

UNITED STATES PATENT AND TRADEMARK OFFICE

PATENT TRIAL AND APPEAL BOARD

Capital One National Association, Hulu, LLC
Petitioner

v.

Implicit, LLC,
Patent Owner

CASE: IPR2024-00643
U.S. PATENT NO. 8,056,075

PETITION FOR *INTER PARTES* REVIEW

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

TABLE OF CONTENTS

MANDATORY NOTICES (37 C.F.R. §42.8(b))	v
A. Real Parties-In-Interest.....	v
B. Related Matters.....	v
C. Lead and Backup Counsel and Service (37 C.F.R. §42.8(b)(3)-(4)) ..	vi
D. Payment of Fees (37 C.F.R. §§42.15 and 42.103)	viii
I. GROUNDS FOR STANDING (37 C.F.R. §42.104(a)).....	1
II. STATEMENT OF PRECISE RELIEF REQUEST AND REASONS THEREFORE	1
III. '075 PATENT OVERVIEW	2
A. Background of the Technology	2
B. Prosecution History	4
IV. CLAIM CONSTRUCTION	5
V. ARGUMENTS	5
A. Level of Ordinary Skill in the Art.....	5
B. Prior Art.....	5
1. Fowlow (Ex.1009)	5
2. Kimera (Ex.1008).....	7
3. Parthasarathy (Ex.1007).....	9
4. Angles (Ex.1010)	12
C. Ground 1: Claims 1-13 are Unpatentable Under 35 U.S.C. §103(a) as Obvious over Fowlow	13
1. Independent Claim 1	13
2. Dependent Claims 2-13.....	25
D. Ground 2: Claims 1-13 are Unpatentable Under 35 U.S.C. §103(a) as Obvious over Fowlow in view of Kimera	33
1. Motivation to Combine	34
2. Independent Claim 1	39
3. Dependent Claims 2-13.....	45

E.	Ground 3: Claims 1-7, 9-13 is Unpatentable Under 35 U.S.C. §103(a) as Obvious over Parthasarathy in view of Kimera	49
1.	Motivation to Combine	49
2.	Independent Claim 1	54
3.	Dependent Claims 2-7, 9-13	63
F.	Ground 4: Claim 8 is Unpatentable Under 35 U.S.C. §103(a) as Obvious over Parthasarathy in view of Kimera and Angles	70
1.	Motivation to Combine	70
2.	Dependent Claim 8	72
G.	The Petition Should Not Be Discretionarily Denied.....	73
1.	<i>Advanced Bionics</i>	73
2.	<i>Fintiv</i>	74
VI.	CONCLUSION.....	75
	CLAIM APPENDIX OF THE CHALLENGED CLAIMS.....	77

LIST OF EXHIBITS

Exhibit No.	Description
1001	U.S. Patent No. 8,056,075 (the “’075 patent”)
1002	Declaration of Erez Zadok, Ph.D.
1003	Curriculum Vitae of Erez Zadok, Ph.D.
1004	Prosecution History of U.S. Patent Application No. 11/933,161
1005	PCT App. No. WO 1998/006033 to Agranat et al. (“Agranat”)
1006	Michael Franz & Thomas Kistler, <u>Slim Binaries</u> , Communications of the ACM, Dec. 1997 (“Franz”)
1007	U.S. Pat. No. 6,347,398 to Parthasarathy et al. (“Parthasarathy”)
1008	Emin Gün Sirer & Brian Bershad, <u>Kimera: A System Architecture for Networked Computers</u> , Univ. of Wash., 1997 (“Kimera”)
1009	U.S. Patent No. 6,260,078 to Fowlow (“Fowlow”)
1010	U.S. Patent No. 5,933,811 to Angles et al. (“Angles”)
1011	<i>Reserved</i>
1012	Affidavit of Nathaniel E. Frank-White, dated 2/9/2024
1013	<i>Reserved</i>
1014	Microsoft Computer Dictionary, 3rd ed., 1997
1015	Hypertext Transfer Protocol – HTTP/1.1, RFC 2068, The Internet Engineering Task Force (IETF), January 1997
1016	HTML 4.0 Specification, The World Wide Web Consortium (W3C) Recommendation, December 18, 1997

Exhibit No.	Description
1017	HTTP State Management Mechanism, RFC 2109, The Internet Engineering Task Force (IETF), February 1997
1018	Abraham Silberschatz and Peter B. Galvin, Operating Systems Concepts, 4th Edition, 1994, Addison-Wesley Publishing
1019	Application of Internet Cache Protocol (ICP), version 2. RFC-2187. September 1997
1020	GNU C Manual Page documentation, 1993
1021	Persistent Client State HTTP Cookies, October 1996

MANDATORY NOTICES (37 C.F.R. §42.8(B))

A. Real Parties-In-Interest

The following are real parties-in-interest pursuant to 37 C.F.R. §42.8(b)(1):

- Capital One, National Association
- Hulu, LLC

B. Related Matters

Patent Owner, Implicit, LLC, asserted U.S. Patent No. 8,056,075 (“’075 patent”) in the following pending litigations:

- *Implicit LLC v. Capital One Financial Corp et al.*, (E.D. Tex.) Case No. 4:23-cv-00770-SDJ; and
- *Implicit, LLC v. Hulu, LLC*, (W.D. Tex) Case No. 6:23-cv-00513-ADA-DTG.

The ’075 patent was previously asserted in the following cases, all of which settled before any *Markman* order issued:

- *Implicit, LLC v. SonicWall Inc.*, (D. Del) Case No. 1-23-cv-01001;
- *Implicit, LLC v. PayPal Holdings, Inc. et al.*, (W.D. Tex.) Case No. 6-22-cv-00686;
- *Implicit, LLC v. Salesforce.com, LLC*, (W.D. Tex.) Case No. 6-22-cv-00364;
- *Implicit, LLC v. Juniper Networks, Inc.*, (E. D. Tex.) Case No. 2-19-cv-00037;
- *Implicit, LLC v. CA, Inc. d/b/a CA Technologies*, (E.D. Tex.) Case No. 2-19-cv-00038;

- *Implicit, LLC v. Fortinet, Inc.*, (E.D. Tex.) Case No. 2-19-cv-00039;
- *Implicit, LLC v. Imperva, Inc.*, (E.D. Tex.) Case No. 2-19-cv-00040;
- *Implicit, LLC v. Sandvine Corporation*, (E.D. Tex.) Case No. 2-19-cv-00041;
- *Implicit, LLC v. Sophos Ltd*, (E.D. Tex.) Case No. 2-19-cv-00042; and
- *Implicit Networks, Inc. v. Hewlett-Packard Company*, (N.D. Cal.) Case No. 3-12-cv-00685.

The '075 patent is directly related to the following issued US Patents and abandoned applications:

- U.S. Patent No. 6,324,685;
- U.S. Patent No. 6,976,248;
- U.S. Patent No. 7,774,740;
- U.S. Patent No. 9,848,031;
- U.S. Patent No. 8,856,779;
- U.S. Patent No. 9,325,740; and
- U.S. Patent Application No. 15,845,414 which was abandoned.

The '075 patent expired in December, 2023, for non-payment of maintenance fees.

C. Lead and Backup Counsel and Service (37 C.F.R. §42.8(b)(3)-(4))

Lead Counsel	
Jason A. Engel Reg. No. 51,654 K&L GATES LLP 70 W. Madison Street, Suite 3300 Chicago, IL 60602 jason.engel.ptab@klgates.com T: +1.312.807.4236 F: +1.312.827.8000	
Backup Counsel	
Erik J. Halverson Reg. No. 73,552 K&L Gates LLP 4 Embarcadero Center, Suite 1200 San Francisco, CA 94111 Erik.Halverson@klgates.com T: +1.415.882.8238 F: +1.415.882.8220	Cameron W. Westin Reg. No. 66,188 O'Melveny & Myers LLP 610 Newport Center Dr., 17th Floor Newport Beach, CA 92660 cwestin@omm.com T: +1.949.823.7987 F: +1.949.823.6994
Jared Lund Reg. No. 78,413 K&L GATES LLP 70 W. Madison Street, Suite 3300 Chicago, IL 60602 Jared.Lund@klgates.com T: +1.312.807.4287 F: +1.312.827.8000	Benjamin Haber Reg. No. 67,129 O'Melveny & Myers LLP 400 South Hope Street, 18th Floor Los Angeles, CA 90071 bhaber@omm.com T: +1.213.430.6000 F: +1.213.430.6407
William M. Fink Reg. No. 72,332 O'Melveny & Myers LLP 1625 Eye Street, NW Washington, DC 20006 tfink@omm.com T: +1.202.383.5300 F: +1.202.383.5414	

Petitioner consents to electronic service by email to the email addresses provided above.

D. Payment of Fees (37 C.F.R. §§42.15 and 42.103)

The required fees are submitted herewith in accordance with 37 C.F.R. §§ 42.103(a) and 42.15(a). If any additional fees are due during this proceeding, Petitioner authorizes the USPTO to charge any required fees to Deposit Account No. 02-1818. Any overpayment or refund of fees may also be deposited in this Deposit Account.

I. GROUNDS FOR STANDING (37 C.F.R. §42.104(A))

Petitioner certifies that the '075 patent is available for IPR and that Petitioner is not barred or estopped from requesting IPR.

II. STATEMENT OF PRECISE RELIEF REQUEST AND REASONS THEREFORE

Petitioner requests the institution of IPR and the cancellation of the Challenged Claims on the following Grounds:

Ground	Basis	Relied-On References	Claim(s)
1	§103 ¹	U.S. Pat. No. 6,260,078 to Fowlow et al. (“Fowlow”) (Ex.1009)	1-13
2	§103	Fowlow and Emin Gün Sirer & Brian Bershad, <u>Kimera: A System Architecture for Networked Computers</u> , Univ. of Wash., 1997 (“Kimera”) (Ex.1008)	1-13
3	§103	U.S. Pat. No. 6,347,398 to Parthasarathy et al. (“Parthasarathy”) (Ex.1007) and Kimera	1-7, 9-13
4	§103	Parthasarathy, Kimera, and U.S. Patent No. 5,933,811 to Angles et al. (“Angles”) (Ex.1010)	8

¹ See 89 FR 14449, 14449-14453 (“Updated Guidance for Making a Proper Determination of Obviousness,” dated Feb. 27, 2024)

The Petition is accompanied by the declaration of Dr. Erez Zadok, an expert in the field of the '075 patent, who opines each of the Challenged Claims are obvious over the prior art. Ex.1002, ¶¶1-44, 177-179, 439.

III. '075 PATENT OVERVIEW

A. Background of the Technology

The '075 patent discloses technology for managing requests sent from a client to a server for applets and applications from the server. Ex.1001, 1:16-18, 2:21-34. In particular, the server receives the request for an application or applet² from the client and then checks whether the request can be served from a local copy, and if not, will attempt to acquire the necessary resources from other sources on a network. *Id.*, Abstract, 3:53-4:17. The completed request (e.g., the applet) is then processed and transmitted to the client. *Id.* Figure 1 of the '075 patent, reproduced below, depicts an exemplary embodiment.

² The terms applet and application are used semi-interchangeably in the specification: “An applet (*e.g.* applets 1-3, 25*a*-25*c*) is any form of program instructions, whether in binary, source or intermediate format. In the case of this architecture, the applet code can either be a self-contained program, or it can be a code fragment associated with a larger application.” Ex.1001, 3:19-24.

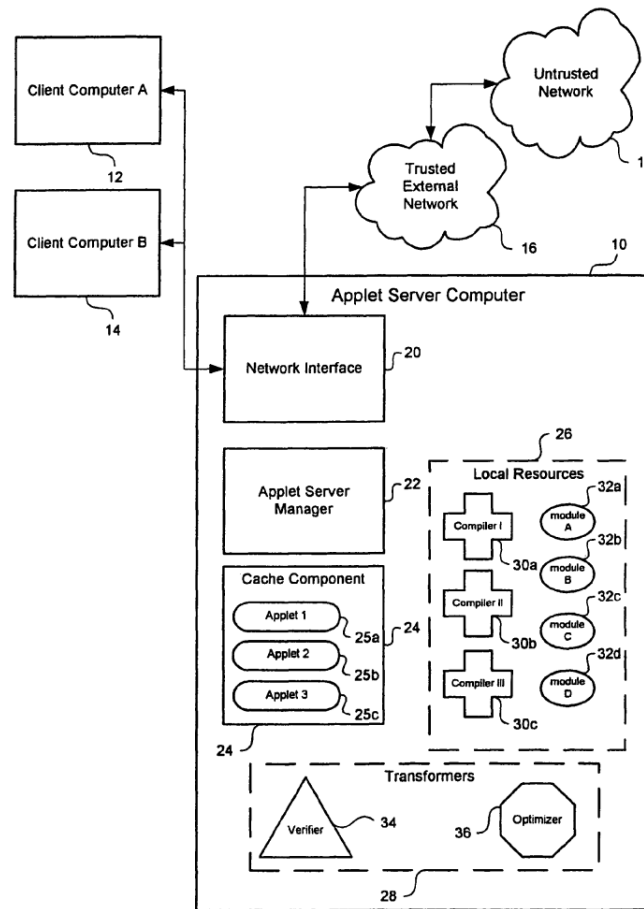


Fig. 1

Id., Fig. 1.

The described method of requesting software stored on a server, processing (e.g., verifying, optimizing, and compressing) the requested software at the server or server manager, and returning the requested software to a client device, was well-known before the earliest effective filing date of the '075 patent, as shown in the references described herein. Ex.1002, ¶¶45-144.

B. Prosecution History

The examiner rejected the sole claim of the as-filed application that led to the '075 patent as anticipated by U.S. Patent No. 5,923,885 ("Johnson"). Ex.1004, 63-67. In particular, the examiner found Johnson teaches "receiving on server computer a request from a client computer for an applet in a form selected from a plurality [of] forms," "compiling applet into selected form from a local resource comprising at least one source module and one compiler which acts on source module to produce selected form," and "transmitting applet in selected form to client computer." *Id.* The applicant cancelled this claim and filed replacement claims in response. *Id.*, 91.

The examiner rejected the new claims (application claims 2-16) and suggested edits during two interviews. *Id.*, 108 (summarizing 8/11/2011 interview), 115 (summarizing 9/19/2011 interview). The applicant and examiner reached agreement on allowable claims after the applicant cancelled the second set of then-pending claims, and added claims 17-29, claims which were entered via an examiner's amendment accompanying the notice of allowance. *Id.*, 115, 118-122.

In allowing the claims, the examiner confirmed much of the allowed subject matter, including networked client computers with managed software distribution of software stored on servers, was known in the art. Ex.1004, 121-122; Ex.1002, ¶¶145-146.

IV. CLAIM CONSTRUCTION

For purposes of this proceeding, Petitioner believes no terms need to be construed as the prior art utilizes substantially the same language such that, under any construction of the terms of the Challenged Claims, the claims are unpatentable. *Nidec Motor Corp. v. Zhongshan Broad Ocean Motor Co. Ltd.*, 868 F.3d 1013, 1017 (Fed. Cir. 2017). Petitioner respectfully reserves its right to respond to any unforeseeable claim constructions Patent Owner may advance.

V. ARGUMENTS

A. Level of Ordinary Skill in the Art

A POSITA with respect to this patent would have had a Bachelor's Degree in computer science, computer engineering, or equivalent degree, and approximately two years of experience working in the computer science or engineering field. A POSITA would have had knowledge of design considerations known in the industry and would have been familiar with then-existing products and solutions, including compilers, programming languages, networking, and web systems and applications. Additional work experience could make up for less education and vice-versa. Ex.1002, ¶¶147-149.

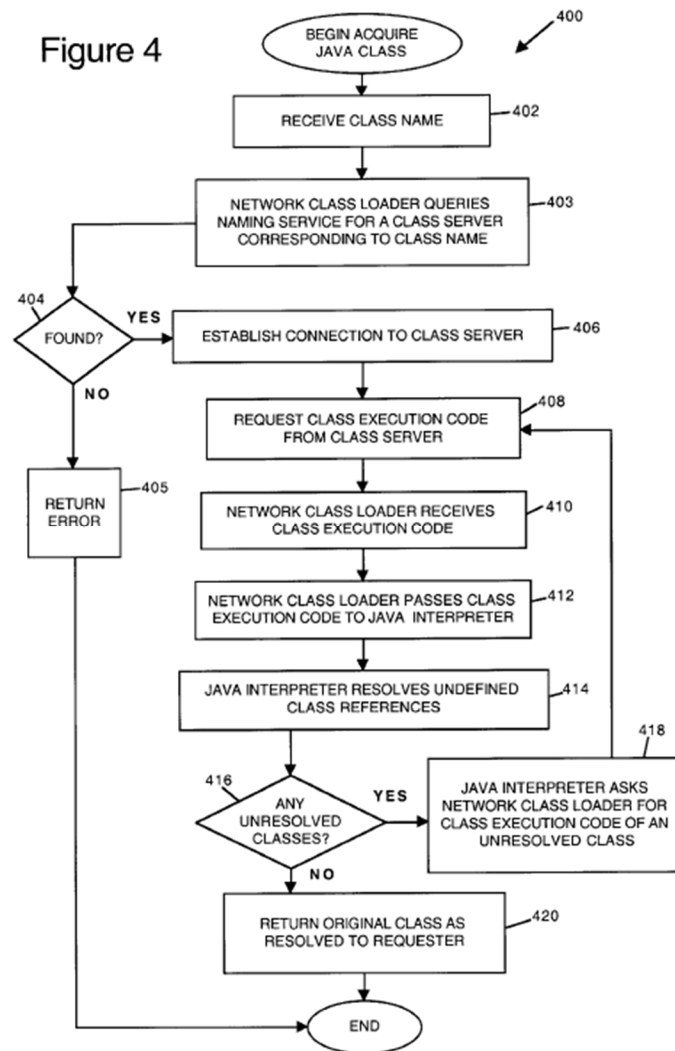
B. Prior Art

1. Fowlow (Ex.1009)

U.S. Patent No. 6,260,078 to Fowlow ("Fowlow") was filed on July 3, 1996,

issued on July 10, 2001, and is prior art under 35 U.S.C. § 102(e). Ex.1009, cover. Fowlow teaches a method of retrieving code from a server or other storage device for use on a client device. *Id.*, Abstract. The client sends requests for desired software classes to an “object request broker,” which communicates with networked applet servers to locate and return the desired code. *Id.*, 3:29-36. The applet servers respond to the request by retrieving and returning the “desired applet execution code to the client by way of the object request broker.” *Id.*, 3:36-42. A Java interpreter on the client determines whether the returned applet execution code contains any “unresolved classes,” *e.g.*, missing dependencies, and if so, sends another request to the object request broker to retrieve the unresolved classes from the server(s). *Id.*, Abstract, 13:33-67. The applet download process taught in Fowlow is depicted below:

Figure 4



Id., Fig. 4; *see also id.*, 11:57-12:11, 13:18-67; Ex.1002, ¶¶150-157.

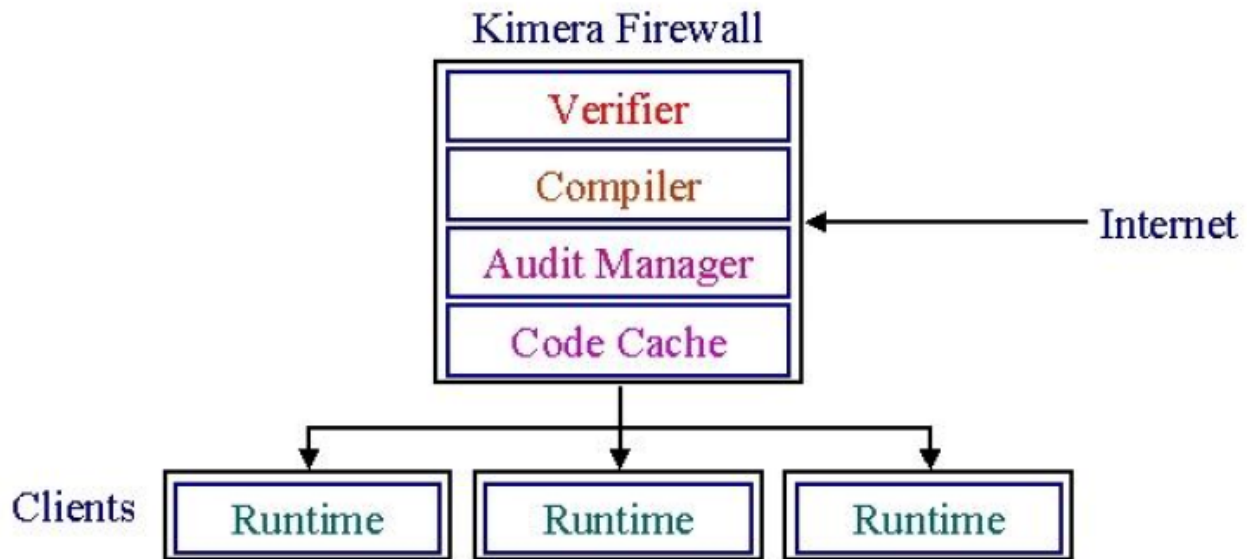
2. Kimera (Ex.1008)

“Kimera: A System Architecture for Networked Computers” was published at least as early as January 15, 1998. Ex.1008, 1; Ex.1012, 1-6, 25. Kimera would have been available to a POSITA as of this date and a POSITA would have been familiar with or able to access Kimera at this time. Ex.1002, ¶166. Moreover, as noted in Section V.G.1, the applicant submitted an IDS listing Kimera in the

applications for the parent and grandparent patents to the '075 patent. Accordingly, Kimera qualifies as prior art to the '075 patent under at least § 102(a).

Kimera teaches a Java Virtual Machine that allows mobile programs (*i.e.*, applets) “to be shipped from a web server to a client’s machine, where they execute locally and are integrated into the client’s environment.” *Id.* Implementations of Java may suffer several deficiencies, including that Java’s security model requires every client to perform its own safety check with a verifier, compiler, and interpreter located on the same physical machine as the Java Virtual Machine. *Id.* The trusted computing base can be costly because each client must include its own verifier, compiler, and interpreter. *Id.* Lastly, a simple upgrade to the Java Virtual Machine implementation at every end-node is difficult and expensive. *Id.*

Kimera discloses computer architecture in which the trusted computing base is located in centralized servers. *Id.*, 1. By locating verification, access control, auditing, and compilation on centralized servers, the same services can be performed more securely and efficiently for multiple clients. *Id.* For example, a security upgrade only affects the central firewall and would not require end-user cooperation. *Id.* In the architecture shown below, a verification service checks incoming code and conformance to interfaces before sending the code to the end-point.



Id.

Kimera discusses “ahead of time” compilation, which a POSITA would have understood to simply mean compilation of client-requested data received into the centralized firewall as part of the fulfillment of the client request, as compared to “just-in-time” compiling at the client itself. *Id.* (noting excessive communication overhead between client and server would dominate the benefits of centralization and that including a compiler on the centralized firewall would “deliver high performance” because this “machine [is] faster than any endpoint”); Ex.1002, ¶171. In sum, Kimera’s architecture provides mandatory safety enforcement, simple security management, and efficient memory and processing of Java clients. *Id.*; Ex.1002, ¶¶166-171.

3. Parthasarathy (Ex.1007)

U.S. Patent No. 6,347,398 to Parthasarathy et al. (“Parthasarathy”) was filed

on November 8, 1999, claims priority to U.S. Patent Application No. 08/764,040 filed December 12, 1996, and qualifies as prior art under 35 U.S.C. § 102(e). Ex.1007, cover. Parthasarathy teaches computers using network application programs to accept user commands and send requests when interfacing with the internet. *Id.*, 1:20-32. One type of network application program, network browsers, uses various protocols such as Hyper Text Transfer Protocol (“HTTP”) to access data on the World-Wide-Web (“Web”). *Id.*, 1:33-46. The Web contains documents created with Hyper Text Markup Language (“HTML”). *Id.*, 1:46-56.

Parthasarathy provides a method to “locate, download, verify, install, register, and display a computer software component from a computer network like the Internet.” *Id.*, Abstract, 1:11-16, 2:65-67. An example network environment for this software distribution is provided below.

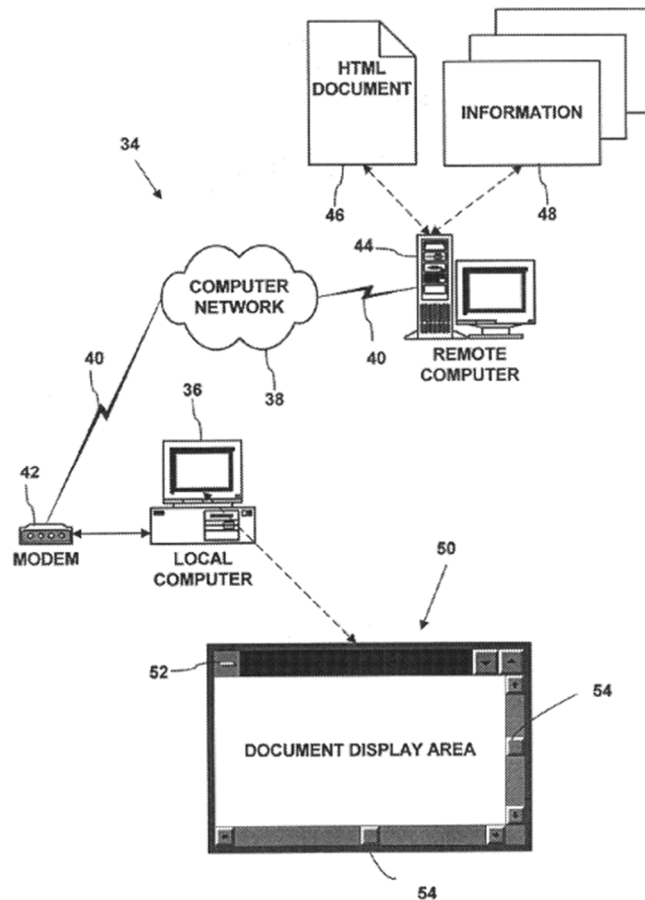


FIG. 2

Id., Fig. 2.

The network environment of Parthasarathy includes a local computer 36 that is connected to a remote computer 44 through the computer network 38 (*e.g.*, the Internet). *Id.*, 6:44-50; 7:3-8. To provide interactive or dynamic behavior when browsing documents, the local computer can request and obtain software components from the remote computer. *Id.*, 8:10-18. In one example, the network browser of the local computer requests to download a desired software component stored on the remote computer. *Id.*, 8:19-20. A software component download

module manages the request based on whether the “desired software component is already installed on the local computer 36” and whether “a more recent version [of the desired software component] is available.” *Id.*, 8:19-29; Ex.1002, ¶¶158-165.

4. Angles (Ex.1010)

U.S. Patent No. 5,933,811 to Angles et al. (“Angles”) was filed on August 20, 1996, issued August 3, 1999, and qualifies as prior art under 35 U.S.C. § 102(e). Ex.1010, cover. Angles is generally directed towards “a system and method for delivering customized electronic advertisements in an interactive communication system” such as the Internet. *Id.*, Abstract. In relevant part, Angles confirms “Persistent Client State Cookies”—which are “file[s] stored on the client computer which contains information such as user names and preferences”—were known as of August 1996, and cites a Netscape cookie specification, found at http://www.netscape.com/newsref/std/cookie_spec.html. *Id.*, 6:59-65; Ex.1021.

Angles explains the “small piece of information which a web server (via a CGI script) can store with a web browser...is useful for having the browser remember some specific information across several pages.” *Id.*, 11:5-12. The example given includes remembering items added to a shopping cart in a “virtual shopping mall” “so that the consumer can pay for all the items [] once he has finished shopping.” *Id.* Angles then describes how to set a cookie using a CGI script. *Id.*, 11:13-49; Ex.1002, ¶¶172-176.

C. Ground 1: Claims 1-13 are Unpatentable Under 35 U.S.C. §103(a) as Obvious over Fowlow

Claims 1-13 are obvious over Fowlow. Ex.1002, ¶¶180-181.

1. Independent Claim 1

- a. [1pre] A method for delivering one or more applets to one or more client computers, comprising, in no particular order, the steps of:*

To the extent the preamble is limiting, it is taught by Fowlow. Ex.1002, ¶182. Fowlow describes a client that “requests the code for the base class” and “requests code for any unresolved classes” from a server, which “retrieves the code by either reading a file from its own local file set, or if the code is not local, queries the naming service for another class server that has access to the code for the base class.” Ex.1009, Abstract, 10:15-18; 13:18-20. The code for these classes teaches the claimed applet, which is delivered to the claimed client computer. *Id.*, 3:32-42, 10:39-42, 13:30-45; Ex.1002, ¶¶183-186.

To the extent the preamble is limiting, a POSITA would have found it taught by Fowlow. Ex.1002, ¶187.

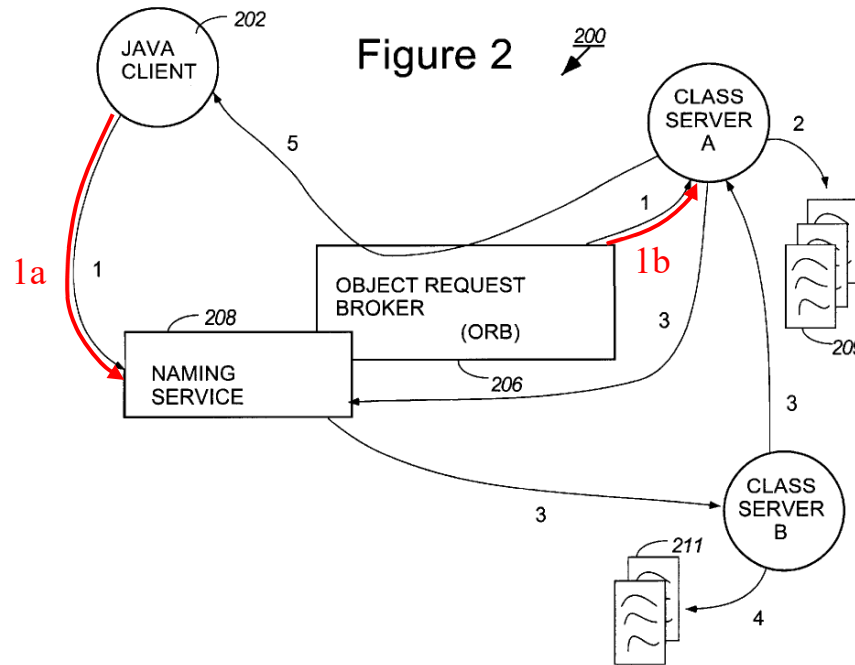
- b. [1a] configuring an applet server manager at a server computer to manage at least one request from the one or more client computers for the one or more applets,*

Fowlow teaches an intermediary Object Request Broker (“ORB”) that acts as an applet server manager, located at a server computer. Ex.1002, ¶¶188-189; *see also* Ex.1009, 5:34-36 (“In this example, ORB object implementation 14 is presented

as a server as shown by its location in the diagram.”), 10:21-23 (noting the ORB is located at the distributed object system). Fowlow explains that the ORB functions “to deliver a call from a client to a servant (target object)” and facilitates communication between a requesting client and “one or more class servers that provide access to the class files needed.” Ex.1009, 5:7-13, 10:15-26 (“An embodiment of the present invention provides *a mechanism by which a requesting client may acquire the classes it needs within a distributed object system* in order to run a particular applet.”) (emphasis added)³.

Fowlow’s ORB functions as “a single point of access to the distributed object system in order to find the class files that it needs.” Ex.1009, 10:23-28. All client code requests flow through the ORB, which in turn manages all requests to the connected servers from the client. *Id.*, 10:18-34, Fig. 2.

³ All emphasis herein is added unless stated otherwise.



Ex.1009, Fig. 2 (annotated); Ex.1002, ¶¶190-191.

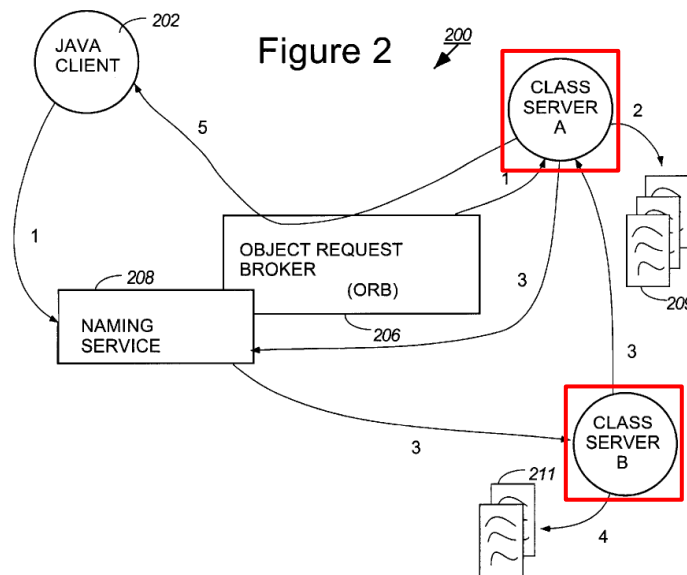
Fowlow's requests are for a specific Java applet or application made up of Java class files stored in one or more servers. Ex.1009, 11:21-30, 1:44-59. Fowlow defines "applet" as "a body of executable program code, which code is effective to implement a function or facility when used in conjunction with a supporting execution environment." *Id.*, 10:45-56 (further defining an applet as "a collection of byte codes representing classes that are executed by a remote interpreter, such as Java classes"). A POSITA would have understood that the ORB would be configured with software to process and manage incoming requests in order to perform the functions described by Fowlow. Ex.1002, ¶190; *see also* Ex.1009, 15:1-

13 (describing the configuration of computers for carrying out the teachings of Fowlow).

Fowlow teaches this limitation. Ex.1002, ¶192.

c. [1b] the applet server manager having access to one or more networks;

Fowlow's ORB communicates with one or more class servers, which collectively are network(s) that provide access to the class files as needed. Ex.1009, 10:18-31, 11:27-35. Figure 2 of Fowlow shows the ORB having access to Class Server A, which additionally has access to Class Server B—collectively a network. *Id.*, 11:59-12:3, Fig. 2.



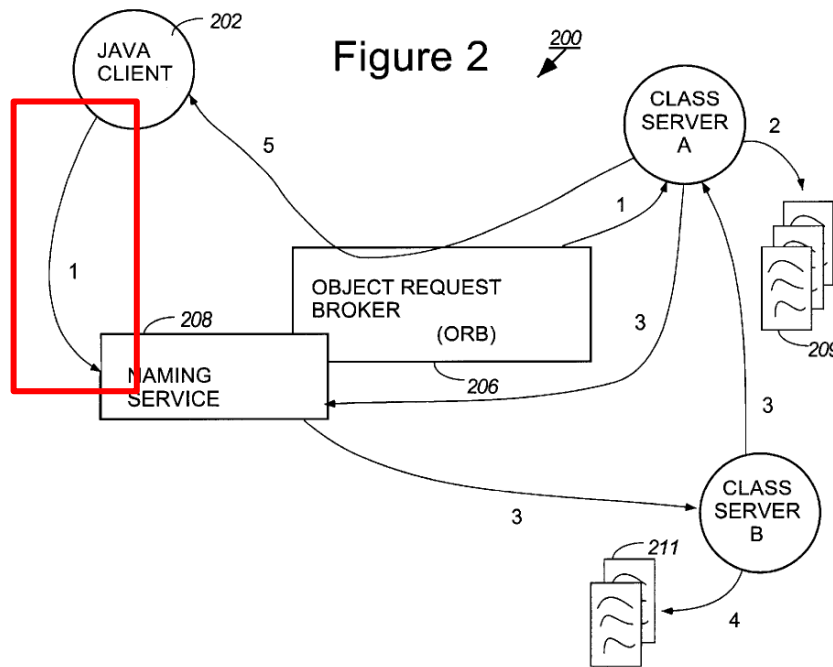
Id., Fig. 2 (annotated). A POSITA would have understood this representation of Fowlow suggests the use of many class servers, which collectively suggests access to one or more networks. Ex.1002, ¶193.

Fowlow also teaches that “a request from a client may pass from an ORB of one distributed object system to an ORB of another,” confirming that Fowlow’s Figure 2 would be replicated to reflect multiple ORBs and server networks. *Id.*, 10:34-36; Ex.1002, ¶194. A POSITA would have understood this disclosure as teaching the applet server manager (*i.e.*, ORB) has access to one or more distributed object systems (*i.e.*, networks) by passing requests to the other networks. Ex.1009, 10:34-36, 15:62-16:1 (describing connections to a “computer or telecommunications network”); Ex.1002, ¶194.

Fowlow, therefore, teaches this limitation. Ex.1002, ¶195.

d. [1c] receiving the at least one request at the applet server manager;

Fowlow describes a client request-server response interaction, which includes the applet server manager (*i.e.*, ORB) receiving the at least one request. For example, Fowlow teaches that a client requests an applet when it “queries the object request broker to determine if there is an applet server available within the system that may be used to obtain particular applet execution code.” Ex.1009, 3:27-42; *see also* Ex.1009, Abstract, 4:3-15 (describing the server receiving a “client request[] [for] additional applet execution code”), 10:18-34, Fig. 2; Ex.1002, ¶196.



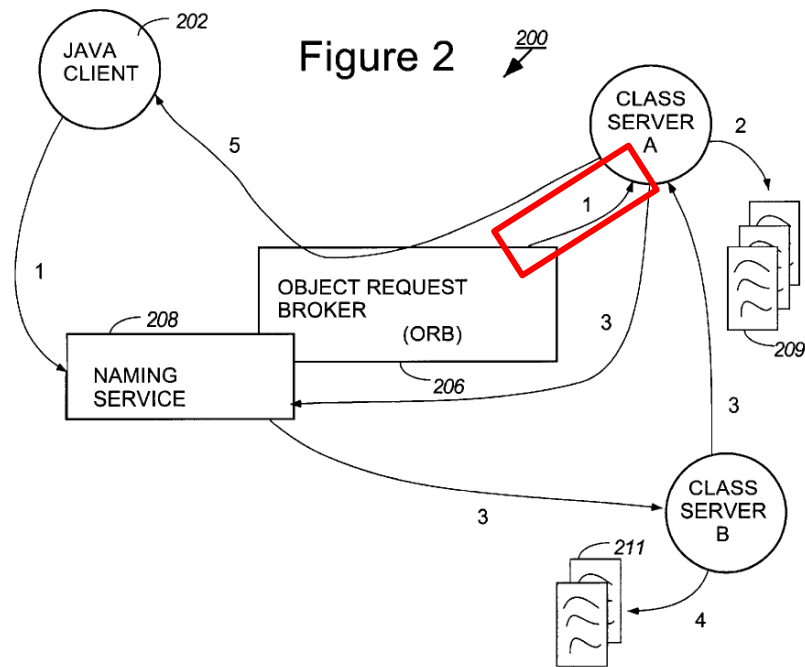
Ex.1009, Fig. 2 (annotated); Ex.1002, ¶¶196-197.

Fowlow, therefore, teaches this limitation. Ex.1002, ¶198.

- e. [Id] passing the at least one request from the applet server manager to at least one of the one or more networks;

As discussed above, Fowlow describes a request, from a client computer, that causes the ORB to *seek the requested software from another server in the distributed network* and return the files to the client once received from the second server or search recursively until the desired code is found. Ex.1009, Abstract, 3:27-4:15, 11:57-12:9; Ex.1002, ¶199. For example, Fowlow “provides a mechanism by which a requesting client may acquire the classes it needs within a distributed object system in order to run a particular applet. In one embodiment, this mechanism *uses an object request broker (ORB) in order to communicate with one or more class*

servers that provide access to the class files needed.” *Id.*, 10:15-28. A POSITA would have understood the ORB of Fowlow passes the request to the networks (*i.e.*, class servers), as evidenced in part by the continued use of “arrow 1” in Fig. 2. Ex.1002, ¶¶199-200.



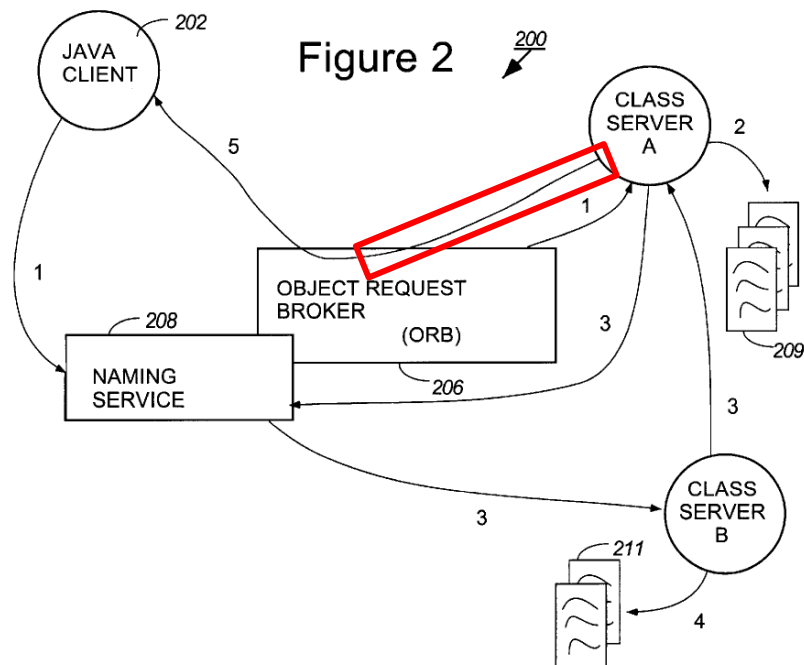
Ex.1009, Fig. 2 (annotated); Ex.1002, ¶200.

Additionally, Fowlow explicitly contemplates “a request from a client may pass from an ORB of one distributed object system to an ORB of another.” Ex.1009, 10:34-36. The first-contacted ORB may, therefore, pass the request from the applet server manager (*i.e.*, ORB) to one or more networks (*i.e.*, an ORB in another distributed object system or network). *Id.*; Ex.1002, ¶201.

Fowlow, therefore, teaches this limitation. Ex.1002, ¶202.

- f. [1e] receiving the one or more applets at the applet server manager from the at least one of the one or more networks;*

After class files that make up an applet have been located on the one or more servers, the file(s) are returned to the ORB from which the request was initially directed (*i.e.*, the applet server manager). Ex.1009, 11:63-12:9 (“Arrow (5) shows how this class code corresponding to the class name desired by the Java client is finally returned via the ORB 206 to the Java client 202.”); Fig. 2.



Ex.1009, Fig. 2 (annotated); Ex.1002, ¶203. A POSITA would have understood the class files received by the ORB would be an applet. Ex.1009, 10:50-53 (describing an applet as “a collection of byte codes representing classes that are executed by a remote interpreter, such as Java classes”); Ex.1002, ¶204.

Fowlow, therefore, teaches this limitation. Ex.1002, ¶205.

g. [If] processing the one or more applets at the applet server manager, wherein processing the one or more applets includes at least one of the following steps:

compressing the one or more applets before sending the one or more applets to the one or more client computers,

optimizing the one or more applets before sending the one or more applets to the one or more client computers, and

verifying the one or more applets before sending the one or more applets to the one or more client computers; and

Fowlow describes processing the applets at the ORB prior to sending them to the requesting client using “filter 40” on ORB 11, which “may perform a variety of tasks, such as **compression**, encryption, tracing, or debugging, that are applied to communications to and from an object.” Ex.1009, 6:5-8. Additionally, “[t]ransport layer 38 operates to marshal, unmarshal and physically transport information to and from a servant that typically does not share the same process as a client.” *Id.*, 6:9-11. A POSITA would have understood that each of these functions—compressing, encrypting, tracing, debugging, marshaling, and unmarshaling information—are examples of processing the applets at the applet server manager (ORB). Ex.1002, ¶¶206-211.

While filter 40 and transport layer 38 are detailed in relation to Figure 1a, Fowlow expressly confirms the “object request broker, or ORB, [of Figure 2] may be implemented as described above with reference to FIGS. 1a, 1b, and 1c.” *Id.*,

11:21-35. After being processed by the ORB, Fowlow explains that the applet is “finally *returned* via the ORB 206 *to the Java client 202*” as shown by Arrow 5 in Figure 2. Ex.1009, 11:63-12:9; Ex.1002, ¶221; *see also infra* Section V.C.1.h.

Performing the “compressing” step is sufficient to find Fowlow teaches this claim limitation. *Dealertrack, Inc. v. Huber*, 2008 WL 5792509, at *7 (C.D. Cal. Sept. 27, 2008), *aff’d in part, modified in part*, 674 F.3d 1315 (Fed. Cir. 2012); *cf. SIMO Holdings, Inc. v. H.K. uCloudlink Network Tech., Ltd.*, 983 F.3d 1367 (Fed. Cir. 2021). Consistent with the court in *Dealertrack*, “to construe this claim as ‘at least one of [compressing the one or more applets...], at least one of [optimizing the one or more applets...], and at least one of [verifying the one or more applets...],’ would not make sense” as one would not perform multiple iterations of compression on the applet. *Id.*; Ex.1001, 7:28-36; Ex.1002, ¶¶212-213.

To the extent Fowlow’s filter performing compression is insufficient to render obvious this limitation, Fowlow also teaches the “optimizing” and “verifying” steps. Ex.1009, 6:5-8; Ex.1002, ¶¶214-215. Fowlow renders obvious the claimed “verifying” step because a POSITA would have understood encryption and debugging to be a form of verification. Ex.1002, ¶¶218-219, 128-134. For example, a POSITA would have recognized that the filter would analyze the applet code to ensure it meets the requisite level of encryption and was error free, and therefore safe. *Id.* This is consistent with the teachings of the ’075 patent, which describes a

verifier as “a type of transformer that is able to analyze input code and determine areas that might not be safe. The verifier 48 can determine the level of safety.” Ex.1001, 5:34-36; Ex.1002, ¶219.

Fowlow renders obvious the claimed “optimizing” step because a POSITA would have understood that tracing and debugging suggest optimizing the applet received from the network at least because both functions have long been used to analyze programs and source code for the purpose of optimization. Ex.1002, ¶¶119, 134-135, 216-217. The '075 patent describes “optimizers” as “transformer programs [that] analyze code, making improvements to well known code fragments by substituting in optimized but equivalent code fragments.” Ex.1001, 6:6-9. In tracing and debugging the applet code, a POSITA would have understood this suggests making improvements to the code, for example, to eliminate excessive or incorrect class calls. Ex.1002, ¶217. A POSITA would have been motivated to include the filter on the ORB in Figure 2 to provide efficient, secure, and reliable applets to the client. Ex.1009, 6:5-8, 6:43-47; Ex.1002, ¶¶125, 134, 220.⁴

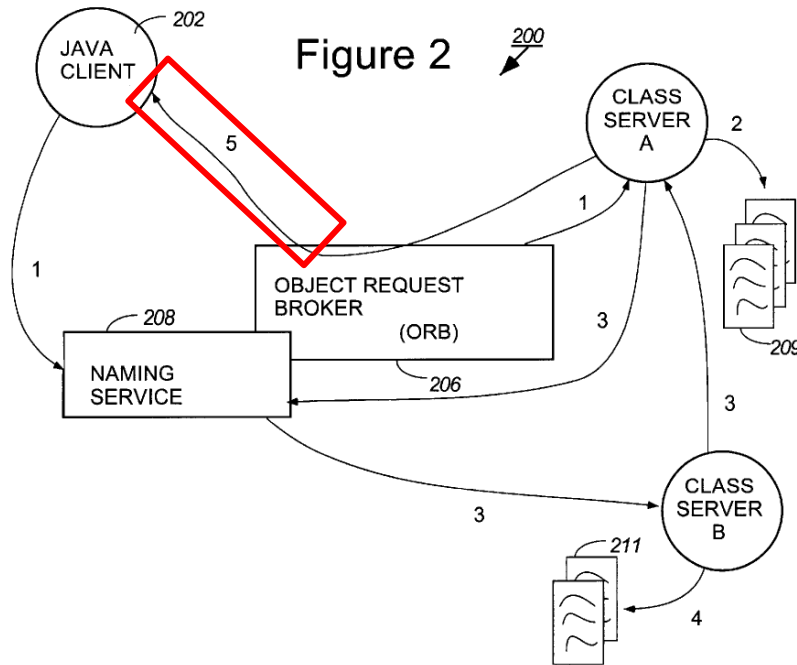
This “compression, encryption, tracing, or debugging,” of Fowlow suggests to a POSITA the claimed compressing (*e.g.*, filter 40’s compression), optimizing

⁴ Use of optimizers while processing applets was well-known in the prior art. Ex.1002, ¶¶114-119 (citing Ex.1006, 4, 13).

(e.g., tracing and debugging), and verifying (e.g., encryption and debugging) steps. Ex.1002, ¶¶214-219. Fowlow, therefore, teaches this limitation. Ex.1002, ¶¶211, 222.

h. [1g] sending the one or more applets from the applet server manager to the one or more client computers.

Fowlow discloses sending the requested applet from the ORB to the client. For example, Fowlow explains that once the applet server retrieves the desired applet execution code it is “able to return this portion of applet execution code to the client *by way of the object request broker.*” Ex.1009, 3:38-42. A POSITA would have understood this disclosure to mean that the ORB (not the applet server) sends the requested applet to the client computer, as required by this element. Similarly, as shown in Figure 2, Fowlow teaches that the “class code corresponding to the class name desired by the Java client is finally returned [to the client] via the ORB.” Ex.1009, 12:6-9.



Id., Fig. 2 (annotated); Ex.1002, ¶223. A POSITA would have understood the applet sent to the client could be the same applet that is processed by the ORB. Ex.1002, ¶224. Fowlow, therefore, teaches this limitation. Ex.1002, ¶225.

Fowlow discloses, or at least teaches, each limitation of claim 1, rendering this claim obvious. Ex.1002, ¶226.

2. Dependent Claims 2-13

- a. 2. The method of claim 1 wherein processing the one or more applets further includes discarding at least one of the one or more applets based on one or more policies.

Fowlow teaches performing security verification with the subcontract layer 26, which “may identify that the feature of security is to be used for a particular object.” Ex.1009, 5:59-6:1, 8:2-9. The security verification and action on that verification would lead to any applet that fails to meet the requisite level for the pre-

set qualities of service being rejected as part of the security verification. Ex.1009, 8:2-13; Ex.1002, ¶227. Fowlow's filter 40 also includes encryption functionality. Ex.1009, 6:5-8. A POSITA would understand that filter 40 would discard applets received from class servers, but that failed encryption verification. Ex.1002, ¶¶128-133 (discussing encryption, cryptographic hashing, digital signatures, and data integrity), 228; *see also* Ex.1014, 388-89 (describing using public key encryption, which allows "anyone possessing the corresponding public key" to decrypt a message), 131, 432 (defining "SHA," which "is used by the sender and the receiver of a message in computing and verifying a digital signature, for security purposes").

Fowlow renders obvious this claim. Ex.1002, ¶229.

- b. 3. The method of claim 1 wherein processing the one or more applets further includes transforming at least one of the one or more applets by modifying the data in the at least one of the one or more applets before sending the one or more applets to the one or more client computers.*

As described above in Section V.C.1.g, Fowlow teaches processing the applet, including compressing, verifying, and optimizing the applet, before sending the applet to the client computer. Ex.1009, 5:59-65, 6:5-11, 6:43-47, 10:50-56, 11:21-35; Ex.1001, 7:28-36; Ex.1002, ¶¶206-222. For example, filter 40 performs "a variety of tasks, such as compression, encryption, tracing, [and] debugging." Ex.1009, 6:5-8. A POSITA would have understood that each of these functions

would transform the applet by modifying data in the applet before it is sent to the client computer. Ex.1002, ¶¶230-232.

Fowlow renders obvious this claim. Ex.1002, ¶232.

- c. 4. The method of claim 1 wherein at least one of the one or more networks is a trusted network.*

“Secure Protocol 42 is a secure interoperability protocol that secures the internet inter-ORB protocol and helps to transmit information through transport layer 38 in a secure fashion....The internet inter-ORB protocol is a protocol that typically communicates between processes on different machines.” Ex.1009, 6:43-49. “[I]n some cases, the internet inter-ORB protocol may communicate between processes on the same machine. The security server 54 is a security administration server that secures the services that are used between processes on different computers.” Ex.1009, 6:49-53.

A POSITA would have understood the difference between inter-ORB “processes on the same machine” and “processes on different machines” as trusted and untrusted networks, respectively, given the enhanced security required in the second scenario. Ex.1002, ¶¶233-235. This is similar to how the ’075 Patent describes distinguishing trusted and untrusted networks. Ex.1001, 6:27-33.

Fowlow renders obvious this claim. Ex.1002, ¶236.

- d. 5. The method of claim 1 wherein at least one of the one or more networks is an untrusted network.*

As discussed above, Fowlow teaches internet inter-ORB “processes on the same machine” and “processes on different machines,” which a POSITA would have understood as trusted and untrusted networks. Ex.1009, 6:43-53; Ex.1002, ¶¶237, 233-235. For untrusted networks, security server 54 is additionally utilized to “secure[] the services that are used between processes on different computers.” Ex.1009, 6:51-53; Ex.1002, ¶238.

Fowlow renders obvious this claim. Ex.1002, ¶239.

- e. 6. The method of claim 1 wherein the at least one request specifies the name of at least one of the one or more applets using a Uniform Resource Locator.*

Fowlow discloses that it was known that “the location of [the] Java classes [that comprise an applet and that are located] within the computer network are identified using a Universal Resource Locator (URL) address.” Ex.1009, 2:13-32. Further, the “HTML page data that a user may acquire through [a browser] contains references to applet classes that may be used to execute small programs in parts of the page” and the “code for the applet...is found by reference to a specific URL address that identifies a particular computer.” *Id.*, 2:37-54. A POSITA would have understood the request would specify the “hard-wired” file name of the requested applet using a URL. *Id.*; Ex.1002, ¶240.

Fowlow renders obvious this claim. Ex.1002, ¶241.

- f. 7. The method of claim 1 wherein the at least one request specifies at least one of the one or more applets using an HTTP header.*

Fowlow discloses that “a browser [client] typically uses the hypertext transfer or ‘http’ protocol” to request data, such as Java classes, stored on servers. Ex.1009, 2:22-25, 7:11-23, 10:49-50; Ex.1002, ¶242. Because, as Fowlow teaches, clients “typically use[] the” HTTP protocol, a POSITA would have understood that it would have been obvious for the requested applet to be specified in the HTTP header. Ex.1009, 2:22-25, Ex.1002, ¶¶63-65, 242; Ex.1014, 305.

Fowlow renders obvious this claim. Ex.1002, ¶243.

- g. 8. The method of claim 1 wherein the at least one request specifies at least one of the one or more applets using a cookie.*

Fowlow teaches “the form of the request [for an application or applet] will depend to a large extent upon the nature of the programming language used to create the client.” Ex.1009, 7:11-23, 10:28-34. Fowlow also teaches an implementation repository on the ORB that “is used to store information relating to object servers” including “information such as...any arguments to the [server] program, and any environment variables to pass to the program, etc.” *Id.*, 6:61-67. A POSITA would have understood the arguments and environment variables stored in the

implementation repository of the ORB would be a form of cookie. Ex.1002, ¶¶244-246, 66-71, Ex.1017, 1, Ex.1014, 119.

Fowlow renders obvious this claim. Ex.1002, ¶247.

- h. 9. The method of claim 1 wherein processing the one or more applets includes producing at least one of the one or more applets in the form of a web page.*

Fowlow describes a method of locating and downloading applets for use on a client machine. Ex.1009, Abstract, 10:15-65. These applets, according to Fowlow, “may refer to a wide variety of types of execution code.” *Id.*, 10:45-56. One such example of a type of execution code is a set of Java classes that “can be embedded in another application such as an HTML document in order to provide interactive, executable content on a **Web page**.” *Id.*, 10:56-62. A POSITA would have understood that the requested applet could be an HTML document, *i.e.*, a web page, that may additionally include embedded applets. *Id.*, 2:34-40; Ex.1002, ¶¶248, 72-77. If the request was directed to a web page, a POSITA would have understood that in processing the applet in response to the request, the modified ORB would include the additional step of “producing at least one of the one or more applets in the form of a web page.” Ex.1002, ¶248.

Fowlow renders obvious this claim. Ex.1002, ¶249.

- i. *10. The method of claim 1 wherein processing the one or more applets includes producing at least one of the one or more applets in the form of a portion of a web page.*

Fowlow discloses “the HTML page data that a user may acquire through Netscape contains references to *applet classes* that may be used to execute small programs *in parts of the page*. Thus, Netscape would be directed to locate and download the code for the applet to run in the frame that the page defines.” Ex.1009, 2:37-47, 4:3-15. Fowlow also states “applet may mean in particular a small Java program that can be embedded in another application such as an HTML document in order to provide interactive, executable content on a Web page....The presentation of a particular object may be used to run an executable program in a small window within a larger window.” *Id.*, 10:59-62. A POSITA would have understood “a portion of a web page” to include a small Java applet program embedded in part of a web page. Ex.1002, ¶¶250, 72-80.

Fowlow renders obvious this claim. Ex.1002, ¶251.

- j. *11. The method of claim 1 wherein sending the one or more applets includes sending at least one of the one or more applets to the one or more client computers in an HTTP response.*

As discussed in relation to claim 7, Fowlow discloses HTTP requests. Ex.1009, 2:22-25, 7:11-23, 10:49-50; Ex.1002, ¶¶242-243. A POSITA would have understood that the http exchange, and in particular, “[t]he http server...send[ing]

classes...back to the requesting browser for execution” involves sending the requested applet to the client in an HTTP response. *Id.*; Ex.1002, ¶¶252, 55, 62.

Fowlow renders obvious this claim. Ex.1002, ¶253.

- k. 12. The method of claim 1 wherein the at least one request is processed prior to passing the at least one request from the applet server manager to at least one of the one or more networks.*

Fowlow’s ORB performs a number of processing steps to assist with locating applet software classes before passing the request to the network: (1) ***locating software on a server within the distributed object network***, (2) searching recursively until the desired code is found within the network, or (3) if the desired code is not found within the network, passing requests to ORBs of other networks. Ex.1009, Abstract, 3:27-4:15, 10:34-36, 11:57-12:9. A POSITA would have understood this local looking functionality to be processing the request at the ORB before passing the request to the network. Ex.1002, ¶¶254-255; *see also* Ex.1002, ¶¶256-257 (noting the ORB also performs compressing, encrypting, tracing, and debugging communications ***to*** servers (Ex.1009, 6:5-8); converting the request into appropriate protocols for a particular domain by the transport layer 38 (Ex.1009, 8:27-31); and pairing the client request made in the client’s native language with a stub function 25 found using a subcontract layer 36 (Ex.1009, 7:39-65)).

Fowlow renders obvious this claim. Ex.1002, ¶258.

- l. 13. The method of claim 1 wherein processing the one or more applets further includes performing security verification based on one or more policies.*

Fowlow teaches performing security verification with the subcontract layer 26, which “may identify that the feature of security is to be used for a particular object.” Ex.1009, 5:59-6:1. “A subcontract implemented by subcontract layer 36 determines a specific quality of service for use by an object....A quality of service is a set of service properties. Among possible service properties that are selectable are qualities relating to server activation, *security*, transactions, filterability, and clean shut-down.” *Id.*, 8:2-9. A POSITA would have understood this subcontract layer performs security verification based on one or more policies, such as the predetermined quality of service. Ex.1002, ¶¶259-260 (citing Ex.1009, 8:2-13).

Fowlow renders obvious this claim. Ex.1002, ¶261.

D. Ground 2: Claims 1-13 are Unpatentable Under 35 U.S.C. §103(a) as Obvious over Fowlow in view of Kimera

As detailed in Sections V.C.1 and V.C.2, Fowlow renders claims 1-13 obvious. As discussed in Section V.C.1.g, the *Dealertrack* court held explicit disclosure is not required for each step in a list where the list is grammatically comparable to the '075 patent claims. *See Dealertrack*, 2008 WL 5792509 at *7. For example, for the reasons detailed below, a POSITA would have been motivated to incorporate the more discretely disclosed verification and optimization

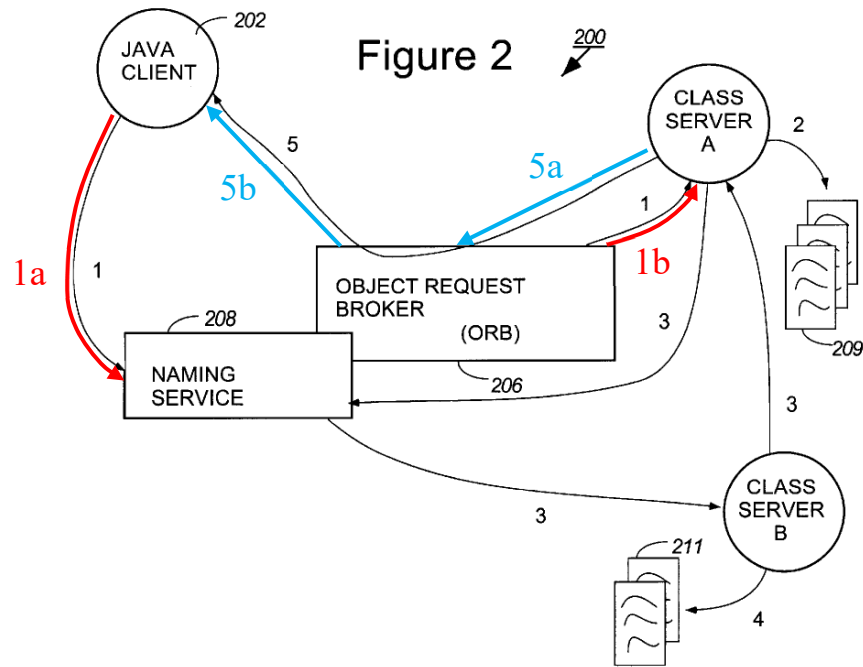
functionality of Kimera's centralized server into Fowlow's ORB, further confirming claims 1-13 of the '075 patent would have been obvious to a POSITA. Ex.1002, ¶262. To the extent Ground 1 does not render obvious the challenged claims, the combination of Fowlow and Kimera, which expressly addresses all three elements of the listed processing, does.

1. Motivation to Combine

Fowlow acquires software applications and classes via requests to an ORB, which locates the requested software class files in a server on the network and returns the files to the client through an object request broker. Ex.1009, 3:26-42. Kimera, relatedly, describes acquiring software applications from the Internet via a "centralized server" that includes a verification service, compiler, audit manager, and code cache. Ex.1008, 1-2. Fowlow and Kimera are both directed to acquiring software capable of running on client devices from a remote network and are analogous art to the '075 patent and each other. Ex.1002, ¶¶263-266.

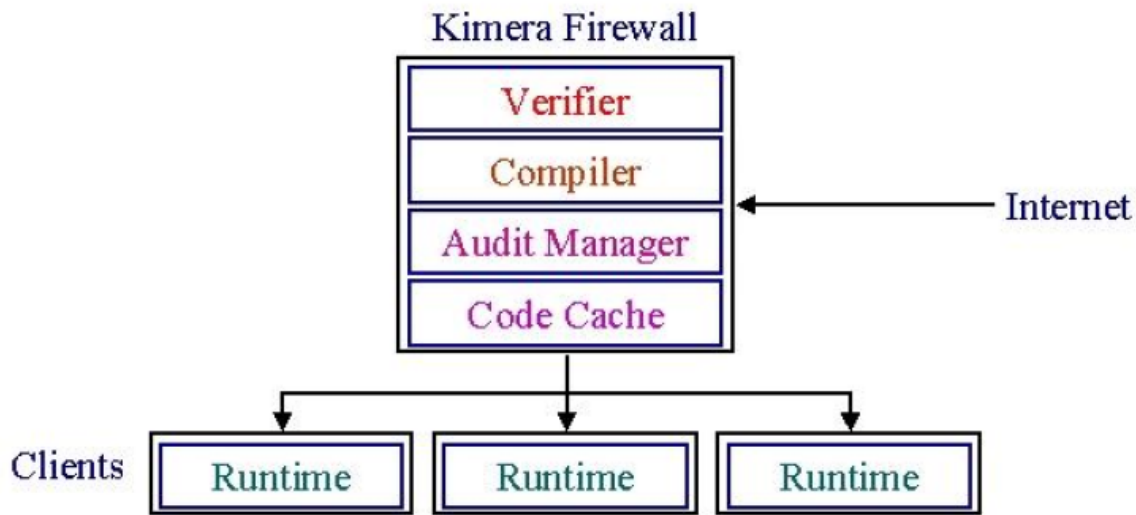
Fowlow teaches a loop involving the client device, an ORB, and one or more servers, that runs recursively to locate and download a base class of a certain software, verify whether code for all dependencies exists on the client, and if not, again attempt to locate and download any additional software to resolve any missing dependencies. Ex.1009, Abstract, 3:31-42, 3:58-4:15, 13:18-67. This loop can be seen in Fowlow Fig. 2, reproduced below, with the initial request flowing from client

to server via the ORB (arrow 1a-1b) and the results returning to the client (arrow 5a-5b). If the client finds missing dependencies, the entire loop is completed again to locate and download the needed classes. Ex.1009, Abstract, 4:3-15, 13:18-67.



Ex.1009, Fig. 2 (annotated); Ex.1002, ¶¶267-268.

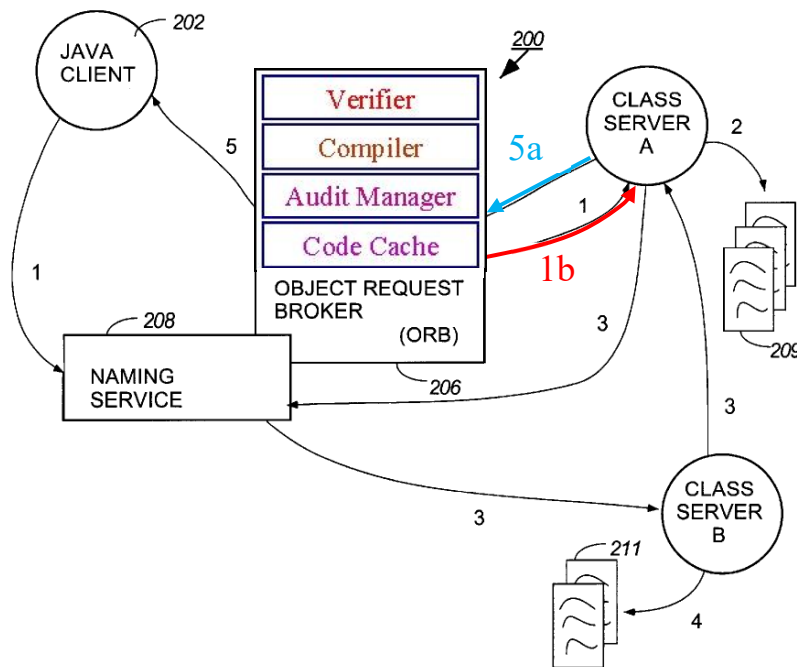
Kimera teaches placing the verifier and compiler onto a centralized server, described in Kimera as the Firewall. This Firewall processes (*i.e.*, compiles, verifies, audits, and caches) data from the Internet and pushes the processed data to clients.



Ex.1008, 1; Ex.1002, ¶269.

Kimera highlights (and addresses) a problem with the class resolution process of Fowlow: significant communication overhead between the client and server, which has the byproduct of making client devices more complex and expensive. Ex.1008, 2. This shortcoming is addressed by moving compilation and verification services to the centralized point of contact between client and server (*i.e.*, Fowlow's ORB) as taught by Kimera and its Firewall. A POSITA would have understood that the Fowlow-Kimera system would receive a base class at the modified, centralized point (*e.g.*, the ORB of Fowlow), use the verifier of the centralized point to discover and resolve any unresolved dependencies by using the modified centralized point to verify the code (as taught by Kimera) and submit additional requests to the class servers for missing classes as needed, before finally compiling the complete applet and sending it to the client. Ex.1002, ¶¶269-271.

The modified loop can be seen as consisting of only arrows 1b and 5a in the modified diagram below.



Ex.1009, Fig. 2 (annotated); Ex.1008, 1; Ex.1002, ¶270. This eliminates extra communication steps between the client and server and minimizes the need for compilers and verifiers on client devices, which a POSITA would have understood would make the overall system faster, cheaper, and more secure. *Id.*, ¶¶271-272 (citing Ex.1008, 1-2).

Locating a verifier in the modified ORB further advances the goals outlined in Fowlow, namely, that the client “knows a base class this it wishes to load, but is mercifully unaware of all the machinations behind the scenes required to find other classes used or needed by this base class.” Ex.1009, 10:26-34. Once the complete

applet has been compiled at the modified ORB, the result would be cached in the “Code Cache” of the modified ORB to “further reduce the latency of binary translation” in response to future requests for the same code. Ex.1008, 2; Ex.1002, ¶273. This desire to reduce latency further would contribute to the optimization of the cached code.

The combination of Fowlow and Kimera would incorporate the functionality of the Kimera Firewall into Fowlow’s Object Request Broker 206, which would continue to facilitate communication between clients and applet servers, and furthermore improve the compilation, verification, and dependency resolution processes. *See* Ex.1009, 5:49-51, 10:18-31, 11:25-33; Ex.1002, ¶275. In the proposed combination, each of the respective disclosures of Fowlow and Kimera function in the same manner as in their respective sources and are therefore simply a rearranging of known components in accordance with known methods to provide predictable results. *KSR*, 550 U.S. at 416; *Smith & Nephew*, 721 F.3d at 1375. Furthermore, given the similarities between Fowlow and Kimera, particular regarding a centralized surrogate between client devices and the Internet, a POSITA would have been able to combine these references with a reasonable expectation of success, avoiding overly-burdensome experimentation. Ex.1002, ¶275.

A POSITA would have been motivated to combine Fowlow and Kimera. *Id.*, ¶¶263-275.

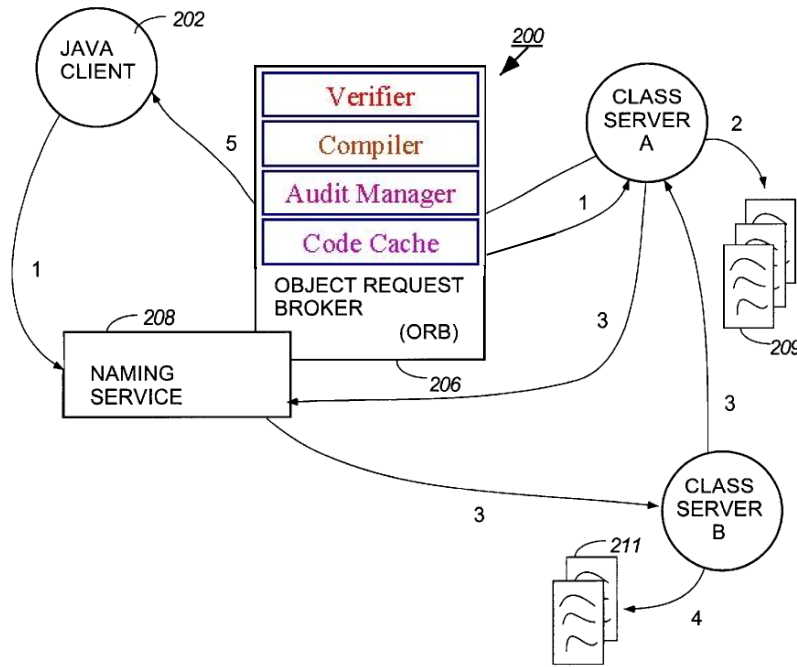
2. Independent Claim 1

a. [1pre]

To the extent the preamble is limiting, Fowlow in view of Kimera teaches it as discussed in Section V.C.1.a. Ex.1009, Abstract, 10:15-18; 13:18-20, 13:30-45, 10:39-42; Ex.1002, ¶¶182-187, 276.

b. [1a]

Fowlow in view of Kimera teaches this limitation as discussed in Section V.C.1.b. Ex.1009, 1:44-59, 5:7-15, 5:34-36, 10:15-34, 10:45-56, 11:21-30, Fig. 2; Ex.1002, ¶¶188-192, 277. As discussed in Section V.D.1, the functionality of Kimera's Firewall/server would be incorporated into Fowlow's ORB to improve system latency through (1) simplifying the dependency resolution process and (2) caching and compiling code at a central location. Ex.1009, 5:49-51, 10:18-31, 11:25-33; Ex.1008, 2; Ex.1002, ¶¶277-278. Fowlow's ORB, modified to include the functionality of the Kimera Firewall, would continue to serve as the claimed applet server manager, facilitate communication between clients and servers while making the entire system overall faster, cheaper, and more secure. Ex.1002, ¶278.



Ex.1009, Fig. 2 (annotated); Ex.1008, 1; Ex.1002, ¶278. A POSITA would have understood the modified ORB would continue to be configured to manage requests from client computers for applets. Ex.1002, ¶278. Fowlow-Kimera teaches this limitation. Ex.1002, ¶279.

c. [1b]

Fowlow in view of Kimera teaches this limitation as discussed in Section V.C.1.c. Ex.1009, 10:18-31, 10:34-36, 11:27-35, 11:59-12:3, Fig. 2; Ex.1002, ¶¶193-195, 280.

d. [1c]

Fowlow in view of Kimera teaches this limitation as discussed in Section V.C.1.d. Ex.1009, Abstract, 3:27-4:15, 10:18-34, 11:57-12:9, Fig. 2; Ex.1002, ¶¶196-198, 281.

e. [1d]

Fowlow in view of Kimera teaches this limitation as discussed in Section V.C.1.e. Ex.1009, Abstract, 3:27-42, 10:15-28, 10:34-36, Fig. 2; Ex.1002, ¶¶199-202, 282.

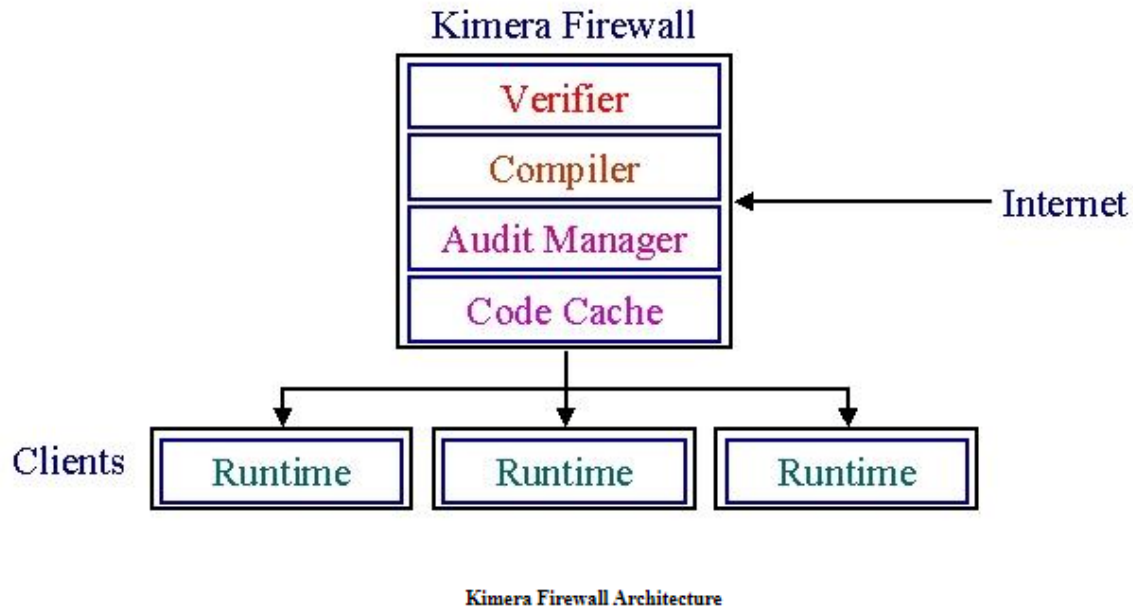
f. [1e]

Fowlow in view of Kimera teaches this limitation as discussed in Section V.C.1.f. Ex.1009, 10:50-53, 11:63-12:9, Fig. 2; Ex.1002, ¶¶203-205, 283. These teachings are also consistent with Kimera's depiction of data, such as Java applets, being received into the Firewall/server from the Internet. Ex.1008, 1-2; Ex.1002, ¶283.

g. [1f]

Fowlow in view of Kimera teaches this limitation as discussed in Section V.C.1.g. Ex.1009, 5:59-65, 6:5-11, 6:43-47, 10:50-56, 11:21-35, Figs. 1a, 2; Ex.1001, 7:28-36; Ex.1002, ¶¶206-222, 284, 292. Adding the functionality of Kimera's firewall to Fowlow's ORB enhances the ORB's processing of the applets by bringing additional processing functionality into the central ORB server. For example, Kimera discloses a "way of structuring Java systems, in which verification and compilation are performed in centralized servers." Ex.1008, 1-2. Kimera describes numerous benefits associated with this centralized arrangement, and in

particular notes that “[c]ompilation from Java bytecode into native machine code is another service that is best performed at a central location.” *Id.*, 2.



Id., 1; Ex.1002, ¶285.

The Kimera architecture includes several components specifically designed to process applets received from the Internet. For example, Kimera includes a compiler that compiles “Java bytecode into native machine code” and a code cache which “cach[es] the results of recent compilations in the central compilation server.” Ex.1008, 2. A POSITA would have understood the *recent* compilation results would be the results of a compilation made in response to recent client requests. Ex.1002, ¶286. Kimera confirms that the centralized location would perform the three claimed processing steps. Ex.1002, ¶¶287-291, 293.

Kimera's "approach to Java system structure is to break the trusted computing base into small, secure components and locate them in security firewalls." Ex.1008, 1. To consolidate resources and reduce client requirements, Kimera discusses consolidating processing functionality into a centralized server, which allows clients to exist as a runtime environment. *Id.*, 2. To avoid excessive communication overhead between client and server, a POSITA would have found it obvious to include compression on all applet files sent to the client, which would reduce the size of communications flowing between client and server. *Id.*; Ex.1002, ¶¶294-295.

Kimera teaches a "verification service" within the modified ORB that "performs checking of incoming code for typesafety and conformance to interfaces." Ex.1008, 2. Further, Kimera details how "tamper-proof signatures, using a cryptographically strong hash function, such as MD5, can be utilized to ensure the safety of the code after it leaves the verifier." *Id.* A POSITA would, therefore, have understood Fowlow-Kimera teaches verifying the applets before sending the applets to a client computer. Ex.1002, ¶300.

Kimera further explains the benefits of the centralized architecture utilized in both Fowlow and Kimera, which is that it "enables the same services [*e.g.*, **verification**, access control, auditing, and compilation services] to be performed **faster**, cheaper, and without introducing security vulnerabilities." Ex.1008, 1. Kimera continues, describing additional benefits include how the centralized server

“makes a sizable reduction in computation and memory requirements of the endpoints” and that “Java enabled network computers [clients] could [] be constructed from cheaper components.” *Id.*, 2.

A POSITA would have understood that this is a form of optimizing the applets from the Java bytecode before sending the one or more applets to the one or more client computers because it results in faster services performed on the most inexpensive clients possible. Ex.1002, ¶¶296-298, 114-119; Ex.1014, 343 (defining “optimization” as “1. In programming, the process of producing more efficient (smaller or faster) programs through selection and design of data structures, algorithms, and instruction sequences. 2. The process of a compiler or assembler in producing efficient executable code.”), 255, 268. Additionally, compiling and caching applets on the centralized server would reduce communication overhead, optimizing the applet retrieval process before applets are sent to client computers. *Id.*, ¶299.

A POSITA would have been motivated to make the combination of Fowlow and Kimera as discussed above in section V.D.1. Ex.1002, ¶288. Kimera explains there are several disadvantages with relying on client computers for compiling applications, including security implications and “very high” client system requirements, making client devices more expensive. Ex.1008, 1. A POSITA would have been motivated to modify or supplement Fowlow’s ORB—which acts as a

point of communication between the client devices and the broader system, locating files and relaying those files from servers to the clients, which are then required to confirm any unresolved dependencies and send additional requests for more files—with the more comprehensive and efficient Kimera Firewall, resulting in a faster, cheaper, and more secure distributed object system. Ex.1002, ¶¶263-275; *see also* Ex.1009, 10:15-36, 11:57-12:11, 12:46-60, Fig. 4 (showing a loop to resolve “unresolved classes”).

A POSITA would have been motivated to combine Fowlow and Kimera, which together, teach this limitation. Ex.1002, ¶301.

h. [lg]

Fowlow in view of Kimera teaches this limitation as discussed in Section V.C.1.h. Ex.1009, 12:6-9, Fig. 2; Ex.1002, ¶¶223-226, 302.

This claim is rendered obvious by Fowlow-Kimera. Ex.1002, ¶303.

3. Dependent Claims 2-13

a. Claim 2.

Fowlow in view of Kimera renders this claim obvious as discussed in Section V.C.2.a. Ex.1009, 5:59-6:1, 8:2-13; Ex.1002, ¶¶227-229, 304. Kimera explains the “verification service performs checking of incoming code for typesafety and conformance to interfaces.” Ex.1008, 2. After code leaves the verifier, “[t]amper-proof signatures, using a cryptographically strong hash function, such as MD5, can

be utilized to ensure the safety of the code....” *Id.* A POSITA would have understood that while performing checking of incoming code for typesafety and conformance, the verification station of the applet server manager would discard applets that failed to satisfy the typesafety and conformance policies. Ex.1002, ¶305. Fowlow-Kimera renders obvious this claim. Ex.1002, ¶306.

b. Claim 3.

Fowlow in view of Kimera renders this claim obvious as discussed in Section V.C.2.b. Ex.1009, 5:59-65, 6:5-11, 6:43-47, 10:50-56, 11:21-35; Ex.1001, 7:28-36; Ex.1002, ¶¶230-232, 307. As discussed in Section V.D.2.g, Fowlow-Kimera teaches processing the applet, including compressing, verifying, and optimizing the applet, before sending the applet to the client computer. Ex.1009, 5:59-65, 6:5-11, 6:43-47, 10:50-56, 11:21-35; Ex.1008, 1-2, Ex.1002, ¶¶206-222, 308. For example, Kimera includes a compiler that compiles “Java bytecode into native machine code...at a central location.” Ex.1008, 2. This central location would be the applet server manager, which both compiles code as well as verifies and applies tamper proof signatures to code passed from the applet server manager to the client. *Id.*; Ex.1002, ¶309. A POSITA would have understood that both compiling code from “Java bytecode into native machine code” and applying cryptographic signatures would transform the applets by modifying the data in the applet before it is sent to

the client computer. Ex.1002, ¶¶310, 130-132. Fowlow-Kimera renders obvious this claim. Ex.1002, ¶311.

c. Claim 4.

Fowlow in view of Kimera renders this claim obvious as discussed in Section V.C.2.c. Ex.1009, 10:15-28, 10:34-36; Ex.1002, ¶¶233-236, 312. Additionally, Kimera discloses a Firewall, which a POSITA would understand is designed to protect a trusted, internal network (*e.g.*, an intranet) from an untrusted, external network (*e.g.*, the Internet). Ex.1008, 1; Ex.1002, ¶313. Thus, Fowlow-Kimera renders obvious this claim. Ex.1002, ¶314.

d. Claim 5.

Fowlow in view of Kimera renders this claim obvious as discussed in Section V.C.2.d. Ex.1009, 10:34-36; Ex.1002, ¶¶237-239, 315. Kimera discloses a Firewall, which separates trusted and untrusted networks. Ex.1008, 1; Ex.1002, ¶313. Kimera's verifier will apply "[t]amper-proof signatures, using a cryptographically strong hash function, such as MD5,...to ensure the safety of the code...." Ex.1008, 2. A POSITA would have understood that tamper-proof signatures would be applied to code that had been received from an untrusted network and subsequently verified to be safe. Ex.1002, ¶316. Fowlow-Kimera renders obvious this claim. Ex.1002, ¶317.

e. Claim 6.

Fowlow in view of Kimera renders this claim obvious as discussed in Section V.C.2.e. Ex.1009, 2:13-32, 2:37-54; Ex.1002, ¶¶240-241, 318.

f. Claim 7.

Fowlow in view of Kimera renders this claim obvious as discussed in Section V.C.2.f. Ex.1009, 2:22-25, 7:11-23, 10:28-34, 10:49-50; Ex.1002, ¶¶242-243, 319.

g. Claim 8.

Fowlow in view of Kimera renders this claim obvious as discussed in Section V.C.2.g. Ex.1009, 6:61-67, 7:11-23, 10:28-34; Ex.1002, ¶¶244-247, 320.

h. Claim 9.

Fowlow in view of Kimera renders this claim obvious as discussed in Section V.C.2.h. Ex.1009, Abstract, 2:34-40 10:15-65; Ex.1002, ¶¶248-249 321.

i. Claim 10.

Fowlow in view of Kimera renders this claim obvious as discussed in Section V.C.2.i. Ex.1009, 2:37-47, 4:3-15; Ex.1002, ¶¶250-251, 322.

j. Claim 11.

Fowlow in view of Kimera renders this claim obvious as discussed in Section V.C.2.j. Ex.1009, 2:22-25, 7:11-23, 10:49-50; Ex.1002, ¶¶252-253, 323.

k. Claim 12.

Fowlow in view of Kimera renders this claim obvious as discussed in Section V.C.2.k. Ex.1009, 6:5-8, 7:39-65, 8:27-31; Ex.1002, ¶¶254-258, 324.

l. Claim 13.

Fowlow in view of Kimera teaches performing security verification based on one or more policies as discussed in Section V.C.2.l. Ex.1009, 5:59-61, 8:2-13; Ex.1002, ¶¶259-261, 325. Additionally, Kimera’s “verification service performs checking of incoming code for typesafety and conformance to interfaces” as discussed above. Ex.1008, 2; Ex.1002, ¶326. Fowlow-Kimera renders obvious this claim. Ex.1002, ¶327.

E. Ground 3: Claims 1-7, 9-13 is Unpatentable Under 35 U.S.C. §103(a) as Obvious over Parthasarathy in view of Kimera

Claims 1-7, and 9-13 are obvious in view of the combination of Parthasarathy and Kimera. Ex.1002, ¶328.

1. Motivation to Combine

Parthasarathy and Kimera are both directed to acquiring software capable of running on client devices from a remote network. Ex.1002, ¶¶329-331. Parthasarathy acquires software applications and classes via a software download control module, which manages additional modules for “downloading, verifying, installing, registering and displaying computer software components....” Ex.1007, 1:11-16, Abstract, 2:65-67. The requested software is located on remote computers

across a network, such as the Internet or an intranet. *Id.* Kimera, relatedly, describes acquiring software applications from the Internet via a “centralized server” that includes a verification service, compiler, audit manager, and code cache. Ex.1008, 1-2. Both are from the same field of endeavor as each other and the ’075 patent.

To accomplish its tasks relating to downloading, verifying, and installing software components, Parthasarathy includes a “software component download module 58” that manages “a data download module 60, a code verification module 62, and a code installation module 64.” *Id.*, Abstract, 2:67-3:14, 8:13-19, Fig. 2.

Parthasarathy’s software component download module resides on the client or local computer, however, certain tasks, such as determining “[i]f a more recent version of the desired software component is available,” requires the module to communicate back and forth with remote computers. Ex.1007, 8:20-31; Ex.1002, ¶332. A POSITA would have recognized that the fact that this overhead would be taking place on various local computers, which would be inefficient and lead a POSITA to explore the benefits of centralizing the module to increase efficiency. Ex.1002, ¶¶332-334; *see also* Ex.1008, 1 (recognizing “the point-to-point security model forces every client to support a local verifier and compiler” meaning “the resource requirements for Java end points are very high”).

To this end, a POSITA would have been motivated to look to Kimera for its teaching of a centralized server (as represented by the Kimera Firewall below) in the

client-remote computer architecture described in Parthasarathy to simplify the system, making it faster, cheaper, and more secure. Ex.1002, ¶335. Indeed, inefficiencies in communication and verification on local computers for Java-based systems for downloading applets is exactly the shortcoming that Kimera identifies and seeks to remedy through a centralized server. Ex.1008, 1 (“locating verification, access control, auditing, and compilation services in centralized servers enables the same services to be performed faster, cheaper, and without introducing security vulnerabilities.”); Ex.1002, ¶¶334-335. The combined system would simply offload the applet locating functionality and verifying of Parthasarathy to the centralized processing server of Kimera in the same way this centralization is implemented in Kimera itself. Ex.1008, 1; Ex.1002, ¶336. Thus, the Parthasarathy-Kimera combination provides a centralized server with the capability of requesting and downloading software components from a remote computer or server and processing and verifying that code before sending it to the local, client computer. Ex.1002, ¶¶336-340.

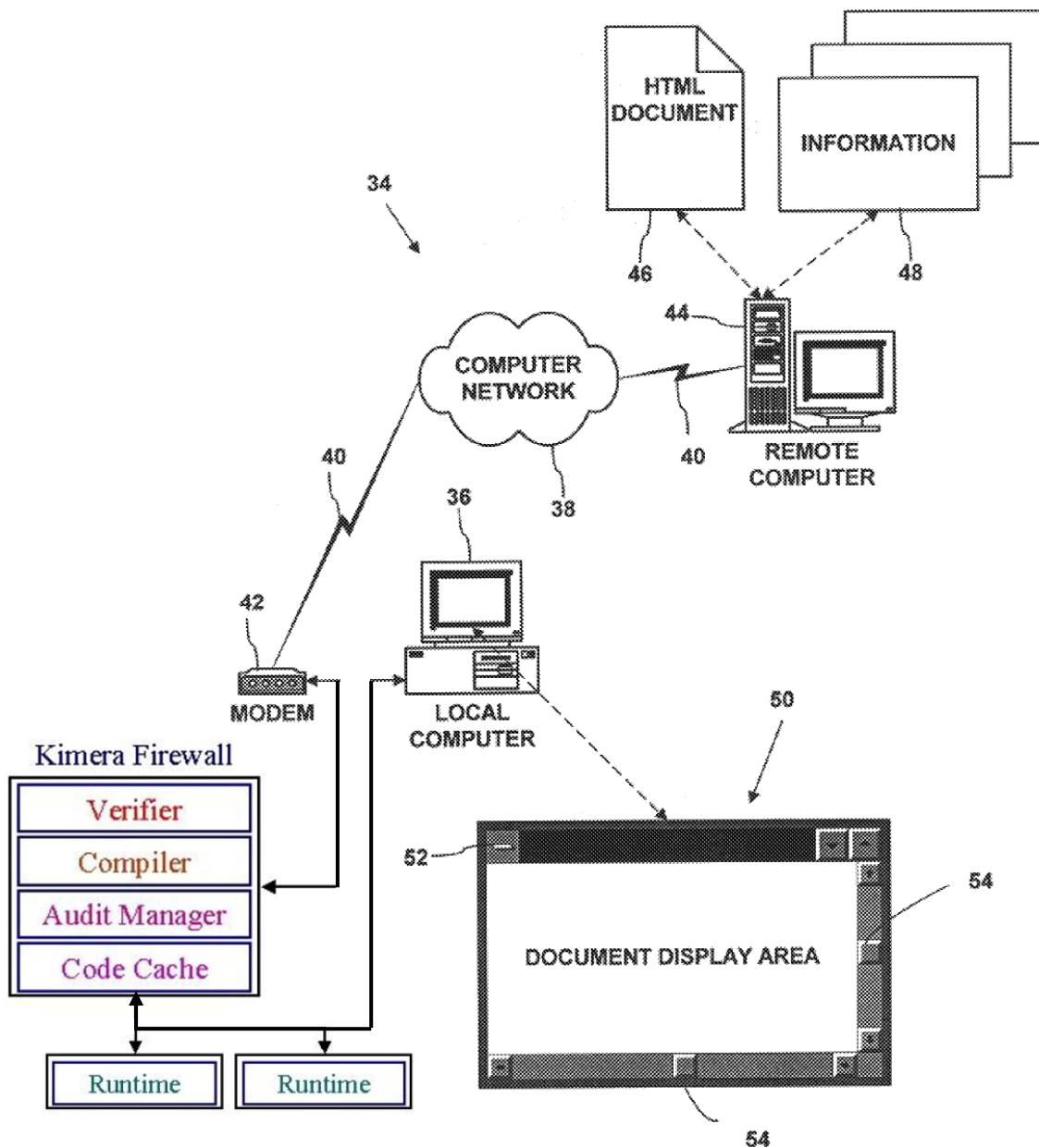


FIG. 2

Ex.1007, Fig. 2 (annotated); Ex.1008, 1; Ex.1002, ¶336. A POSITA would have been motivated to utilize a centralized server taught by Kimera for downloading and verifying each requested software component to achieve the security and efficiency benefits taught by Kimera by running these processes at centralized server as

opposed to on client machines. Ex.1007, 8:66-9:8, 9:33-42, Fig. 4; Ex.1008, 1-2; Ex.1002, ¶¶337-340.

In the combination, each of the respective disclosures of Parthasarathy and Kimera function in the same manner as in their respective sources and are therefore simply a rearranging of known components in accordance with known methods to provide predictable results. *KSR*, 550 U.S. at 416; *Smith & Nephew*, 721 F.3d at 1375. Furthermore, given the similarities between Parthasarathy and Kimera, a POSITA would have made this combination with a high expectation of success. Ex.1002, ¶341. By adding the Kimera firewall or centralized server between the modem and local computer of Parthasarathy, the combined system would incorporate the benefits of Kimera while alleviating the negative concerns addressed in both references. Ex.1002, ¶¶337-342. The centralized server would both prevent rogue local computers from modifying security settings and eliminate the need to verify software on the local computer altogether, bolstering security, and would improve speed by holding processed software components in cache memory. *Id.*, ¶¶331, 338-339. Further, while the local computers need not be modified to combine Parthasarathy and Kimera, further bolstering the expectation of success, it would be possible to eliminate a JIT compiler, verifier, and interpreter from the local computers, making the system cheaper. *Id.*, ¶342; Ex.1008, 2.

A POSITA would have been motivated to combine Parthasarathy and Kimera to arrive at the claimed method. *Id.*, ¶343.

2. Independent Claim 1

a. [Ipre]

To the extent the preamble is limiting, Parthasarathy-Kimera teaches it. Parthasarathy discloses “a network browsing environment” that provides a “method and system to automatically locate, download, verify, install, register and display computer software components from a computer network.” Ex.1007, Abstract, 2:65-67, 4:1-2. The “network browsing environment” includes a local, client computer and a remote, server computer connected through a computer network 38 (*e.g.*, the Internet), where requests are made for software to be downloaded to the client from a server. *Id.*, 4:44-47, 7:3-8, 8:19-31, Figure 2.

To the extent the preamble is limiting, a POSITA would have found it taught by Parthasarathy-Kimera. Ex.1002, ¶¶344-346.

b. [Ia]

Parthasarathy’s “network browsing environment” includes a software component download module (*i.e.*, applet server manager) that manages requests from the local computer (*i.e.*, the client computer) for software components (*i.e.*, applets). Ex.1007, 12:6-13. Once the client computer requests a software and deems a download is necessary (because the client either does not have the requested

software or has an out-of-date version), “[a] data download module downloads computer software components in an asynchronous manner from one or more remote computers. A code verification module verifies that the downloaded computer software components are safe to install on a local computer.... A code installation module installs the verified computer software components on the local computer.” Ex.1007, 2:65-3:14, 8:19-34. Notably, the software component download module (*i.e.*, the applet server manager) manages these other system modules. *Id.*, 2:67-3:6, Fig. 3.

As explained in Section V.E.1, this software component download module would be located in a centralized server in the Parthasarathy-Kimera system in order to increase overall system efficiency and performance. Ex.1008, 1, 2; Ex.1002, ¶¶347-350. Combining these functionalities into a single, centralized server would “enable[] the same services to be performed faster, cheaper, and without introducing security vulnerabilities.” Ex.1008, 2. For example, while Parthasarathy includes a data download module and a code verification module, the centralized Kimera server would be configured to include this functionality. *Id.*; Ex.1007, 2:65-3:14, 8:19-34; Ex.1002, ¶351.

A POSITA would have understood that in the Parthasarathy-Kimera system, requests for software components would pass through the centralized server to enable the server to download and verify code before being installed on the client

computer. Ex.1002, ¶353. Kimera’s code cache, which stores “the results of recent compilations in the central compilation server,” confirms requests would need to be routed through the centralized server to determine whether the requested code was stored in this code cache. Ex.1008, 2; Ex.1002, ¶352. If the software was stored in cache memory, it would be served to the client from the cache. *Id.*, ¶352. If not, a POSITA would have recognized the central server would be configured to manage the requests, seeking to download the software components from other servers across the network. *Id.*, ¶¶353-357; *see also* Ex.1007, 7:3-26 (noting “executable programs” may “reside at the remote computer 44 or other remote computers” connected to the computer network 38).

Kimera’s centralized server functionality, including the compilation, verification, and caching functionality, is consistent with the functionality of the applet server manager of the ’075 patent, further confirming the step of processing applets at an applet server manager was well-known. Ex.1002, ¶359.

Parthasarathy-Kimera, therefore, teaches this limitation. Ex.1002, ¶358.

c. [1b]

The centralized server of Parthasarathy-Kimera has access to one or more networks because Parthasarathy’s method and system “are used by applications (*e.g.*, network browsers, network servers) to automatically download and install software components from code depositories on computer networks (*e.g.*, the

Internet, or local corporate intranets)....” Ex.1007, 2:65-3:27, 7:15-26, 8:67-9:8, Fig. 2. Since this network access of Parthasarathy’s software component download module would be incorporated into the centralized Kimera server as detailed above a POSITA would have understood that it can access files on networks including local intranets or the Internet. *Id.*; Ex.1002, ¶360.

Parthasarathy-Kimera, therefore, teaches this limitation. Ex.1002, ¶361.

d. [1c]

Parthasarathy-Kimera teaches receiving a request for software at a centralized server because Parthasarathy teaches “[w]hen a request is made to download a desired software component stored on a remote computer 44, the software component download module 58 determines if the desired software component is already installed on the local computer 36.” Ex.1007, 8:20-24. As discussed above, Kimera’s centralized server would house the software component download module functionality, which a POSITA would have understood received the request before performing or orchestrating the steps of locating, downloading, verifying, installing, registering, and returning the requested software component. Ex.1007, 2:65-3:21, 8:66-9:8, 9:33-42, Fig. 4; Ex.1008, 1-2; Ex.1002, ¶¶362-363.

Parthasarathy-Kimera, therefore, teaches this limitation. Ex.1002, ¶364.

e. [Id]

Parthasarathy-Kimera teaches passing the request from the applet server manager (*e.g.*, the software component download module located in the centralized server) to the one or more networks (*e.g.*, remote computers on the Internet or a corporate intranet).

To download the desired software control from the remote computer, the desired software component must first be located, which Parthasarathy teaches occurs via a URL or a network search path. Ex.1007, 9:9-21 (“The URL or network search path are used to first locate 68 a control file on a remote computer....The control file is used to determine the location of the software components on one or more remote computers 44 to download to the local computer 36.”). A POSITA would have understood that in locating the requested software, the software component download module of the centralized server would pass the request for a specific software component to a network-connected remote computer, to locate and download the requested software components. *Id.*; Ex.1002, ¶¶365-367.

A POSITA would have expected client requests to be passed through the centralized server in this way because Parthasarathy itself teaches that its software requests contain sufficient information to allow them to be passed from one server to another. Ex.1002, ¶368. For example, Parthasarathy describes that “the control file” in its request includes “references to one or more computer software

components on the specific remote computer *or one or more links to one or more computer software components on one or more other remote computers....*”

Ex.1007, 32:40-44. Inter-linking was commonly used at the time of the '075 patent, such as in standard HTML which “can incorporate other additional information content 48, such as images, audio, video, executable programs, etc. which also *reside at the remote computer 44 or other remote computers.*” Ex.1007, 7:15-19, 1:48-51 (“HTML allows embedded ‘links’ to point to other data or documents, which may be found *on the local computer or other remote Internet host computers.*”). A POSITA would have known that when linked software components were found on another remote computer, the software download control module would then pass the request to that other computer to locate and download the desired software and would have expected this same passing functionality in the centralized server of Parthasarathy-Kimera since all requests would first go to the centralized server at least to see if the needed software was stored in its cache. Ex.1002, ¶368.

Parthasarathy-Kimera, therefore, teaches this limitation. *Id.*, ¶369.

f. [1e]

Parthasarathy-Kimera teaches receiving the requested software (“applet”) at the centralized server (“applet server manager”) because Parthasarathy discloses a “method and system for automatically locating, **downloading**, verifying, registering, installing and displaying a software component from a remote computer...[including

a] data download module [that] downloads computer software components in an asynchronous manner from one or more remote computers.” Ex.1007, 2:61-3:8, 8:19-30, 8:66-9:3.

A POSITA would have understood that all downloaded software components and applets would have been received into the software component download module of the centralized server from the remote computer connected via the Internet or corporate intranet. *See* Ex.1007, 1:20-32; *see also* Ex.1008, 1 (showing all data coming from the Internet into the centralized server before going to client devices); Ex.1002, ¶170. The software components are then processed at the centralized server before being returned to the local, client computers. Ex.1008, 1, 2; Ex.1002, ¶170.

Parthasarathy-Kimera, therefore, teaches this limitation. Ex.1002, ¶171.

g. *[lf]*

The Parthasarathy-Kimera system discloses systems and methods for “downloading, **verifying**, installing, registering and displaying computer **software components**.” Ex.1007, 1:11-16, Abstract, 2:65-67; Ex.1008, 2. Kimera similarly discloses a “verification service” located in the centralized server that “performs checking of incoming code for typesafety and conformance to interfaces.” Ex.1008, 2. As explained in Kimera, moving the verification service to the centralized server “enables the same services to be performed faster, cheaper, and without introducing

security vulnerabilities.” Ex.1008, 1; Ex.1002, ¶¶372-375, 387. Performing the “verifying” step is sufficient to find the Parthasarathy-Kimera system teaches this claim limitation. *Dealertrack*, 2008 WL 5792509 at *7.

To the extent verifying alone is insufficient to render obvious this claim, Parthasarathy in view of Kimera also teaches the “compressing” and “optimizing” steps. Ex.1002, ¶¶377-379. The prior art teaches the claimed “compressing” step through Parthasarathy’s disclosure of a “cabinet file [which] contains one or more files, all of which are downloaded together in a *compressed* ‘cabinet.’” Ex.1007, 15:65-16:1. Compressing the one or more applets before sending them to the client was, therefore, well known. Ex.1002, ¶¶126-127, 379-380; Ex.1014, 107 (compression), 130 (data compression).

Applet compression further comports with the teachings of Kimera, which (1) aims to reduce client system requirements and (2) cautions against excessive communication overhead between the client and the centralized server because this overhead would hamper the benefits of centralization. Ex.1008, 1, 2. A POSITA would, therefore, have known to include compression for all data sent from the applet server manager to the simplified client, allowing client computers with “a sizeable reduction in...memory requirements” to be able to receive the applications. *Id.*, 2; Ex.1002, ¶381. Compiling and caching applets on the centralized server

would also reduce communication overhead, further optimizing the applet retrieval process. Ex.1002, ¶¶382, 116; Ex.1018, 43; Ex.1014, 343.

The Parthasarathy-Kimera system also teaches the “optimizing” step because a POSITA would have known to include optimizing at the centralized server to take advantage of Kimera’s client-simplification goals most effectively. Ex.1008, 1, 2; Ex.1002, ¶¶383-384. Specifically, Kimera describes that moving the “verification, access control, auditing, and compilation services” to the centralized server “makes a sizable reduction in computation and memory requirements of the end-points.” Ex.1008, 1, 2. A POSITA would have understood that these benefits suggest optimizing the applets before sending the one or more applets to the one or more client computers in order to achieve faster services performed on the most inexpensive clients possible. Ex.1002, ¶¶384-386.

A POSITA would have been motivated to combine Parthasarathy and Kimera for the reasons enumerated above. *See* Section V.E.1. Ex.1007, Abstract, 2:67-3:14, 8:13-31, 8:66-9:8, 9:33-42, 23:39-56, Figs. 2, 4; Ex.1008, 1, 2; Ex.1002, ¶¶329-343. Parthasarathy-Kimera, therefore, teaches this limitation. Ex.1002, ¶¶376, 388.

h. [Ig]

As introduced above, the local computer of Parthasarathy can request “to download a desired software component stored on [the] remote computer 44.” Ex.1007, 8:19-20. In response to receiving a request, the software component is

downloaded to the local computer and the “software components are returned 78 to the client application (*e.g.*, the network browser) which made the request for the computer software components.” *Id.*, 29:13-20. In the context of the combination with Kimera, the software would be sent to the client from the centralized server, as described above with reference to elements [1a] and [1e]. Ex.1002, ¶389.

Parthasarathy-Kimera, therefore, teaches this limitation. Ex.1002, ¶390.

Thus, Parthasarathy-Kimera renders obvious claim 1. Ex.1002, ¶391.

3. Dependent Claims 2-7, 9-13

a. Claim 2.

As discussed in Section V.D.3.a, Kimera teaches this claim because applets that failed to satisfy the typesafety and conformance policies would be discarded. Ex.1008, 2; Ex.1002, ¶¶305, 392. A POSITA would have been motivated to combine Parthasarathy and Kimera for the reasons described in Section V.E.1. Ex.1007, Abstract, 2:67-3:14, 8:13-31, 8:66-9:8, 9:33-42, 23:39-56, Figs. 2, 4; Ex.1008, 1, 2; Ex.1002, ¶¶329-343, 392.

Parthasarathy-Kimera, therefore, renders obvious this claim. Ex.1002, ¶393.

b. Claim 3.

As discussed in Section V.E.2.g, Parthasarathy in view of Kimera teaches processing the applet, including compressing, verifying, and optimizing the applet, before sending the applet to the client computer. Ex.1007, Abstract, 1:20-32; 2:65-

3:27, 8:19-35, 8:66-9:3, 15:65-16:1.; Ex.1008, 1-2, Ex.1002, ¶¶372-388. For example, Kimera includes a compiler that compiles “Java bytecode into native machine code...at a central location.” Ex.1008, 2. This central location would be the applet server manager, which both compiles code as well as verifies and applies tamper proof signatures to code passed from the applet server manager to the client. *Id.*; Ex.1002, ¶395. A POSITA would have understood that both compiling code from “Java bytecode into native machine code” and applying cryptographic signatures would transform the applets by modifying the data in the applet before it is sent to the client computer. Ex.1002, ¶¶101, 128-132, 395.

Parthasarathy-Kimera, therefore, renders obvious this claim. Ex.1002, ¶396.

c. Claim 4.

Parthasarathy teaches that certain networks, such as corporate intranets, can be “automatically trusted and need no verification.” Ex.1007, 9:40-42; Ex.1002, ¶397. Additionally, Kimera discloses a Firewall, which a POSITA would understand is designed to protect a trusted, internal network (*e.g.*, an intranet) from an untrusted, external network (*e.g.*, the Internet). Ex.1008, 1; Ex.1002, ¶¶52, 398-399.

Parthasarathy-Kimera, therefore, renders obvious this claim. Ex.1002, ¶400.

d. Claim 5.

Parthasarathy refers to a variety of networks, including both the Internet and corporate intranets. Ex.1007, Abstract, 1:11-16, 3:22-27, 3:45-57. Parthasarathy explicitly describes corporate intranets as “automatically trusted and need[ing] no verification” because “all software components [are located] behind a corporate firewall.” *Id.*, 9:40-42. A POSITA would have understood that in contrast to the trusted internal cache of software components, the Internet at-large would include untrusted networks. Ex.1002, ¶¶52, 401; Ex.1014, 197.

Additionally, Kimera discloses a Firewall, which separates trusted and untrusted networks. Ex.1008, 1; Ex.1002, ¶¶398, 403. Kimera’s verifier will apply “[t]amper-proof signatures, using a cryptographically strong hash function, such as MD5,...to ensure the safety of the code....” Ex.1008, 2. A POSITA would have understood that tamper-proof signatures would be applied to code that had been received from an untrusted network and subsequently verified to be safe. Ex.1002, ¶402.

Parthasarathy-Kimera, therefore, renders obvious this claim. Ex.1002, ¶404.

e. Claim 6.

Parthasarathy discloses systems and methods to “locate, download, verify, install, register and display computer software components from a computer network.” Ex.1007, Abstract. Initially, “a request is made to download a desired

software component stored on a remote computer 44.” *Id.*, 8:19-20. The request “us[es] one of two designators: a URL or a network search path” to identify and locate the computer software component via a control file. *Id.*, 9:9-21.

Parthasarathy also claims a “control file [that] includes an initialization file that includes one or more universal resource locators pointing to one or more computer software components...being specified for the operating system or the CPU of the local computer.” *Id.*, cl. 57; Ex.1002, ¶405.

Parthasarathy-Kimera, therefore, renders obvious this claim. Ex.1002, ¶406.

f. Claim 7.

Parthasarathy discloses systems and methods to “locate, download, verify, install, register and display computer software components from a computer network.” Ex.1007, Abstract. In the request, the local computer may identify the form of the application, as specified in a HTTP header. *Id.*, 14:65-15:13; Ex.1002, ¶407. For example, “an Object Store on the network search path is an HTTP server that services requests for a desired control file.” *Id.*, 14:65-67. The control file would be located from the various Object Stores in the search path and “[s]pecifically, an Object Store will receive an HTTP POST request with data in the format below:

CLSID = {class id}
Version = a, b, c, d
MIMETYPE = mimetype”

Id., 14:67-15:7. “The Object Store parses this information, checks an internal database, and either fails, or redirects the HTTP request to the control file on the next remote computer in the network search path.” *Id.*, 15:8-13; Ex.1002, ¶408.

Parthasarathy also discloses queries to Object Stores that “include HTTP headers for ACCEPT (MIME type) and ACCEPT-LANGUAGE, thus specifying the desired platforms and language-localized implementation for a [software] component.” *Id.*, 15:30-38. When HTTP headers are added to HTTP requests, Object Stores can serve different code implementations based on differing platforms or even different languages. *Id.*; Ex.1002, ¶409.

Parthasarathy-Kimera, therefore, renders obvious this claim. Ex.1002, ¶410.

g. Claim 9.

“Documents for browsing with the illustrated browser can reside as files of a file system stored in the computer’s secondary storage 32 (FIG. 1), or reside as resources at a remote computer 44 (also referred to as a ‘site’ or ‘store’) connected to the computer network 38, such as a World Wide Web site on the Internet.” Ex.1007 at 7:3-8. A POSITA would have understood the “document for browsing with the illustrated browser” would be HTML documents, also known as web pages. Ex.1002, ¶411; *see also* Ex.1007, 9:3-8.

Parthasarathy-Kimera, therefore, renders obvious this claim. Ex.1002, ¶412.

h. Claim 10.

Parthasarathy's local computer browses for "files of a file system stored in the computer's secondary storage 32 (FIG. 1), or reside as resources at a remote computer 44 (also referred to as a 'site' or 'store') connected to the computer network 38, such as a World Wide Web site on the Internet." Ex.1007, 7:3-8. "Almost any type of computer software components can be obtained from a local or remote computer using system 56 and method 66 for many types of files or documents." *Id.*, 9:3-8. A POSITA would have understood the "document for browsing with the illustrated browser" and additional "files or documents" would include HTML documents defining an entire web page, or a portion of a web page. Ex.1002, ¶413; *see also* Ex.1007, 9:3-8.

Further, the web page, may include embedded HTML <OBJECT> tags for dynamic or interactive multimedia. *Id.*, 9:63-67. The HTML <OBJECT> tags include multiple attributes, including, for example, the CODEBASE attribute, which "represent[s] the desired version of a software component on a remote server." *Id.*, 10:4-27. By requesting software corresponding to an embedded HTML <OBJECT> tag, a POSITA would have understood that the identified form is a portion of the web page in which the tag is embedded. Ex.1002, ¶414.

Parthasarathy-Kimera, therefore, renders obvious this claim. Ex.1002, ¶415.

i. Claim 11.

Parthasarathy discloses a method of downloading computer software components from a computer network using HTTP requests. Ex.1007, 8:66-9:3, 14:65-15:13, 15:39-44. A POSITA would have understood that in the HTTP protocol, data returned to the client is sent in an HTTP response. *Id.*, 1:45-46, 15:39-44; Ex.1002, ¶¶416-417, 55, 62.

Parthasarathy-Kimera, therefore, renders obvious this claim. Ex.1002, ¶418.

j. Claim 12.

“When a request is made to download a desired software component stored on a remote computer 44, the software component download module 58 determines if the desired software component is already installed on the local computer 36. If the desired software component is already installed on the local computer 36, then the desired software component on the local computer 34 is checked to see if a more recent version is available.” Ex.1007, 8:20-27. This step of determining whether the desired software component is either already installed on the local computer or is not the most recent version confirms Parthasarathy teaches processing the request before the request is sent to the network. *Id.*, 8:27-33; Ex.1002, ¶¶419-420.

Parthasarathy-Kimera, therefore, renders obvious this claim. Ex.1002, ¶421.

k. Claim 13.

As discussed in Section V.D.3.1, Kimera teaches this claim because Kimera checks incoming code for typesafety and conformance to interfaces. Ex.1008, 2; Ex.1002, ¶¶326, 422. A POSITA would have been motivated to combine Parthasarathy and Kimera for the reasons described in Section V.E.1. Ex.1007, Abstract, 2:67-3:14, 8:13-31, 8:66-9:8, 9:33-42, 23:39-56, Figs. 2, 4; Ex.1008, 1, 2; Ex.1002, ¶¶329-343, 422.

Parthasarathy-Kimera, therefore, renders obvious this claim. Ex.1002, ¶423.

F. Ground 4: Claim 8 is Unpatentable Under 35 U.S.C. §103(a) as Obvious over Parthasarathy in view of Kimera and Angles

Claim 8 is obvious in view of the combination of Parthasarathy, Kimera, and Angles. Ex.1002, ¶424.

1. Motivation to Combine

Parthasarathy and Kimera are both directed to requesting software capable of running on client devices from a remote network, such as the Internet, and delivering the requested software to the client, as described in Section V.E.1. Ex.1007, Abstract, 2:67-3:14, 8:13-31, 8:66-9:8, 9:33-42, 23:39-56, Figs. 2, 4; Ex.1008, 1, 2; Ex.1002, ¶¶329-343, 425. The requests utilize “HTTP headers for ACCEPT (MIME type) and ACCEPT-LANGUAGE [that] specify[] the desired platforms and

language-localized implementation for a [software] component.” *See* Ex.1007, 15:30-38; Ex.1002, ¶426.

Angles similarly describes “a system and method for delivering customized electronic advertisements in an interactive communication system.” Ex.1010, Abstract. Angles describes assigning a consumer member code used for tracking consumer registration with an advertisement provider service. *Id.*, 10:60-11:2. This member code is stored as a cookie. *Id.*, 11:2-4. The cookie is a part of the “HTTP header line” that is transmitted with all HTTP requests. *Id.*, 11:13-15; Ex.1002, ¶¶427-428, 66-71.

To the extent Parthasarathy’s disclosure of use of HTTP requests and HTTP headers did not already implicitly incorporate cookies as part of their HTTP processing, which were well known in 1996, a POSITA would have been motivated to combine at least the cookie disclosure of Angles into the Parthasarathy-Kimera system to augment the HTTP headers to more accurately and efficiently locate, download, verify, install, and register requested software components, and improve the user experience when browsing the Web. Ex.1002, ¶¶429-431; *see also* Ex.1010, 11:7-12. Specifically, if the HTTP headers of the Parthasarathy-Kimera system included headers, the browser could “remember some specific information across several pages.” A POSITA would recognize the benefits of this functionality and

would have been motivated to include cookies in the HTTP headers of requests for software components and other applets. Ex.1002, ¶¶429-431.

Finally, in the combination, each of the respective disclosures of Parthasarathy, Kimera, and Angles function in the same manner as in their respective sources and are therefore simply a rearranging of known components in accordance with known methods to provide predictable results. *KSR*, 550 U.S. at 416; *Smith & Nephew*, 721 F.3d at 1375; Ex.1002, ¶432.

2. Dependent Claim 8

a. Claim 8.

The Parthasarathy-Kimera system teaches the method of claim 1 for the reasons set forth in Section V.E.2. Ex.1007, Abstract, 1:20-32, 1:48-51, 2:61-3:27, 4:1-2, 4:44-47, 7:3-26, 8:19-35, 8:66-9:21, 9:33-42, 12:6-13, 15:65-16:1, 29:13-20, 32:40-44, Figs. 2-4; Ex.1008, 1-2; Ex.1002, ¶¶344-391. A POSITA would have further incorporated the cookies disclosed in Angles to enable functionality such as filling a virtual shopping cart by adding items from a virtual shopping mall, and pay for all of the items once the consumer had finished collecting their desired items. Ex.1010, 11:7-12; Ex.1002, ¶¶434-435 (citing Ex.1017, 2). In this way, once a user requested an applet, such as a webpage or other software component, the requested applet would be specified according to the cookie. Ex.1010, 11:7-12; Ex.1002, ¶¶433-436.

For example, when a user first visits a webpage and selects items to include in a virtual shopping cart, a POSITA would have understood a cookie is created and stored on the client machine to remember the contents of the shopping cart. Ex.1010, 11:7-27; Ex.1002, ¶¶434-435. “Whenever the browsing software sends a HTTP request for a URL on a server for which it has stored cookies, it includes a line in the form: Cookie: NAME=VALUE; NAME=VALUE;...which lists all cookies that apply to that particular URL.” Ex.1010, 11:27-31. A POSITA would have understood that as the client moved to another webpage of the virtual shopping mall, the HTTP request sent to the applet server manager and ultimately to the network would be further specified by the cookie stored on the client device. Ex.1002, ¶¶433, 436-437.

Parthasarathy-Kimera-Angles, therefore, renders obvious this claim. Ex.1002, ¶438.

G. The Petition Should Not Be Discretionarily Denied

1. *Advanced Bionics*

The claims of the '075 patent have not been considered in view of any of the prior art relied upon in the asserted grounds of this Petition. Accordingly, the present Petition should not be discretionarily denied under 35 U.S.C. §325(d). *Advanced Bionics, LLC v. MED-EL Elektromedizinische Geräte GmbH*, IPR2019-01469, Paper 6 (PTAB Feb. 13, 2020) (precedential).

The parent and grandparent patents to the '075 patent included Kimera in an IDS, but the applicant did not include Kimera in any IDS submitted in the application that resulted in the '075 patent. Ex.1004, 125. In fact, the applicant did not submit any IDS in the prosecution of this patent. *See generally* Ex.1004. Even in the parent and grandparent patents, Kimera was cited in an IDS, but did not form the basis of any rejection or other substantive discussion.⁵ The Board has consistently found that when a reference is not the basis of rejection, and merely cited in an IDS, it weighs “strongly against” exercising discretionary denial. *See, e.g., CODE200, UAB v. Bright Data Ltd.*, IPR2022-00353, Paper 8 at 10 (PTAB July 1, 2022); *Samsung Elecs., Co. v. G+ Commc'ns, LLC*, IPR2022-01598, Paper 10 at 13 (PTAB Apr. 4, 2023).

2. *Fintiv*

The Board should not deny this Petition in view of 35 U.S.C. §314(a). *Apple Inc. v. Fintiv, Inc.*, IPR2020-00019, Paper 11 (PTAB Mar. 20, 2020) sets forth six factors to consider in a discretionary denial under 35 U.S.C. §314(a). When assessed holistically, these six factors do not warrant discretionary denial.

First, upon institution of IPR, Petitioner will file a motion to stay.

⁵ In the immediate parent application, the examiner crossed out the page of the IDS citing Kimera.

Second, there is no trial date presently set for any litigation involving the '075 patent. This favors not exercising discretionary denial.

Third, the parallel proceeding is in its infancy. No claim construction, fact discovery, nor expert discovery has occurred.

Fourth, this petition challenges more claims than have been asserted in the parallel proceeding, meaning that there are issues in this proceeding that would not be addressed in the district court.⁶

Fifth, while Co-Petitioners Capital One, National Association and Hulu, LLC are defendants in the parallel proceedings, there has been very little, if any, overlap of the issues because the parallel proceedings are in their infancy.

Sixth, the merits of the arguments presented to the Board here present a compelling unpatentability challenge, also weighing in favor of institution.

Balancing these factors holistically, the Board should not exercise its discretion to deny this Petition.

VI. CONCLUSION

The Petition demonstrates a reasonable likelihood that at least one claim is

⁶ Petitioner reserves the right to file a Sotera-type stipulation at a later date. *See BMW of North America, LLC v. NorthStar Systems LLC*, No. IPR2023-00890, 2023 WL 7449411, at *3 (P.T.A.B. Nov. 9, 2023).

unpatentable. Ex.1002, ¶439. Accordingly, all grounds in the Petition should be instituted. *SAS Institute Inc. v. Iancu*, 138 S.Ct. 1348, 1359-60 (2018).

Respectfully submitted by

K&L GATES LLP,

By: /Jason A. Engel/
Jason A. Engel
Reg. No. 51,654

CLAIM APPENDIX OF THE CHALLENGED CLAIMS

- [1pre] A method for delivering one or more applets to one or more client computers, comprising, in no particular order, the steps of:
- [1a] configuring an applet server manager at a server computer to manage at least one request from the one or more client computers for the one or more applets,
- [1b] the applet server manager having access to one or more networks;
- [1c] receiving the at least one request at the applet server manager;
- [1d] passing the at least one request from the applet server manager to at least one of the one or more networks;
- [1e] receiving the one or more applets at the applet server manager from the at least one of the one or more networks;
- [1f] processing the one or more applets at the applet server manager, wherein processing the one or more applets includes at least one of the following steps:
- compressing the one or more applets before sending the one or more applets to the one or more client computers,
- optimizing the one or more applets before sending the one or more applets to the one or more client computers, and
- verifying the one or more applets before sending the one or more applets to the one or more client computers; and
- [1g] sending the one or more applets from the applet server manager to the one or more client computers.

- [2] The method of claim 1 wherein processing the one or more applets further includes discarding at least one of the one or more applets based on one or more policies.
- [3] The method of claim 1 wherein processing the one or more applets further includes transforming at least one of the one or more applets by modifying the data in the at least one of the one or more applets before sending the one or more applets to the one or more client computers.
- [4] The method of claim 1 wherein at least one of the one or more networks is a trusted network.
- [5] The method of claim 1 wherein at least one of the one or more networks is an untrusted network.
- [6] The method of claim 1 wherein the at least one request specifies the name of at least one of the one or more applets using a Uniform Resource Locator.
- [7] The method of claim 1 wherein the at least one request specifies at least one of the one or more applets using an HTTP header.
- [8] The method of claim 1 wherein the at least one request specifies at least one of the one or more applets using a cookie.

- [9] The method of claim 1 wherein processing the one or more applets includes producing at least one of the one or more applets in the form of a web page.
- [10] The method of claim 1 wherein processing the one or more applets includes producing at least one of the one or more applets in the form of a portion of a web page.
- [11] The method of claim 1 wherein sending the one or more applets includes sending at least one of the one or more applets to the one or more client computers in an HTTP response.
- [12] The method of claim 1 wherein the at least one request is processed prior to passing the at least one request from the applet server manager to at least one of the one or more networks.
- [13] The method of claim 1 wherein processing the one or more applets further includes performing security verification based on one or more policies.

Certification of Service Under 37 C.F.R. §42.6(e)(4)

A copy of this Petition for *Inter Partes* Review and supporting materials has been served at the following correspondence address of record for the subject patent via Federal Express Priority Overnight® on this 15th day of March, 2024:

KOWERT, HOOD, MUNYON, RANKIN & GOETZEL, P.C.
Jeffrey Hood
1120 S. Capital of Texas Hwy
Building 2, Suite 300
Austin, TX 78746

With an electronic courtesy copy to litigation counsel:

James M. Lennon
jlennon@devlinlawfirm.com
Timothy Devlin
tdevlin@devlinlawfirm.com
Christopher Clayton
cclayton@devlinlawfirm.com
1526 Gilpin Ave
Wilmington, Delaware 19806
DEVLIN LAW FIRM LLC

By: /Jason A. Engel/
Jason A. Engel
Reg. No. 51,654
Customer No. 24573
K&L GATES LLP
Jason.Engel.PTAB@klgates.com
telephone number: (312) 807-4236
fax number: (312) 827-8145
70 W. Madison Street, Suite 3100
Chicago, IL 60602

Certification of Word Count Under 37 C.F.R. §42.24(d)

The undersigned hereby certifies that the foregoing Petition for *Inter Partes* Review contains **13,546** words, not including a table of contents, table of authorities, mandatory notices under §42.8, certificate of service, certificate of word count, appendix of exhibits or appendix of claim listing as specified by 37 C.F.R. §42.24, according to the word count feature of the word-processing software used to prepare the Petition.

By: /Jason A. Engel/
Jason A. Engel
Reg. No. 51,654
Customer No. 24573
K&L GATES LLP
Jason.Engel.PTAB@klgates.com
telephone number: (312) 807-4236
fax number: (312) 827-8145
70 W. Madison Street, Suite 3100
Chicago, IL 60602