Riddle discloses all the limitations of claim 13, which depends from independent claim 1.

¹⁰⁴³ Riddle, 4:16-17, 10:52-56, 12:50-59, 13:1-31, 13:59-60, .

¹⁰⁴⁴ Ex. 1031 (Packer), 4:12-16.

¹⁰⁴⁵ ’789 Patent, Fig. 1 (showing connection points 121, 123, 125), 9:4-19.

- a. *'789 Claim Element 13.1: "wherein step (d) includes if the packet is of an existing flow, obtaining the last encountered state of the flow and performing any state operations specified for the state of the flow starting from the last encountered state of the flow; and"*

881. Riddle discloses this claim element, which is substantially identical to '751 claim element 10.1,¹⁰⁴⁶ similar to '751 claim element 1.4, and disclosed in the prior art for the same reasons. I incorporate by reference my discussions of '751 claim elements 1.4 and 10.1 as if fully set forth herein.

- b. *'789 Claim Element 13.2: "wherein step (e) includes if the packet is of a new flow, performing any state operations required for the initial state of the new flow"*

882. Riddle discloses this claim element, which is similar to '751 claim element 1.5 and disclosed in the prior art for the same reasons.¹⁰⁴⁷ I incorporate by reference my discussion of '751 claim element 1.5 as if fully set forth herein.¹⁰⁴⁸

5. Dependent '789 Claim 14

883. Riddle discloses all the limitations of claim 14, which recites: "A method according to claim 13, wherein the state processing of each received packet of a flow

¹⁰⁴⁶ '751 claim element 10.1 recites "identifying" instead of "obtaining."

¹⁰⁴⁷ As noted, '751 claim element 1.5 encompasses '751 claim element 10.2 and, in addition, recites "storing a new flow-entry for the new flow in the flow-entry database, including storing one or more statistical measures kept in the flow-entry."

¹⁰⁴⁸ My discussion of '751 claim element 10.2, which is identical to '789 claim element 13.2, likewise incorporates my discussion of '751 claim element 1.5.

further the identifying of the application program of the flow.” This claim is substantially identical to ’751 dependent claim 15 (which additionally recites “further including receiving further packets”) and is disclosed in the prior art for the same reasons. I incorporate by reference my discussion of ’751 dependent claim 15 as if fully set forth herein, including my discussion of ’099 claim elements 1.7 and 1.10, which I incorporate by reference into my discussion of ’751 dependent claim 15.

6. Dependent ’789 Claim 15

884. Riddle discloses all limitations of claim 15, which recites: “A method according to claim 13, wherein the state operations include updating the flow-[entry], including storing identifying information for future packets to be identified with the flow-entry.”¹⁰⁴⁹ This claim is substantially identical to ’751 dependent claim 14 and disclosed in the prior art for the same reasons.¹⁰⁵⁰ I incorporate by reference my discussion of ’751 dependent claim 14 as if fully set forth herein, including my discussion of ’646 claim elements 7.9 and 16.5, which I incorporate by reference into my discussion of ’751 dependent claim 14.

7. Dependent ’789 Claim 16

885. Riddle discloses all the limitations of claim 16, which recites: “A method according to claim 15, wherein the state processing of each received packet of a flow

¹⁰⁴⁹ It is my understanding that the ’789 Patent’s Certificate of Correction corrects a typographical error in this claim.

¹⁰⁵⁰ The only difference is that ’751 claim 14 depends from ’751 claim 10.

further the identifying of the application program of the flow.” This claim is substantially similar to ’751 dependent claim 15 (which additionally recites “further including receiving further packets”) and is disclosed in the prior art for the same reasons.¹⁰⁵¹ I incorporate by reference my discussion of ’751 dependent claim 15 as if fully set forth herein, including my discussion of ’099 claim elements 1.7 and 1.10, which I incorporate by reference into my discussion of ’751 dependent claim 15.

8. Dependent ’789 Claim 17

886. Riddle discloses all the limitations of claim 17, which recites: “A method according to claim 13, wherein the state operations include searching the parser record for the existence of one or more reference strings.”

887. Patentee’s ’903 Provisional (incorporated-by-reference into the ’789 Patent) describes searching the packet payload for one or more “specific” or “key” strings, such as the word “image,” “gif,” “PCN-The Poin” (which relates to PointCast), or “.hts HTTP/1.0,” among others.¹⁰⁵²

888. Riddle states it searches for *patterns*, which are reference strings:

[A] traffic class could be created such as all URIs matching “.html” for*

¹⁰⁵¹ ’789 dependent claim 16 also is substantially identical to ’789 dependent claim 14. The only difference is that the former depends from claim 15, while the latter depends from claim 13.

¹⁰⁵² Ex. 1016 (’903 Provisional), 42, 108-110, 223-225.

all servers, or all URI patterns matching “.gif” for server X, or for access to server Y with URI pattern “/sales/*” from client Z, wherein “*” is a wildcard character, i.e., a character which matches all other character combinations.* Traffic class attributes left unspecified will simply match any value for that attribute. For example, a traffic class that accesses data objects within a certain directory path of a web server is *specified by a URI pattern* of the directory path to be managed, e.g. “/sales/*”.¹⁰⁵³

889. Riddle also discloses searching for PointCast URLs that begin with “/FIDO-1/.”¹⁰⁵⁴

890. Riddle further discloses that “Web traffic may also be classified by HTTP header types such as Content-Type (MIME type) or User-Agent.”¹⁰⁵⁵ Moreover, in Riddle’s preferred embodiment, classification can extend to examination of the data contained in a flow’s packets.¹⁰⁵⁶ Accordingly, Riddle discloses state operations that search the parser record for one or more patterns, such as searching for URI patterns matching “*.html”, “*.gif”, a directory path, e.g. “/sales/*” or PointCast URLs that begin with “/FIDO-1/”.¹⁰⁵⁷

¹⁰⁵³ Riddle, 8:67-9:11.

¹⁰⁵⁴ Riddle, 11:47-67.

¹⁰⁵⁵ Riddle, 9:24-26

¹⁰⁵⁶ Riddle, 11:48-49.

¹⁰⁵⁷ Riddle, 8:67:9-11, 11:47-67.

9. Independent '789 Claim 19

- a. *'789 Claim 19's Preamble: "A packet monitor for examining packets passing through a connection point on a computer network, each packets [sic] conforming to one or more protocols, the monitor comprising"*

891. Riddle discloses all elements of this preamble. This preamble is identical to the preamble of '646 claim 1 and disclosed by the prior art for the same reasons. I incorporate by reference my discussion of '646 claim 1's preamble as if fully set forth herein.¹⁰⁵⁸

- b. *'789 Claim Element 19.1: "(a) a packet acquisition device coupled to the connection point and configured to receive packets passing through the connection point"*

892. This claim element is identical to '646 claim element 1.1 and disclosed in the prior art for the same reasons. I incorporate by reference my discussion of '646 claim element 1.1 as if fully set forth herein.¹⁰⁵⁹

¹⁰⁵⁸ '789 claim 19's preamble is also identical to the preambles of '646 claim 7 (except for a corrected typographical error) and '751 claim 17, my discussions of which likewise incorporate my discussion of '646 claim 1's preamble.

¹⁰⁵⁹ '789 claim element 19.1 is also identical to '646 claim element 7.1 and '751 claim element 17.1, my discussions of which likewise incorporate my discussion of '646 claim element 1.1.

- c. *'789 Claim Element 19.2: "(b) an input buffer memory coupled to and configured to accept a packet from the packet acquisition device"*

893. This claim element is identical to '646 claim element 7.2 and is disclosed in the prior art for the same reasons. I incorporate by reference my discussion of '646 claim element 7.2 as if fully set forth herein.

- d. *'789 Claim Element 19.3: "(c) a parser subsystem coupled to the input buffer memory and including a slicer, the parsing subsystem configured to extract selected portions of the accepted packet and to output a parser record containing the selected portions"*

894. This claim element is identical to '646 claim element 7.3 except '789 claim element 19.3 also recites "and including a slicer." I incorporate by reference my discussion of '646 claim element 7.3 as if fully set forth herein.

895. The '789 Patent describes that the extraction engine is a slicer, which is shown below in '789 Patent's Figure 3.¹⁰⁶⁰ The '789 Patent also states that: "For each protocol recognized, the slicer extracts important packet elements from the packet. These form a signature (i.e., key) for the packet. The slicer also preferably generates a hash for rapidly identifying a flow that may have this signature from a database of known flows."¹⁰⁶¹ Further, "[t]he protocol table includes the parameters needed by the pattern analysis and recognition process 304 (implemented by

¹⁰⁶⁰ '789 Patent, 5:59 ("extraction engine (slicer)"), 22:31-32 ("the extraction engine (also called a 'slicer') 1007").

¹⁰⁶¹ '789 Patent, 6:20-24.

PRE 1006) to evaluate the header information in the packet that is associated with that protocol, and parameters needed by extraction process 306 (implemented by slicer 1007) to process the packet header.”¹⁰⁶² Additionally, “the operations of the extraction engine [i.e., the slicer] are those carried out in blocks 306 and 312 of FIG. 3.”¹⁰⁶³

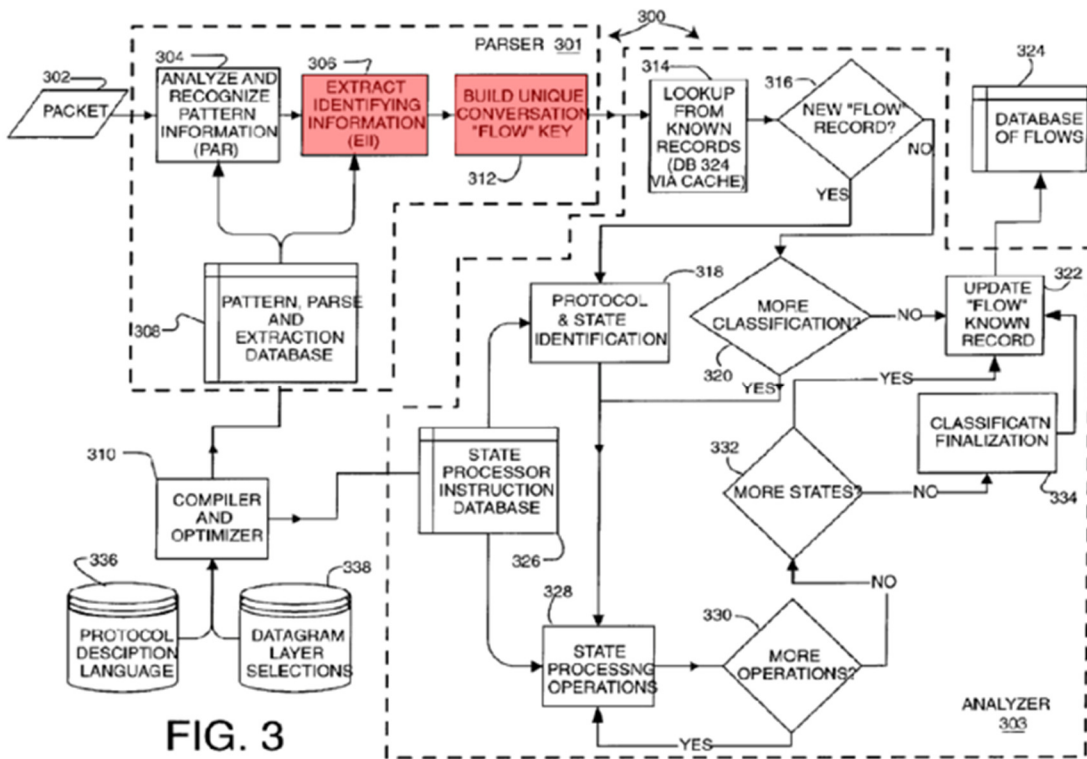


FIG. 3

896. Riddle teaches the claimed “slicer” under that term’s ordinary customary meaning and the construction agreed upon by Patentee and Palo Alto Networks. As discussed regarding ’646 claim element 7.3, Riddle teaches a parser subsystem that

¹⁰⁶² ’789 Patent, 17:44-51.

¹⁰⁶³ ’789 Patent, 22:35-37.

extracts selected portions of packets. That discussion also shows how Riddle discloses a slicer by way of demonstrating that Riddle parses packets into flow specifications. As such, Riddle discloses '789 claim element 19.3 for the same reasons as set forth above for '646 claim element 7.3. I incorporate by reference my discussion of '646 claim element 7.3 as if fully set forth herein.

- e. *'789 Claim Element 19.4: “(d) a memory for storing a database comprising none or more flow-entries for previously encountered conversational flows, each flow-entry identified by identifying information stored in the flow-entry”*

897. This claim element is similar to '646 claim element 1.2 and disclosed or rendered obvious by the prior art for the same reasons. I incorporate by reference my discussion of '646 claim element 1.2 as if fully set forth herein.¹⁰⁶⁴

898. To the extent Patentee argues that '789 elements 19.2 and 19.4 require separate physical memories, i.e., one input buffer memory and one memory for storing a database, Riddle in view of Ferdinand renders these elements obvious for the same reasons set forth above in my discussion of '646 claim element 7.4, which I also incorporate by reference.

¹⁰⁶⁴ '789 claim element 19.4 is also substantially similar to '646 claim element 7.4, my discussion of which likewise incorporates my discussion of '646 claim element 1.2.

- f. '789 Claim Element 19.5: “(e) a lookup engine coupled to the output of the parser subsystem and to the flow-entry memory and configured to lookup whether the particular packet whose parser record is output by the parser subsystem has a matching flow-entry, the looking up using at least some of the selected packet portions and determining if the packet is of an existing flow; and”*

899. This claim element is identical to '646 claim element 7.5 and is disclosed in the prior art for the same reasons. I incorporate by reference my discussion of '646 claim element 7.5 as if fully set forth herein.

- g. '789 Claim Element 19.6: “(f) a flow insertion engine coupled to the flow-entry memory and to the lookup engine and configured to create a flow-entry in the flow-entry database, the flow-entry including identifying information for future packets to be identified with the new flow-entry”*

900. This claim element is identical to '646 claim element 7.7 and is disclosed in the prior art for the same reasons. I incorporate by reference my discussion of '646 claim element 7.7 as if fully set forth herein.

901. To the extent Patentee argues that Riddle's processor programmed with code cannot satisfy the “parser subsystem,” “lookup engine,” and/or “flow insertion engine” in '789 claim elements 19.3, 19.5, 19.6, 19.7, 19.8, or 19.9 because these limitations require separate pieces of hardware, it would have been obvious to a POSITA to modify Riddle's processor and programming code to be separate hardware components. This is because using dedicated hardware for various functions,

especially functions as common as parsing, data lookup, and protocol/state identification, would have been readily understood by a POSITA. And the Challenged Patents acknowledge that a POSITA would have appreciated the benefits and drawbacks of using separate hardware components versus software running on fast processors:

Each of the individual hardware elements through which the data flows in the system are now described with reference to FIGS. 10 and 11. Note that while we are describing a particular hardware implementation of the invention embodiment of FIG. 3, *it would be clear to one skilled in the art that the flow of FIG. 3 may alternatively be implemented in software running on one or more general-purpose processors, or only partly implemented in hardware.* An implementation of the invention that can operate in software is shown in FIG. 14. The hardware embodiment (FIGS. 10 and 11) can operate at over a million packets per second, while the software system of FIG. 14 may be suitable for slower networks. *To one skilled in the art it would be clear that more and more of the system may be implemented in software as processors become faster.*¹⁰⁶⁵

902. Further, Ferdinand discloses its monitor can include separate hardware components for performing various functions, such as real time parser (RTP) 32, database 36, boot/load 22, and memory transport module 34, event manager 38, and

¹⁰⁶⁵ '789 Patent, 21:32-47; '099 Patent, 21:25-38.

control module 42.¹⁰⁶⁶ As provided below, Ferdinand's Figure 5 illustrates its monitor having separate hardware components.

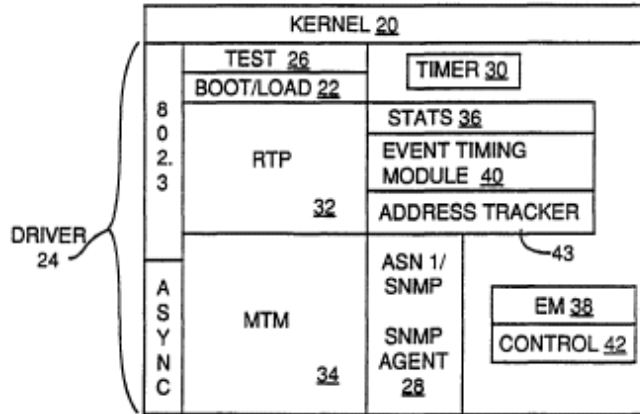


FIG 5

903. Desiring increased performance, a POSITA would have been motivated to utilize dedicated hardware components for parsing/extraction, lookups, and insertions. On the other hand, a POSITA would have understood that Riddle's use of a processor for these functions is less expensive and a more extensible solution than using dedicated hardware components.

¹⁰⁶⁶ Ferdinand, 19:5-13.

h. '789 Claim Element 19.7: "the lookup engine configured such that if the packet is of an existing flow, the monitor classifies the packet as belonging to the found existing flow; and"

904. This claim element is identical to '646 claim element 7.8 and is disclosed in the prior art for the same reasons. I incorporate by reference my discussion of '646 claim element 7.8 as if fully set forth herein.

i. '789 Claim Element 19.8: "if the packet is of a new flow, the flow insertion engine stores a new flow-entry for the new flow in the flow-entry database, including identifying information for future packets to be identified with the new flow-entry"

905. This claim element is identical to '646 claim element 7.9 and is disclosed in the prior art for the same reasons. I incorporate by reference my discussion of '646 claim element 7.9 as if fully set forth herein.

j. '789 Claim Element 19.9: "wherein the operation of the parser subsystem depends on one or more of the protocols to which the packet conforms"

906. This claim element is identical to '646 claim element 7.10 and is disclosed in the prior art for the same reasons. I incorporate by reference my discussion of '646 claim element 7.10 as if fully set forth herein.

907. For the above reasons, it is my opinion that Riddle in view of Ferdinand renders obvious independent claim 19 of the '789 Patent.

10. Dependent '789 Claim 20

908. Claim 20 recites: “A monitor according to claim 19, wherein each packet passing through the connection point is accepted by the packet buffer memory and examined by the monitor in real time.”

909. In my discussion of '789 claim element 19.1, I explained that Riddle discloses a packet acquisition device coupled to the connection point and configured to receive packets passing through the connection point. In my discussion of '789 claim element 19.2, I explained that Riddle teaches an input buffer memory coupled to and configured to accept a packet from the packet acquisition device. In my discussion of '099 claim 1's preamble, I explained that Riddle discloses monitoring packets in real time. And as I discuss regarding '789 claim 2, Riddle monitors each packet passing through the connection point in real time. For the same reasons set forth in those discussions, which I incorporate by reference as if fully set forth herein, Riddle discloses claim 20's “wherein each packet passing through the connection point is accepted by the packet buffer memory and examined by the monitor in real time.”

11. Dependent '789 Claim 42

910. Riddle discloses all the limitations of claim 42, which recites: “A monitor according to claim 19, wherein the lookup engine begins processing as soon as a parser record arrives from the parser subsystem.”

911. As I noted in the previous section, and as I explained in my discussion of '099 claim 1's preamble, Riddle discloses monitoring packets in real-time. And as previously discussed, a POSITA would have understood that real-time monitoring occurs when Riddle's lookup engine begins processing as soon as a parser record arrives from the parser subsystem.

912. For example, Riddle discloses examining packets in real-time to manage network bandwidth based on flow classification. Riddle emphasizes the need to allocate bandwidth to flows based on information ascertainable from multiple OSI layers of each flow, which improves data transmission efficiency and avoids unwanted data transfer stoppage.¹⁰⁶⁷ To improve bandwidth management, Riddle describes "policies to assign available bandwidth from a single logical link to network flows."¹⁰⁶⁸ From these disclosures, a POSITA would have understood that Riddle teaches traffic-examination and classification in real-time to assign available bandwidth.

913. Indeed, Riddle seeks to provide "a method for analyzing real traffic in a customer's network and *automatically* producing a list of the 'found traffic'"¹⁰⁶⁹ Riddle specifies that its monitor can determine a packet's traffic class, such as Real

¹⁰⁶⁷ Riddle, 1:54-61, 2:4-13.

¹⁰⁶⁸ Riddle, 2:66-67.

¹⁰⁶⁹ Riddle, 3:67-4:2, Abstract, 3:32-39, 4:6-9, 10:57-59.

Time Protocol used for point-to-point telephony.¹⁰⁷⁰ Based on these disclosures, a POSITA would have understood that Riddle teaches examining packets in real-time to optimally assign bandwidth based on flow rates.

914. As discussed above in Section IV.A.1, Riddle incorporates-by-reference the teachings of U.S. Patent 6,046,980 (“Packer”) as though fully set forth in Riddle. Like Riddle, Packer describes a traffic classification system and specifies that traffic flows are examined “continuously and automatically.”¹⁰⁷¹ This teaching further shows that Riddle discloses examining packets in real-time by its lookup engine processing as soon as a parser record arrives from the parser.

915. Thus, for all the above reasons, it is my opinion that Riddle in view of Ferdinand renders obvious claims 1, 2, 13-17, 19, 20, and 42 of the ’789 Patent.

B. For the ’789 Patent, Riddle in View of Ferdinand and Further in View of Baker Renders Obvious Dependent Claim 31

916. It is my opinion that a POSITA would have recognized that each and every limitation of ’789 claim 31 is disclosed or rendered obvious by the prior art. Specifically, it is my opinion that ’789 claim 31 is obvious over Riddle in view of Ferdinand and further view of Baker.

¹⁰⁷⁰ Riddle, 12:3-12.

¹⁰⁷¹ Ex. 1031 (Packer), 4:12-16.

1. Reasons to Modify the Combination of Riddle and Ferdinand Further in View of Baker

917. As described above with respect to the '099 Patent in Section VII.B.1 and VII.B.2, a POSITA would have been motivated and found it obvious to combine the teachings of Riddle in view of Ferdinand and further in view of Baker.

2. Dependent '789 Claim 31

918. Claim 31 recites:

A packet monitor according to claim 19, further comprising a compiler processor coupled to the parsing/extraction operations memory,¹⁰⁷² the compiler processor configured to run a compilation process that includes receiving commands in a high-level protocol description language that describe the protocols that may be used in packets encountered by the monitor and any children protocols thereof, and translating the protocol description language commands into a plurality of parsing/extraction operations that are initialized into the parsing/extraction operations memory.

919. This claim is identical to '099 claim 4, and disclosed in the prior art for the same reasons. I incorporate by reference my discussions of '099 claim 4 as if fully set forth herein.

¹⁰⁷² As best understood, it appears that “the parsing/extraction operations memory” may refer to '789 claim 19’s “input buffer memory.”

C. For the '789 Patent, Riddle in View of Ferdinand and Further in View of Wakeman Renders Obvious Dependent Claims 33 and 34

920. It is my opinion that a POSITA would have recognized that each and every limitation of '789 claims 33 and 34 is disclosed or rendered obvious by the prior art. Specifically, it is my opinion that '789 claims 33 and 34 are obvious over Riddle in view of Ferdinand and Wakeman.

1. Reasons to Modify the Combination of Riddle and Ferdinand Further in View of Wakeman

921. As described above with respect to the '646 Patent in Section VII.A.1, a POSITA would have been motivated and found it obvious to combine the teachings of Riddle in view of Ferdinand and further in view of Wakeman.

2. Dependent '789 Claim 33

922. Claim 33 recites: "A packet monitor according to claim 19, further comprising: a cache subsystem coupled to and between the lookup engine and the flow-entry database memory providing for fast access of a set of likely-to-be-accessed flow entries from the flow-entry database." This claim is identical to '646 claim element 7.6, similar to '646 claim elements 1.3 and 1.4, and disclosed in the prior art for the same reasons. I incorporate by reference my discussions of those elements as if fully set forth herein.

3. Dependent '789 Claim 34

923. Claim 34 recites: “A packet monitor according to claim 33, wherein the cache subsystem is an associative cache subsystem including one or more content addressable memory cells (CAMs).” This claim is identical to '646 dependent claim 3 and disclosed in the prior art the same reasons. I incorporate by reference my discussion of '646 dependent claim 3 as if fully set forth herein.

924. Thus, for all the above reasons, it is my opinion that Riddle in view of Ferdinand and Wakeman renders obvious claims 33 and 34 of the '789 Patent.

D. For the '789 Patent, Riddle in View of Ferdinand and Hasani Renders Obvious Claims 44, 48, and 49

925. It is my opinion that a POSITA would have recognized that each and every limitation of '789 claims 44, 48, and 49 is disclosed or rendered obvious by the prior art. Specifically, it is my opinion that '789 claims 44, 48, and 49 are obvious over Riddle in view of Ferdinand and Hasani.

1. Reasons to Modify the Combination of Riddle and Ferdinand Further in View of Hasani

926. Riddle and Hasani are in the same field of endeavor and contain overlapping disclosures with similar purposes. As described above in Sections IV.A and VII.A.1, Riddle discloses a packet monitor that parses packets and stores pars-

ing/extraction operations to carry out state operations to identify a previously-encountered conversational flow or to store a new conversational flow.¹⁰⁷³ Further, Riddle seeks to address “methods for automatically classifying packet traffic based upon information gathered from ... multiple layers in a multi-layer protocol network.”¹⁰⁷⁴ And Riddle details methods to automatically classify traffic based on protocol layer independent categories:

Network traffic is automatically classified under existing classes, beginning with the broadest classes, an inbound traffic class and an outbound traffic class, *in protocol layer independent categories. For example, a particular instance of traffic may be classified according to its transport layer characteristics, e.g., Internet Protocol port number, as well as its application layer information, e.g., SMTP.* Characteristics such as MIME types may also be automatically identified. Standard protocols, such as, IPX, SNA, and services, such as, SMTP and FTP are recognized for automatic classification. *Classification is performed to the most specific level determinable. For example, in select embodiments, non-IP traffic, such as SNA, may be classified only by protocol, whereas Internet Protocol traffic may be classified to the /etc/services level.* Classification beyond a terminal classification level is detected and prevented. For example, in a select embodiment, a class

¹⁰⁷³ E.g., Riddle, 8:47-9:27, 12:26-53, claim 8, Figs. 4A-4B.

¹⁰⁷⁴ Riddle, 3:36-39.

matching “ipx” or “nntp” will not be further automatically classified.¹⁰⁷⁵

927. And as described above in Section IV.G, Hasani discloses a packet parser and a database of parsing/extraction operations.¹⁰⁷⁶ And Hasani details that its stored parsing/extraction operations include information used to determine protocol dependent extraction operations from packet data.¹⁰⁷⁷

928. As I describe below regarding ’789 claim element 44.2 and claim 49, a POSITA would have been motivated and found it obvious for Riddle’s stored parsing/extraction operations to include information on how to determine protocol dependent extraction operations from packet data that indicate a protocol used in the packet. A POSITA further would have been motivated to do so because Hasani teaches protocol dependent packet extraction operations aid in extracting information specific to well-known standards, such as IEEE 802.2.¹⁰⁷⁸

929. Further, a POSITA would have been motivated to have Riddle’s parsing/extraction operations include information used to determine protocol dependent extraction operations from packet data because the POSITA would have appreciated

¹⁰⁷⁵ Riddle, 10:57-11:9; 12:1-12 (discussing detection of RTP and RTSP protocols).

¹⁰⁷⁶ Hasani, 6:12-57, Figs. 3, 4, 5B.

¹⁰⁷⁷ Hasani, 5:9-34, 11:28-39.

¹⁰⁷⁸ Hasani, 5:9-34, 11:28-39.

that this modification would make access and maintenance of these operations more efficient.¹⁰⁷⁹

2. Independent '789 Claim 44

- a. '789 Claim 44 Preamble: "A method of examining packets passing through a connection point on a computer network, the method comprising"*

930. Riddle discloses all elements of this preamble. This preamble is similar to the preamble of '646 claim 1 and disclosed in the prior art for the same reasons. I incorporate by reference my discussion of the preamble of '646 claim 1 as if fully set forth herein.

- b. '789 Claim Element 44.1: "(a) receiving a packet from a packet acquisition device"*

931. This claim element is similar to '646 claim element 1.1 and disclosed in the prior art for the same reasons. I incorporate by reference my discussion of '646 claim element 1.1 as if fully set forth herein.¹⁰⁸⁰

¹⁰⁷⁹ '789 Prosecution History, 176-177 (10/05/2004 Office Action, p.3).

¹⁰⁸⁰ '789 claim element 44.1 is also identical to '646 claim element 16.1 and '789 claim element 1.1, my discussions of which likewise incorporate my discussion of '646 claim element 1.1

- c. *'789 Claim Element 44.2: “(b) performing one or more parsing/extraction operations on the packet according to a database of parsing/extraction operations to create a parser record comprising a function of selected portions of the packet, the database of parsing/extraction operations including information on how to determine a set of one or more protocol dependent extraction operations from data in the packet that indicate a protocol is used in the packet”*

932. Riddle discloses this claim element, which is similar to '789 claim element 1.2, the only difference being that this element recites “a database of parsing/extraction,” with “the database of parsing/extraction operations including information on how to determine a set of one or more protocol dependent extraction operations from data in the packet that indicate a protocol is used in the packet.” Further, '789 claim element 1.2 is similar to '646 claim element 7.3, and I incorporate by reference my discussion of '789 claim element 1.2 and '646 claim element 7.3 as if fully set forth herein. As explained in those discussions, Riddle discloses performing one or more parsing/extraction operations on the packet to create a parser record comprising a function of selected portions of the packet.

933. Further, Riddle discloses a database of parsing/extraction operations including information on how to determine a set of one or more protocol-dependent ex-

traction operations from data in the packet. For example, Riddle discloses that traffic classes are determined using a knowledge base that may be stored in a database 306 in memory.¹⁰⁸¹

934. Riddle also discloses that the knowledge base 306 stored in memory includes information describing how to determine at least one of the protocols used by a packet in a flow from data in the packet, i.e., the heuristics for determining traffic classes. Riddle relates “to digital packet telecommunications, and particularly to management of network bandwidth based on information ascertainable from multiple layers of OSI network model.”¹⁰⁸² And the OSI network model is prior art to both Riddle and the Challenged Patents, as Riddle’s Figure 1D confirms:

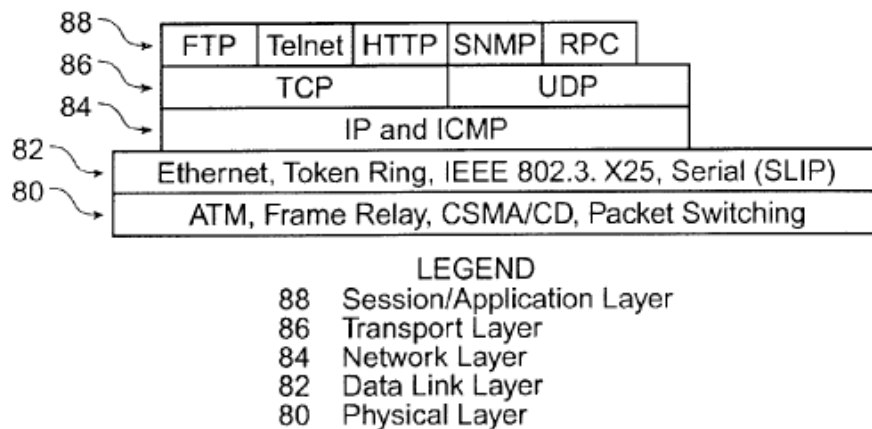


FIG. 1D
(PRIOR ART)

¹⁰⁸¹ Riddle, 12:26-41 (“Automatic Traffic Classification Processing”), Fig. 3.

¹⁰⁸² Riddle, 1:54-57.

935. Riddle realized that “there is no teaching in the prior art of methods for automatically classifying packet traffic based upon information gathered from a [sic] multiple layers in a multi-layer protocol network.”¹⁰⁸³

936. Riddle teaches methods for automatically classifying packet flows for use in allocating bandwidth resources by a rule of assignment of a service level.¹⁰⁸⁴ The methods disclosed in Riddle comprise “applying individual instances of traffic classification paradigms to packet network flows based on selectable information obtained from a plurality of layers of a multi-layered communication protocol in order to define a characteristic class, then mapping the flow to the defined traffic class.”¹⁰⁸⁵ Riddle’s “automatic classification is sufficiently robust to classify a complete enumeration of the possible traffic.”¹⁰⁸⁶ Further, Riddle’s invention provides “techniques to automatically classify a plurality of heterogeneous packets in a packet telecommunications system for management of network bandwidth in systems such as a private area network, a wide area network or an internetwork.”¹⁰⁸⁷

937. In Riddle, a traffic class is defined as, “All traffic between a client and a server endpoints. A single instance of a traffic class is called a flow. Traffic classes

¹⁰⁸³ Riddle, 3:36-39.

¹⁰⁸⁴ Riddle, Abstract, 4:7-10.

¹⁰⁸⁵ Riddle, Abstract, 4:10-15.

¹⁰⁸⁶ Riddle, Abstract, 4:15-17.

¹⁰⁸⁷ Riddle, 4:55-60.

have properties or class attributes such as, directionality, which is the property of traffic to be flowing inbound or outbound.”¹⁰⁸⁸ Riddle further defines, “[a] flow is a single instance of a traffic class. For example, all packets in a TCP connection belong to the same flow. As do all packets in a UDP session.”¹⁰⁸⁹

938. When Riddle classifies traffic, it determines the protocols being used.¹⁰⁹⁰ For instance, Riddle could define a traffic class for an FTP application using a client IP address inside a network, and server IP address outside a network.¹⁰⁹¹ Further, Riddle teaches that it will automatically classify traffic based on protocol layer independent categories,¹⁰⁹² and provide additional confirmation that protocols are detected.¹⁰⁹³ And from these disclosures, a POSITA would have understood Riddle to disclose a storage system 35 for storing instructions (code) in form of a knowledge base embodied in a relational database, for parsing and extracting information from flows (i.e., from the packets of a flow) to classify packets or flows into traffic classes by determining one or more protocols used in the flow.

¹⁰⁸⁸ Riddle, 5:42-45

¹⁰⁸⁹ Riddle, 5:17-20.

¹⁰⁹⁰ Riddle, 8:47-9:27 (“Traffic Class” and “Classifying Traffic”). Additionally, Table 2 provides examples of information from which traffic classes may be built. Riddle, 9:64-65. A traffic class could be defined for an FTP application using a client IP address inside a network, and server IP address outside a network.

¹⁰⁹¹ Riddle, Table 2, 9:64-65.

¹⁰⁹² Riddle, 10:57-11:9 (“Automatic Traffic Classification”).

¹⁰⁹³ Riddle, 12:1-12 (discussing detecting Real Time Protocol (RTP) and Real Time Streaming Protocol (RTSP)).

939. To the extent Patentee argues that Riddle’s knowledge base 306 does not contain the claimed “information on how to determine a set of one or more protocol dependent extraction operations from data in the packet that indicate a protocol is used in the packet,” a POSITA would have found it obvious to include that information based on Riddle, a POSITA’s own knowledge, or Hasani.

940. As discussed above, Riddle discloses performing state operations including determining and updating statistical metrics such as the duplicate count, the most recent time traffic with the same identifying characteristics was encountered, and the byte count. Each of these state operations is dependent upon the protocol since Riddle discloses creating traffic classes based upon protocols.¹⁰⁹⁴

941. Moreover, as noted in my discussion of ‘099 claim element 1.2, Riddle discloses “[a] knowledge base 306” that “contains heuristics for determining traffic classes.”¹⁰⁹⁵ And a POSITA would have understood that the state operations that

¹⁰⁹⁴ Riddle, Table 2, Figs. 2A, 2B, 8:46-9:27, 10:27-11:23. When received packets match a traffic class (step 406), Riddle then determines the received time (step 408), counts duplicates (step 410), and determines byte count (step 412). Riddle, Fig. 4A, 12:42-60. Each of these state operations are dependent upon the protocol because they are determined when received packets match a traffic class which is protocol specific.

¹⁰⁹⁵ Riddle, 12:32-35.

Riddle performs are stored in knowledge base 306 because they relate to how Riddle classifies traffic based upon protocols (i.e., the stored heuristics are “protocol dependent”).

942. Further, Hasani discloses storing parsing and extraction operations for packets in a database.¹⁰⁹⁶ In particular, Hasani details that its stored parsing and extraction operations include information used to determine protocol dependent extraction operations from packet data.¹⁰⁹⁷ For example, Hasani discloses creating a forwarding vector based on stored protocol-dependent information:

Each of the MAC layer, LLC layer, and PID fields contains a variety of allowable data. Parser 180 has as input: the results of reading the MAC layer fields as presented on RMC bus 172; and the results of parser 180 reading the MAC, the LLC header fields, and the PID field of the packet from the RMC bus 172. Parser 180 also has as input a parser database 182. A memory allocation diagram 230 for parser database 182 is shown in FIG. 5B. By comparing the contents of the parser database 182 with the contents of the fields of the packet, the parser creates a forwarding vector for the packet. The forwarding vector is transferred on line 184 to control block 186 of the packet memory, and also to packet memory controller 188. The forwarding vector then determines the fate of the packet by providing information to the packet memory controller 188.¹⁰⁹⁸

¹⁰⁹⁶ Hasani, 6:27-57, Figs. 3, 4 (parser database 182), 5.

¹⁰⁹⁷ Hasani, 5:9-34, 11:28-39.

¹⁰⁹⁸ Hasani, 6:12-26.

943. Similarly, Hasani teaches protocol-dependent packet filtering:

Protocol IDentifications, PID, Field Filtering

Section 240, BLOCK 10-14, is allocated to filtering the PID field.

LLC filtering based on PID is required for SNAP SAP packets. The Protocol IDentifier field, PID, of the packet is five (5) bytes long. FIG. 11 shows a PID entry in the parser database, found in PID section 240 of FIG. 5B. The description of PID Section 240 is similar to the description for the destination addresses, DA section 234, except that the PID section 240 has five blocks of memory rather than six (6), corresponding to the PID field of the packet. Also, BIT <63> in each entry is reserved for the Unknown user.

The DSAP entries and PID entries are two mutually exclusive filtering fields. That is, a packet is either filtered on DSAP or on PID. For a SNAP SAP packet the filtering is based upon the five (5) bytes of PID. For a NON SNAP PACKET the filtering is based on one byte of DSAP. From the DSAP and the PID fields, the parser supports 64 unique values.¹⁰⁹⁹

944. Hasani teaches that such protocol-dependent packet extraction operations aid in extracting information specific to well-known standards, such as IEEE 802.2.¹¹⁰⁰

As Hasani notes, “the standard IEEE 802.2 MAC and LLC headers may contain a maximum of twenty two (22) bytes,” and the “arrival rate of bits is, in some cases,

¹⁰⁹⁹ Hasani, 11:28-46.

¹¹⁰⁰ Hasani, 5:9-34, 11:28-39.

faster than the CPU of the host computer can execute software to read the packets into memory.”¹¹⁰¹ Hasani’s relational database therefore is necessary to sort and store the packets arriving at a computer from a LAN at the speed at which the bits arrive at the interface.¹¹⁰²

945. Thus, a POSITA would have known that a relational database could store protocol-dependent state operations, and based on Riddle’s disclosures or in view of Hasani, would have been motivated to store “protocol dependent state operations” in Riddle’s database because it would make access and maintenance of these operations more efficient. Further, it would accomplish Riddle’s objective of examining packets in real-time.¹¹⁰³ As such, using Hasani’s database organization in Riddle’s knowledge base 306, amounts to nothing more than combining known prior art technologies used in their ordinary and predictable manner to store parsing/extraction operations.

¹¹⁰¹ Hasani, 1:47-59.

¹¹⁰² E.g., Hasani, 2:1-3.

¹¹⁰³ Riddle, 1:54-2:13, 2:66-67, 3:67-4:2, 12:3-12; Ex. 1031 (Packer, incorporated-by-reference into Riddle), 4:12-16.

- d. '789 Claim Element 44.3: “(c) looking up a flow-entry database comprising none or more flow-entries for previously encountered conversational flows, the looking up using at least some of the selected packet portions, and determining if the packet is of an existing flow”

946. This claim element is substantially identical to '646 claim element 16.3, similar to '646 claim element 1.2, and disclosed in the prior art for the same reasons.¹¹⁰⁴ I incorporate by reference my discussion of '646 claim elements 1.2 and 16.3 as if fully set forth herein.¹¹⁰⁵

- e. '789 Claim Element 44.4: “(d) if the packet is of an existing flow, obtaining the last encountered state of the flow and performing any state operations specified for the state of the flow starting from the last encountered state of the flow; and”

947. This claim element is identical to '789 claim element 13.1, similar to '751 claim element 10.1, and disclosed in the prior art for the same reasons.¹¹⁰⁶ I incorporate by reference my discussion of '789 claim element 13.1 and '751 claim element 10.1 as if fully set forth herein.

¹¹⁰⁴ The only difference between '789 claim element 44.3 and '646 claim element 16.3 is that the latter requires “the lookup being via a cache.”

¹¹⁰⁵ '789 claim element 44.3 is also identical to '789 claim element 1.3, my discussion of which likewise incorporates my discussion of '646 claim element 1.1

¹¹⁰⁶ '751 claim element 10.1 requires “if the packet is of an existing flow, identifying the last encountered state of the flow and performing any state operations specified for the state of the flow starting from the last encountered state of the flow.”

f. '789 Claim Element 44.5: “(e) if the packet is of a new flow, performing any analysis required for the initial state of the new flow and storing a new flow-entry for the new flow in the flow-entry database, including identifying information for future packets to be identified with the new flow-entry”

948. This claim element is similar to '751 claim element 1.5, with the only difference being that this element requires “performing any analysis required for the initial state of the new flow.” All limitations of '789 claim element 44.5 are disclosed in the prior art for the reasons set forth above in my discussions of '751 claim element 1.5 and '646 claim element 1.5, which explains how Riddle discloses “performing any state operations required for the initial state of the new flow.”¹¹⁰⁷ I incorporate by reference my discussions of '751 claim element 1.5 and '646 claim element 1.5 as if fully set forth herein.

949. Thus, for all the above reasons, it is my opinion that Riddle in view of Ferdinand and Hasani renders obvious independent claim 44 of the '789 Patent.

3. Dependent '789 Claim 48

950. Riddle discloses all the limitations of claim 48, which recites: “A method according to claim 44, further comprising forming a signature from the selected packet portions, wherein the lookup operation uses the signature and wherein the

¹¹⁰⁷ '646 claim element 1.5 recites a state processor “to perform any state operations required for the initial state of the new flow in the case that the packet is [not] from an existing flow.”

identifying information stored in the new or updated flow-entry is a signature for identifying future packets.”

951. The ’789 Patent describes a signature as “selected parts of a packet that will allow monitor 108 to identify efficiently any packets that belong to the same flow.”¹¹⁰⁸ And as shown in Figure 4A, Riddle discloses creating a flow-entry in the flow-entry database for new flows with identifying information, i.e., the signature. That identifying information includes selected portions of the packet, e.g., “identifying characteristics, such as protocol type (SAP), IP protocol number, server port, traffic type if known, MIME type, a time of occurrence of the traffic.”¹¹⁰⁹

952. To the extent Patentee argues that the signature, i.e., identifying characteristics, must be hashed, then such a modification would have been obvious to a POSITA based upon the disclosure of Riddle or a POSITA’s own knowledge for the same reasons set forth above in my discussion of ’646 claim element 7.3, which I incorporate by reference.¹¹¹⁰

¹¹⁰⁸ ’789 Patent, 10:60-63.

¹¹⁰⁹ Riddle, 12:42-59, Fig. 4A at 402 (“parse flow specification from a packet of the flow”), 404 (“compare flow specification with existing classification tree”), 406 (“traffic matches a class?”), 408 (“enter into a saved list characteristics of the traffic”), and 410 (“suppress duplicates”).

¹¹¹⁰ Like the ’646 Patent, the ’789 Patent acknowledges that hashing signatures and the benefits of doing so were well known to POSITAs. ’789 Patent, 13:37-43. As discussed above in Section IV.A.1 and my discussion of ’646 claim element 7.3, Riddle incorporates-by-reference Packer, which teaches using hash tables to index flows for TCP connections. Ex. 1031 (Packer), 15:43-51, Fig. 4A. A POSITA

4. Dependent '789 Claim 49

953. Riddle discloses all the limitations of claim 49 or renders it obvious in view of Hasani. Claim 49 recites: “A method according to claim 44, wherein the state operations are according to a database of protocol dependent state operations.”

954. As discussed above, Riddle discloses performing state operations including determining and updating statistical metrics such as the duplicate count, the most recent time traffic with the same identifying characteristics was encountered, and the byte count. Each of these state operations is dependent upon the protocol since Riddle discloses creating traffic classes based upon protocols.¹¹¹¹

955. Moreover, as noted in my discussion of '099 claim element 1.2, Riddle discloses “[a] knowledge base 306” that “contains heuristics for determining traffic classes.”¹¹¹² And a POSITA would have understood that the state operations that

would have appreciated that the same, well-known function could be applied in the combinations identified herein, decreasing look up times.

¹¹¹¹ Riddle, Table 2, Figs. 2A, 2B, 8:46-9:27, 10:27-11:23. When received packets match a traffic class (step 406), Riddle then determines the received time (step 408), counts duplicates (step 410), and determines byte count (step 412). Riddle, Fig. 4A, 12:42-60. Each of these state operations are dependent upon the protocol because they are determined when received packets match a traffic class which is protocol specific. Additionally, as noted in my discussion of '099 claim element 1.2, Riddle discloses “[a] knowledge base 306” that “contains heuristics for determining traffic classes.” Riddle, 12:32-35. And a POSITA would have understood that the state operations that Riddle performs are stored in knowledge base 306 because they relate to how Riddle classifies traffic.

¹¹¹² Riddle, 12:32-35.

Riddle performs are stored in knowledge base 306 because they relate to how Riddle classifies traffic (i.e., the stored heuristics are “protocol dependent”).

956. To the extent Patentee argues that Riddle’s database does not contain “protocol dependent state operations,” such a modification would be obvious to a POSITA based on Riddle, a POSITA’s own knowledge, or Hasani. As discussed above regarding ’789 claim element 44.2 Hasani discloses storing parsing and extraction operations for packets in a database.¹¹¹³ In particular, Hasani details that its stored parsing and extraction operations include information used to determine protocol dependent extraction operations from packet data.¹¹¹⁴ For example, Hasani teaches protocol-dependent packet filtering and creating a forwarding vector based on stored protocol-dependent information.¹¹¹⁵ And Hasani details that such protocol-dependent packet extraction operations aid in extracting information specific to well-known standards, such as IEEE 802.2.¹¹¹⁶ Thus, Hasani’s relational database was necessary to sort and store the packets arriving at computer from a LAN at the speed at which the bits arrive at the interface.¹¹¹⁷

¹¹¹³ Hasani, 6:27-57, Figs. 3, 4 (parser database 182), 5.

¹¹¹⁴ Hasani, 5:9-34, 11:28-46.

¹¹¹⁵ Hasani, 6:12-26.

¹¹¹⁶ Hasani, 5:9-34, 11:28-39, 1:47-59 (“[T]he standard IEEE 802.2 MAC and LLC headers may contain a maximum of twenty two (22) bytes.... This arrival rate of bits [was], in some cases, faster than the CPU of the host computer can execute software to read the packets into memory.”).

¹¹¹⁷ E.g., Hasani, 2:1-3.

957. As detailed above regarding '789 claim element 44.2, based on Riddle's and Hasani's teachings, a POSITA would have been motivated to store "protocol dependent state operations" in Riddle's database because it would make access and maintenance of these operations more efficient. Further, it would accomplish Riddle's objective of examining packets in real-time.¹¹¹⁸ So even if Riddle did not disclose storing "protocol dependent state operations" in knowledge base 306, doing so amounts to nothing more than combining known prior art technologies used in their ordinary and predictable manner to store parsing/extraction operations.

958. Thus, for all the above reasons, it is my opinion that Riddle in view of Ferdinand and Hasani renders obvious claims 44, 48, and 49 of the '789 Patent.

E. For the '789 Patent, Riddle in View of Ferdinand and Further in View of Yu Renders Obvious Claims 1-2, 13-17, 19-20, and 42.

959. It is my opinion that a POSITA would have recognized that each and every limitation of the '789 claims 1-2, 13-17, 19-20, and 42 is disclosed or rendered obvious in light of Riddle in view of Ferdinand and further in view of Yu. Specifically, my opinions regarding Riddle in view of Ferdinand are exactly the same as those above in Section XI.A, but further include the teachings of Yu. Thus, as if fully set forth here, I incorporate the discussion from Section XI.A regarding the

¹¹¹⁸ Riddle, 1:54-2:13, 2:66-67, 3:67-4:2, 12:3-12; Ex. 1031 (Packer, incorporated-by-reference into Riddle), 4:12-16.

obviousness of '789 claims 1-2, 13-17, 19-20, and 42 over Riddle in view of Ferdinand.

960. As discussed above, all of the Challenged Claims require “conversational flows.” For example, '789 claim element 1.3 recites “flow-entries for previously encountered conversational flows.” While Riddle itself teaches identifying conversational flows, Yu further demonstrates identifying conversational flows through its “flow classification.”¹¹¹⁹ Further, Yu teaches flow classification logic that “keeps track of the flow’s state until matching criteria is met” when identifying whether a packet belongs to a conversational flow.¹¹²⁰

961. As discussed with respect to the obviousness of '099 claims 1 and 2 over Riddle in view of Ferdinand and further in view of Yu in Section VII.C, Yu teaches identifying the claimed “conversational flow” by Yu’s flow classifier linking multiple “streams” into a “flow” based on application or application data.¹¹²¹ I incorporate by reference that discussion as if fully set forth herein.

962. As discussed with respect the obviousness of '099 claims 1 and 2 over Riddle in view of Ferdinand and further in view of Yu in Section VII.C, I explain how a POSITA would have been motivated to combine Yu’s teachings into Riddle’s monitor. I incorporate by reference that discussion as if fully set forth herein. For

¹¹¹⁹ Yu, 1:56-60, 3:32-36, 3:47-49; 4:1-8.

¹¹²⁰ Yu, 4:62-64.

¹¹²¹ Yu, 1:56-60, 3:32-49; 4:1-8.

the same reasons, it is my opinion that combining the teachings of Riddle, Ferdinand, and Yu renders obvious all the claim elements relating to “conversational flows.”

963. As set forth in my analysis of the ’789 Patent in Sections XI.A.2 through XI.A.11 above, Riddle and Ferdinand disclose or render obvious all the remaining elements of ’789 claims 1-2, 13-17, 19-20, and 42. Thus, it is my opinion that Riddle in view of Ferdinand and further in view of Yu renders obvious ’789 claims 1-2, 13-17, 19-20, and 42.

F. For the ’789 Patent, Riddle in View of Ferdinand and Baker and Further in View of Yu Renders Obvious Claim 31.

964. It is my opinion that a POSITA would have recognized that each and every limitation of the ’789 claim 31 is disclosed or rendered obvious in light of Riddle in view of Ferdinand and Baker and further in view of Yu. Specifically, my opinions regarding Riddle in view of Ferdinand and Baker are exactly the same as those above in Section XI.B, but further include the teachings of Yu. Thus, as if fully set forth here, I incorporate the discussion from Section XI.B regarding the obviousness of ’789 claim 31 over Riddle in view of Ferdinand and Baker.

965. As discussed above, all of the Challenged Claims require “conversational flows.” For example, ’789 claim element 19.4 recites “flow-entries for previously encountered conversational flows” wherein claim 31 depends from claim 19.

While Riddle itself teaches identifying conversational flows, Yu further demonstrates identifying conversational flows through its “flow classification.”¹¹²² Further, Yu teaches flow classification logic that “keeps track of the flow’s state until matching criteria is met” when identifying whether a packet belongs to a conversational flow.¹¹²³

966. As discussed with respect to the obviousness of ’099 claims 1 and 2 over Riddle in view of Ferdinand and further in view of Yu in Section VII.C, Yu teaches identifying the claimed “conversational flow” by Yu’s flow classifier linking multiple “streams” into a “flow” based on application or application data.¹¹²⁴ I incorporate by reference that discussion as if fully set forth herein.

967. As discussed with respect the obviousness of ’099 claims 1 and 2 over Riddle in view of Ferdinand and further in view of Yu in Section VII.C, I explain how a POSITA would have been motivated to combine Yu’s teachings into Riddle’s monitor. I incorporate by reference that discussion as if fully set forth herein. For the same reasons, it is my opinion that combining the teachings of Riddle, Ferdinand, Baker, and Yu renders obvious all the claim elements relating to “conversational flows.”

968. As set forth in my analysis of the ’789 Patent in Sections XI.B.2 above,

¹¹²² Yu, 1:56-60, 3:32-36, 3:47-49; 4:1-8.

¹¹²³ Yu, 4:62-64.

¹¹²⁴ Yu, 1:56-60, 3:32-49; 4:1-8.

Riddle, Ferdinand, and Baker disclose or render obvious all the remaining elements of '789 claim 31 which depends from claim 19. Thus, it is my opinion that Riddle in view of Ferdinand and Baker and further in view of Yu renders obvious '789 claim 31.

G. For the '789 Patent, Riddle in View of Ferdinand and Wakeman and Further in View of Yu Renders Obvious Claims 33-34.

969. It is my opinion that a POSITA would have recognized that each and every limitation of the '789 claims 33-34 is disclosed or rendered obvious in light of Riddle in view of Ferdinand and Wakeman and further in view of Yu. Specifically, my opinions regarding Riddle in view of Ferdinand and Wakeman are exactly the same as those above in Section XI.C, but further include the teachings of Yu. Thus, as if fully set forth here, I incorporate the discussion from Section XI.C regarding the obviousness of '789 claims 33-34 over Riddle in view of Ferdinand and Wakeman.

970. As discussed above, all of the Challenged Claims require “conversational flows.” For example, '789 claim element 19.4 recites “flow-entries for previously encountered conversational flows” wherein claims 33-34 depend from claim 19.

While Riddle itself teaches identifying conversational flows, Yu further demonstrates identifying conversational flows through its “flow classification.”¹¹²⁵ Further, Yu teaches flow classification logic that “keeps track of the flow’s state until matching criteria is met” when identifying whether a packet belongs to a conversational flow.¹¹²⁶

971. As discussed with respect to the obviousness of ’099 claims 1 and 2 over Riddle in view of Ferdinand and further in view of Yu in Section VII.C, Yu teaches identifying the claimed “conversational flow” by Yu’s flow classifier linking multiple “streams” into a “flow” based on application or application data.¹¹²⁷ I incorporate by reference that discussion as if fully set forth herein.

972. As discussed with respect the obviousness of ’099 claims 1 and 2 over Riddle in view of Ferdinand and further in view of Yu in Section VII.C, I explain how a POSITA would have been motivated to combine Yu’s teachings into Riddle’s monitor. I incorporate by reference that discussion as if fully set forth herein. For the same reasons, it is my opinion that combining the teachings of Riddle, Ferdinand, Wakeman, and Yu renders obvious all the claim elements relating to “conversational flows.”

973. As set forth in my analysis of the ’789 Patent in Sections XI.C.2 through

¹¹²⁵ Yu, 1:56-60, 3:32-36, 3:47-49; 4:1-8.

¹¹²⁶ Yu, 4:62-64.

¹¹²⁷ Yu, 1:56-60, 3:32-49; 4:1-8.

XI.C.3 above, Riddle, Ferdinand, and Wakeman disclose or render obvious all the remaining elements of '789 claims 33-34 which depends from claim 19. Thus, it is my opinion that Riddle in view of Ferdinand and Wakeman and further in view of Yu renders obvious '789 claims 33-34.

H. For the '789 Patent, Riddle in View of Ferdinand and Hasani and Further in View of Yu Renders Obvious Claims 44 and 48-49.

974. It is my opinion that a POSITA would have recognized that each and every limitation of the '789 claims 44 and 48-49 is disclosed or rendered obvious in light of Riddle in view of Ferdinand and Hasani and further in view of Yu. Specifically, my opinions regarding Riddle in view of Ferdinand and Hasani are exactly the same as those above in Section XI.D, but further include the teachings of Yu. Thus, as if fully set forth here, I incorporate the discussion from Section XI.D regarding the obviousness of '789 Claims 44 and 48-49 over Riddle in view of Ferdinand and Hasani.

975. As discussed above, all of the Challenged Claims require “conversational flows.” For example, '789 claim element 44.3 recites “flow-entries for previously encountered conversational flows.” While Riddle itself teaches identifying conversational flows, Yu further demonstrates identifying conversational flows through its “flow classification.”¹¹²⁸ Further, Yu teaches flow classification logic that

¹¹²⁸ Yu, 1:56-60, 3:32-36, 3:47-49; 4:1-8.

“keeps track of the flow’s state until matching criteria is met” when identifying whether a packet belongs to a conversational flow.¹¹²⁹

976. As discussed with respect to the obviousness of ’099 claims 1 and 2 over Riddle in view of Ferdinand and further in view of Yu in Section VII.C, Yu teaches identifying the claimed “conversational flow” by Yu’s flow classifier linking multiple “streams” into a “flow” based on application or application data.¹¹³⁰ I incorporate by reference that discussion as if fully set forth herein.

977. As discussed with respect the obviousness of ’099 claims 1 and 2 over Riddle in view of Ferdinand and further in view of Yu in Section VII.C, I explain how a POSITA would have been motivated to combine Yu’s teachings into Riddle’s monitor. I incorporate by reference that discussion as if fully set forth herein. For the same reasons, it is my opinion that combining the teachings of Riddle, Ferdinand, Hasani, and Yu renders obvious all the claim elements relating to “conversational flows.”

978. As set forth in my analysis of the ’789 Patent in Sections XI.D.2 through XI.D.4 above, Riddle, Ferdinand, and Hasani disclose or render obvious all the remaining elements of ’789 claims 44 and 48-49. Thus, it is my opinion that Riddle in view of Ferdinand and Hasani and further in view of Yu renders obvious ’789

¹¹²⁹ Yu, 4:62-64.

¹¹³⁰ Yu, 1:56-60, 3:32-49; 4:1-8.

claims 44 and 48-49.

I. For the '789 Patent, Riddle in View of Ferdinand and Further in View of RFC1945 Renders Obvious Claims 1-2, 13-17, 19-20, and 42.

979. It is my opinion that a POSITA would have recognized that each and every limitation of the '789 claims 1-2, 13-17, 19-20, and 42 is disclosed or rendered obvious in light of Riddle in view of Ferdinand and further in view of RFC. Specifically, my opinions regarding Riddle in view of Ferdinand are exactly the same as those above in Section XI.A, but further include the teachings of RFC1945. Thus, as if fully set forth here, I incorporate the discussion from Section XI.A regarding the obviousness of '789 claims 1-2, 13-17, 19-20, and 42 over Riddle in view of Ferdinand.

980. As discussed above, all of the Challenged Claims require “conversational flows.” For example, '789 claim element 1.3 recites “flow-entries for previously encountered conversational flows.” While Riddle itself teaches identifying conversational flows, RFC1945 further demonstrates identifying conversational flows through the additional example of the use of HTTP header fields.

981. As discussed with respect to the obviousness of '099 claims 1 and 2 in view Riddle, Ferdinand, and RFC1945 in Section VII.E, I understand that Patentee and its technical expert have taken the position that the HTTP Referrer field can be used to create a conversational flow. I incorporate by reference that discussion as if

fully set forth herein.

982. As discussed with respect the obviousness of '099 claims 1 and 2 over Riddle in view of Ferdinand and further in view of RFC1945 in Section VII.E, I explain how a POSITA would have been motivated to combine RFC1945's teachings into Riddle's monitor. I incorporate by reference that discussion as if fully set forth herein. For the same reasons, it is my opinion that combining the teachings of Riddle, Ferdinand, and RFC1945 renders obvious all the claim elements relating to "conversational flows."

983. As set forth in my analysis of the '789 Patent in Sections XI.A.2 through XI.A.11 above, Riddle and Ferdinand disclose or render obvious all the remaining elements of '789 claims 1-2, 13-17, 19-20, and 42. Thus, it is my opinion that Riddle in view of Ferdinand and further in view of RFC1945 renders obvious '789 claims 1-2, 13-17, 19-20, and 42.

J. For the '789 Patent, Riddle in View of Ferdinand and Baker and Further in View of RFC1945 Renders Obvious Claim 31.

984. It is my opinion that a POSITA would have recognized that each and every limitation of the '789 claim 31 is disclosed or rendered obvious in light of Riddle in view of Ferdinand and Baker and further in view of RFC1945. Specifically, my opinions regarding Riddle in view of Ferdinand and Baker are exactly the same as those above in Section XI.B, but further include the teachings of RFC1945. Thus, as if fully set forth here, I incorporate the discussion from Section XI.B regarding

the obviousness of '789 claim 31 over Riddle in view of Ferdinand and Baker.

985. As discussed above, all of the Challenged Claims require “conversational flows.” For example, '789 claim element 19.4 recites “flow-entries for previously encountered conversational flows” wherein claim 31 depends from claim 19.

While Riddle itself teaches identifying conversational flows, RFC1945 further demonstrates identifying conversational flows through the additional example of the use of HTTP header fields.

986. As discussed with respect to the obviousness of '099 claims 1 and 2 in view Riddle, Ferdinand, and RFC1945 in Section VII.E, I understand that Patentee and its technical expert have taken the position that the HTTP Referrer field can be used to create a conversational flow. I incorporate by reference that discussion as if fully set forth herein.

987. As discussed with respect the obviousness of '099 claims 1 and 2 over Riddle in view of Ferdinand and further in view of RFC1945 in Section VII.E, I explain how a POSITA would have been motivated to combine RFC1945's teachings into Riddle's monitor. I incorporate by reference that discussion as if fully set forth herein. For the same reasons, it is my opinion that combining the teachings of Riddle, Ferdinand, Baker, and RFC1945 renders obvious all the claim elements relating to “conversational flows.”

988. As set forth in my analysis of the '789 Patent in Sections XI.B.2 above,

Riddle, Ferdinand, and Baker disclose or render obvious all the remaining elements of '789 claim 31 which depends from claim 19. Thus, it is my opinion that Riddle in view of Ferdinand and Baker and further in view of RFC1945 renders obvious '789 claim 31.

K. For the '789 Patent, Riddle in View of Ferdinand and Wakeman and Further in View of RFC1945 Renders Obvious Claims 33-34.

989. It is my opinion that a POSITA would have recognized that each and every limitation of the '789 claims 33-34 is disclosed or rendered obvious in light of Riddle in view of Ferdinand and Wakeman and further in view of RFC1945. Specifically, my opinions regarding Riddle in view of Ferdinand and Wakeman are exactly the same as those above in Section XI.C, but further include the teachings of RFC1945. Thus, as if fully set forth here, I incorporate the discussion from Section XI.C regarding the obviousness of '789 claims 33-34 over Riddle in view of Ferdinand and Wakeman.

990. As discussed above, all of the Challenged Claims require “conversational flows.” For example, '789 claim element 19.4 recites “flow-entries for previously encountered conversational flows” wherein claims 33-34 depend from claim 19. While Riddle itself teaches identifying conversational flows, RFC1945 further demonstrates identifying conversational flows through the additional example of the use of HTTP header fields.

991. As discussed with respect to the obviousness of '099 claims 1 and 2 in view

Riddle, Ferdinand, and RFC1945 in Section VII.E, I understand that Patentee and its technical expert have taken the position that the HTTP Referrer field can be used to create a conversational flow. I incorporate by reference that discussion as if fully set forth herein.

992. As discussed with respect the obviousness of '099 claims 1 and 2 over Riddle in view of Ferdinand and further in view of RFC1945 in Section VII.E, I explain how a POSITA would have been motivated to combine RFC1945's teachings into Riddle's monitor. I incorporate by reference that discussion as if fully set forth herein. For the same reasons, it is my opinion that combining the teachings of Riddle, Ferdinand, Wakeman, and RFC1945 renders obvious all the claim elements relating to "conversational flows."

993. As set forth in my analysis of the '789 Patent in Sections XI.C.2 through XI.C.3 above, Riddle, Ferdinand, and Wakeman disclose or render obvious all the remaining elements of '789 claims 33-34 which depends from claim 19. Thus, it is my opinion that Riddle in view of Ferdinand and Wakeman and further in view of RFC1945 renders obvious '789 claims 33-34.

L. For the '789 Patent, Riddle in View of Ferdinand and Hasani and Further in View of RFC1945 Renders Obvious Claims 44 and 48-49.

994. It is my opinion that a POSITA would have recognized that each and every limitation of the '789 claims 44 and 48-49 is disclosed or rendered obvious in light

of Riddle in view of Ferdinand and Hasani and further in view of RFC1945. Specifically, my opinions regarding Riddle in view of Ferdinand and Hasani are exactly the same as those above in Section XI.D, but further include the teachings of RFC1945. Thus, as if fully set forth here, I incorporate the discussion from Section XI.D regarding the obviousness of '789 Claims 44 and 48-49 over Riddle in view of Ferdinand and Hasani.

995. As discussed above, all of the Challenged Claims require “conversational flows.” For example, '789 claim element 44.3 recites “flow-entries for previously encountered conversational flows.” While Riddle itself teaches identifying conversational flows, RFC1945 further demonstrates identifying conversational flows through the additional example of the use of HTTP header fields.

996. As discussed with respect to the obviousness of '099 claims 1 and 2 in view Riddle, Ferdinand, and RFC1945 in Section VII.E, I understand that Patentee and its technical expert have taken the position that the HTTP Referrer field can be used to create a conversational flow. I incorporate by reference that discussion as if fully set forth herein.

997. As discussed with respect the obviousness of '099 claims 1 and 2 over Riddle in view of Ferdinand and further in view of RFC1945 in Section VII.E, I explain how a POSITA would have been motivated to combine RFC1945's teachings into Riddle's monitor. I incorporate by reference that discussion as if fully set forth

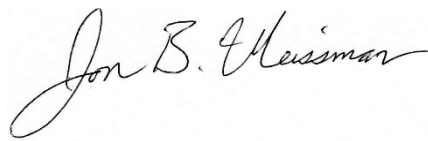
herein. I incorporate by reference that discussion as if fully set forth herein. For the same reasons, it is my opinion that combining the teachings of Riddle, Ferdinand, Hasani, and RFC1945 renders obvious all the claim elements relating to “conversational flows.”

998. As set forth in my analysis of the '789 Patent in Sections XI.D.2 through XI.D.4 above, Riddle, Ferdinand, and Hasani disclose or render obvious all the remaining elements of '789 claims 44 and 48-49. Thus, it is my opinion that Riddle in view of Ferdinand and Hasani and further in view of RFC1945 renders obvious '789 claims 44 and 48-49.

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

Dated: February 3, 2020

By:

A handwritten signature in cursive script that reads "Jon B. Weissman". The signature is written in black ink on a white background.

Jon B. Weissman