

UNITED STATES PATENT AND TRADEMARK OFFICE

PATENT TRIAL AND APPEAL BOARD

Hulu, LLC; Capital One, National Association
Petitioner

v.

Implicit, LLC,
Patent Owner

CASE: IPR2024-00787
U.S. PATENT NO. 6,976,248

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)) .	vii
D. PAYMENT OF FEES (37 C.F.R. §§42.15 and 42.103).....	viii
II. GROUNDS FOR STANDING (37 C.F.R. §42.104(a)).....	1
III. STATEMENT OF PRECISE RELIEF REQUEST AND REASONS THEREFORE	1
IV. SUMMARY OF THE '248 PATENT	2
A. Background of the Technology	2
B. Prosecution History	4
V. CLAIM CONSTRUCTION	5
VI. ARGUMENTS	5
A. Level of Ordinary Skill in the Art.....	5
B. Prior Art.....	6
1. Johnson (Ex.1011)	6
2. Parthasarathy (Ex.1007).....	7
3. Kimera (Ex.1008).....	9
4. Fowlow (Ex.1009)	11
C. Ground 1: Claims 1 and 9 are Unpatentable Under 35 U.S.C. §103(a) as Obvious over Johnson	13
1. Independent Claims 1 and 9.....	13
D. Ground 2: Claims 1-13 are Unpatentable Under 35 U.S.C. §103(a) as Obvious over Parthasarathy in view of Kimera.....	21
1. Motivation to Combine	21
2. Independent Claim 1	26
3. Dependent Claims 2-4.....	34
4. Independent Claim 5	37

5.	Dependent Claims 6-8.....	39
6.	Independent Claim 9	41
7.	Dependent Claims 10-13.....	42
E.	Ground 3: Claims 1-13 are Unpatentable Under 35 U.S.C. §103(a) as Obvious over Fowlow in view of Kimera	46
1.	Motivation to Combine	46
2.	Independent Claim 1	52
3.	Dependent Claims 2-4.....	60
4.	Independent Claim 5	63
5.	Dependent Claims 6-8.....	67
6.	Independent Claim 9	68
7.	Dependent Claims 10-13.....	69
F.	The Petition Should Not Be Discretionarily Denied.....	74
1.	<i>Advanced Bionics</i>	74
2.	<i>Fintiv</i>	75
VII.	CONCLUSION.....	76
I.	CLAIM APPENDIX OF THE CHALLENGED CLAIMS	77

LIST OF EXHIBITS

Exhibit No.	Description
1001	U.S. Patent No. 6,976,248 (the “’248 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. 09/968,704
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, Sean McDirmid, & 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	U.S. Patent No. 5,923,885 to Johnson (“Johnson”)
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
1022	Original claims of U.S. Patent Application No. 11/933,161 (Oct. 31, 2007)
1023	Non-final Rejection of U.S. Patent Application No. 11/933,161 (Feb. 14, 2011)
1024	Applicant Response to Non-final Rejection of U.S. Patent Application No. 11/933,161 (Aug. 11, 2011)

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):

- Hulu, LLC; and
- Capital One, National Association.

B. Related Matters

Patent Owner, Implicit, LLC, asserted U.S. Patent No. 6,976,248 (“’248 patent”) in the following 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 ’248 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 Networks, Inc. v. Hewlett-Packard Company*. (N.D. Cal.) Case No. 3-10-cv-03746;
- *Implicit Networks, Inc. v. VMWare, Inc. et al.* (N.D. Cal.) Case No. 3-10-cv-00720;
- *Implicit Networks, Inc. v. Sybase, Inc. et al.* (N.D. Cal.) Case No. 3-09-cv-01478;
- *Implicit Networks, Inc. v. International Business Machines Corporation et al.* (N.D. Cal.) Case No. 3-09-cv-01342;
- *Sun Microsystems Inc. v. Implicit Networks, Inc.* (N.D. Cal.) Case No. 3-09-cv-00201; and
- *Implicit Networks, Inc. v. International Business Machines Corporation et al.* (N.D. Cal.) Case No. 2-08-cv-01080.

The '248 patent is directly related to the following issued U.S. Patents and abandoned applications:

- U.S. Patent No. 6,324,685;
- U.S. Patent No. 7,774,740;
- U.S. Patent No. 8,056,075;
- U.S. Patent No. 8,856,779;
- U.S. Patent No. 9,325,740;

- U.S. Patent No. 9,848,031; and
- U.S. Patent Application No. 15/845,414 which was abandoned.

Petitioner also has challenged U.S. Patent No. 8,056,075 in IPR2024-00643.

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

Lead Counsel
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

Backup 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	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
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	

Petitioner consents to electronic service by email in 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. 50-0639. Any overpayment or refund of fees may also be deposited in this Deposit Account.

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

Petitioner certifies that the '248 patent is available for IPR and that Petitioner is not barred or estopped from requesting IPR.

III. STATEMENT OF PRECISE RELIEF REQUEST AND REASONS THEREFORE

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

Ground	Basis	Relied-On References	Claim(s)
1	§103 ¹	U.S. Pat. No. 5,923,885 to Johnson et al. (“Johnson”) (Ex.1011)	1, 9
2	§103	U.S. Pat. No. 6,347,398 to Parthasarathy et al. (“Parthasarathy”) (Ex.1007) and Emin Gün Sirer, Sean McDirmid, & 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. 6,260,078 to Fowlow et al. (“Fowlow”) (Ex.1009) and Kimera	1-13

The Petition is accompanied by the declaration of Dr. Erez Zadok, an expert in the field of the '248 patent, who opines each of the Challenged Claims are obvious over the prior art. Ex.1002, ¶¶1-44, 202-204, 448.

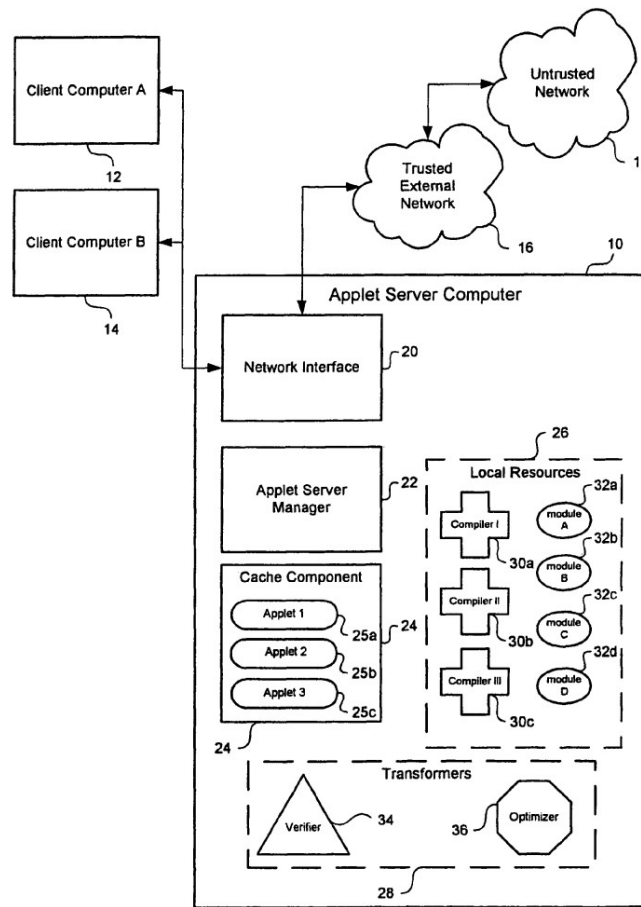
¹ See 89 FR 14449, 14449-14453 (“Updated Guidance for Making a Proper Determination of Obviousness,” dated Feb. 27, 2024).

IV. SUMMARY OF THE '248 PATENT

A. Background of the Technology

The '248 patent discloses server architecture that delivers applications requested by a client computer. Ex.1001, 2:20-34, Abstract. With such architecture, the client can request and execute code without having to verify or compile the code from local resources. *Id.* When a server receives a request for an application or applet² from a client, the server checks whether the requested form is available in a cache, and if it is not, will attempt to build the applet using resources local to the server by selecting and compiling program code modules into an applet. *Id.*, Abstract, 2:20-34, 3:55-4:20. The compiled applet is then transmitted to the requesting client computer. *Id.* Figure 1 of the '248 patent depicts an applet server computer 10 that sits between client computers 12, 14 and external networks 16, 18.

² The terms applet and application are used semi-interchangeably in the specification: “An applet (e.g. applets 1-3, 25a-25c) 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, 3:21-26.



Id., Fig. 1.

The process of compilation and transformation³ of program code, as well as transmission of that compiled and transformed code to a client, in response to a request from a client to a server was well-known before the earliest effective filing date of the '248 patent as shown in the references described herein. Ex.1002, ¶¶45-147.

³ "Verification, optimization and compression are examples of types of transformation operations." Ex.1001, 3:63-64.

B. Prosecution History

The '248 patent was rejected once, finding claims 1-13 unpatentable under the double-patenting doctrine and independent claims 1, 5, and 9 obvious over U.S. Patent No. 5,805,829 ("Cohen") in view of U.S. Patent No. 5,828,840 ("Cowan"). Ex.1004, 44-47. The applicant filed a terminal disclaimer, and argued Cohen failed to teach "formatting the request, in a 'form selected from a plurality of forms' as recited in independent claim 1." *Id.*, 64-67. According to the applicant, Cohen also "does not disclose the ability to compile an applet according to a form selected from a plurality of forms" and "simply transmits 'intermediate code' to the server." *Id.*, 66. Similarly, the applicant argued "Cohen does not transmit applets according to a selected format chosen from a plurality of formats." *Id.* "Cowan," according to the applicant "does not remedy the deficiencies identified in Cohen" and "simply discloses that the server may determine whether a client can execute Java applications directly (a network terminal) or not (non-network terminals)." *Id.*, 66-67.

In the grandchild patent application, U.S. Patent Application No. 11/933,161, the applicant originally filed a single claim, which is a verbatim copy of claim 1 of the '248 patent (*Compare* Ex.1022 *with* Ex.1001, 7:20-34) which the examiner rejected as anticipated by Johnson. Ex.1023, 3-6. Johnson was not considered during prosecution of the '248 patent and is analyzed below. *See* Ex.1024, 3, 6

(cancelling original claim 1, noting only that “Applicant...respectfully believes that Johnson does not read on any of Applicant’s new claims 2-16.”); Ex.1002, ¶¶157-162.

V. 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) citing *Vivid Techs., Inc. v. Am. Sci. & Eng’g, Inc.*, 200 F.3d 795, 803 (Fed. Cir. 1999). Petitioner reserves its right to respond to any unforeseeable claim constructions Patent Owner may advance.

VI. 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, ¶¶163-165.

B. Prior Art

1. Johnson (Ex.1011)

U.S. Pat. No. 5,923,885 to Johnson et al. (“Johnson”) was filed on October 31, 1996, issued on July 13, 1999, and is prior art under 35 U.S.C. § 102(e). Ex.1011, cover. Johnson teaches a method for installing and operating selected software applications on a client computer in communication with a server computer over a computer network. *Id.*, Abstract. Using the disclosed network, “users can select software from central sources and execute that software on their local machines as if that software was installed originally on those local machines.” *Id.*, 12:3-7, 3:15-25. Specifically, a user loads a web page on browser software, which is installed on the client computer. *Id.*, 8:15-27. In turn, the web page includes the selected software for transforming the browser interface. *Id.*, 8:24-32.

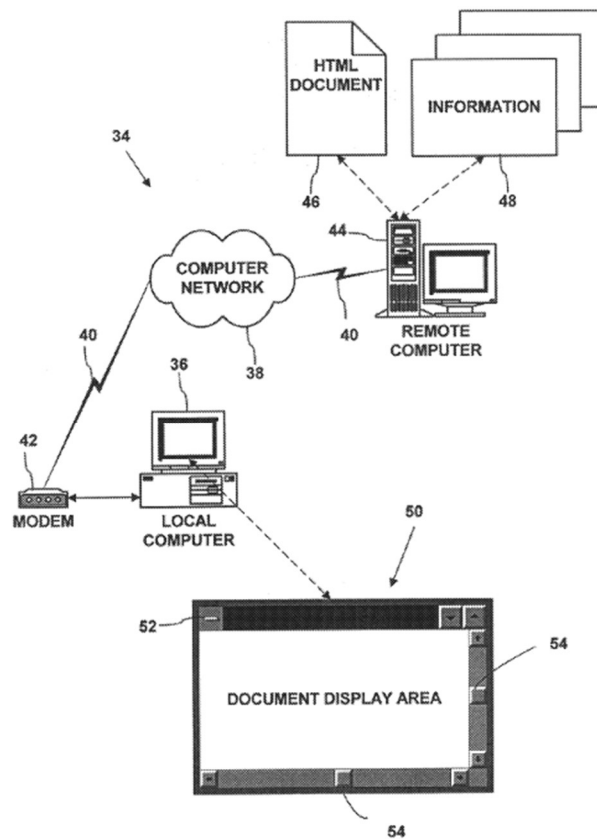
Johnson was not considered during the prosecution of the ’248 patent. However, during the prosecution of subsequently filed Application No. 11/933,161, the as-filed claim 1 was *identical* to challenged claim 1 of the ’248 patent. *Compare* Ex.1022 *with* Ex.1001, 7:20-34. The Patent Office rejected the subject matter of claim 1 of the ’248 patent as anticipated in view of Johnson. Ex.1023, 3-6. Applicant did not argue against this rejection, but rather replaced the then-pending

claim (that mirrored claim 1 of the '248 patent) with other claims that ultimately issued as U.S. Patent No. 8,056,075. Ex.1024, 6; Ex.1002, ¶¶166-178.

2. 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, the network browser, uses various protocols such as Hyper Text Transfer Protocol (“HTTP”) to access data on the World-Wide-Web (“Web”). *Id.*, 1:39-46. The Web contains documents created with Hyper Text Markup Language (“HTML”). *Id.*, 1:46-53.

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



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.*, 8:10-20. In one example, the network browser of the local computer requests to download a desired software component stored on the remote computer. *Id.* A software component download module manages the request based on whether the “desired software component is already

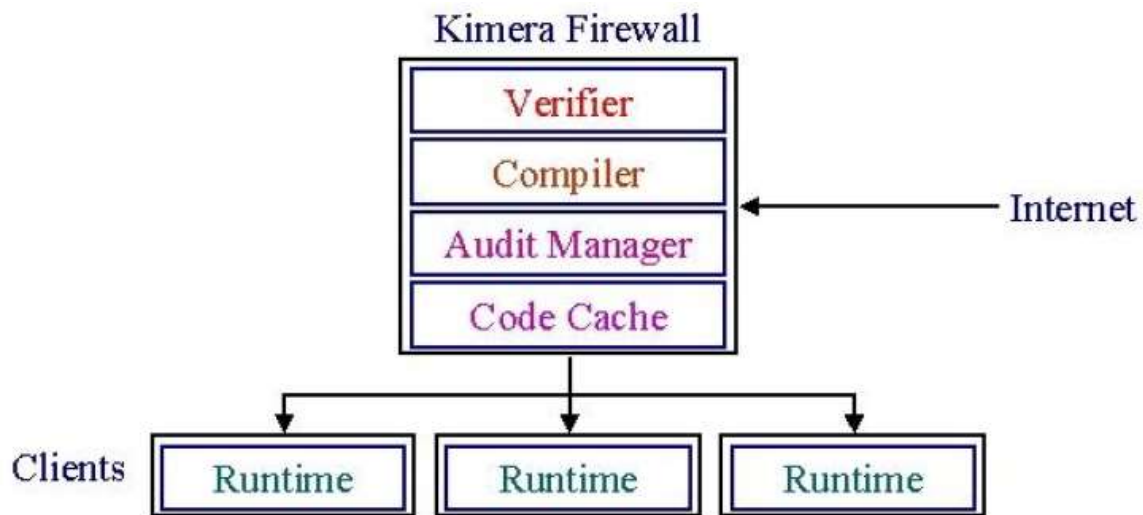
installed on the local computer 36” and whether “a more recent version [of the desired software component] is available.” *Id.*, 8:19-30; Ex.1002, ¶¶179-186.

3. Kimera (Ex.1008)

“Kimera: A System Architecture for Networked Computers” was published at least as early as January 15, 1998. Ex.1008, 1. Kimera qualifies as prior art to the ’248 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.* 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 endpoint.



Id.

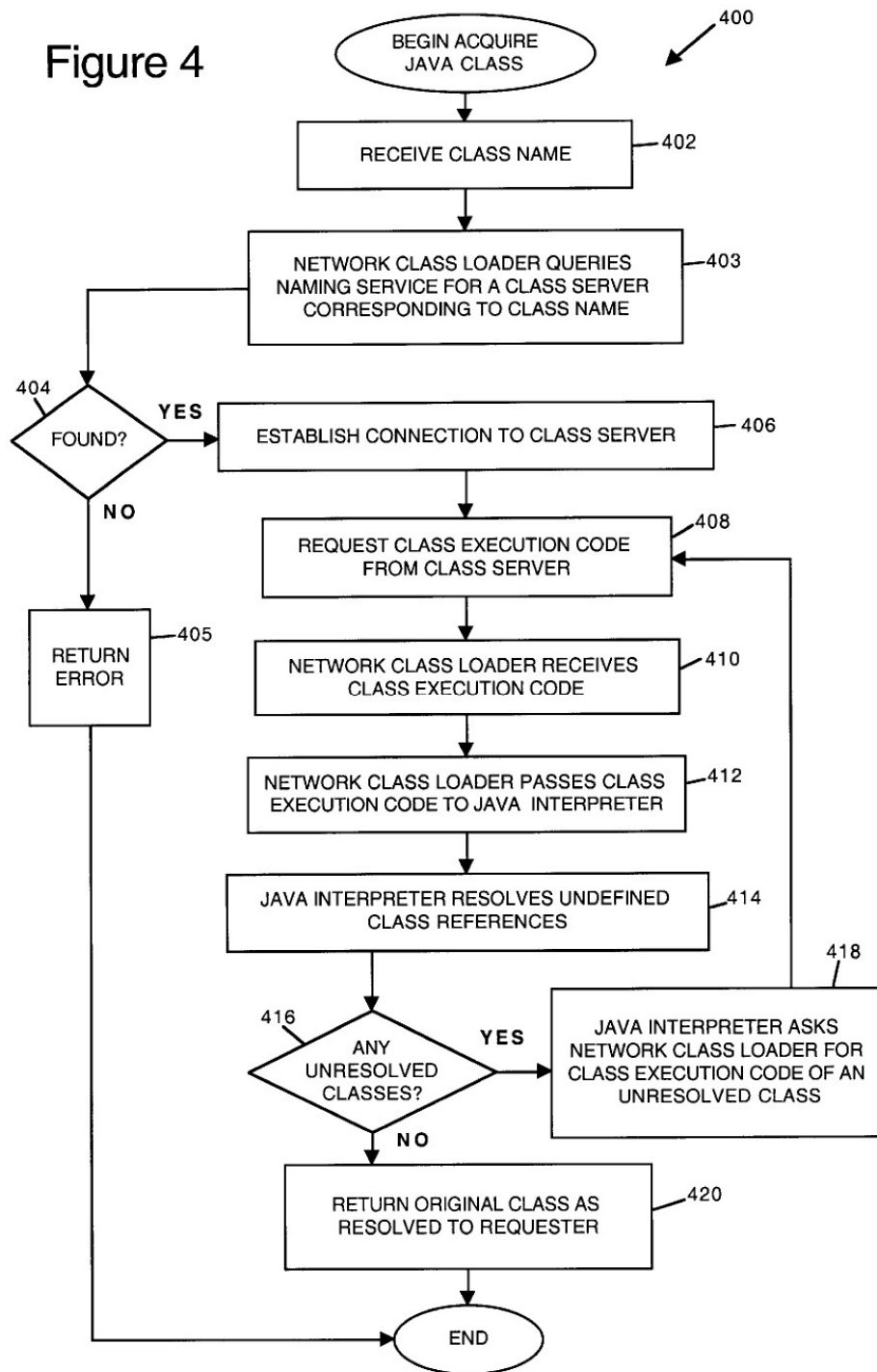
Kimera discusses “ahead of time” compilation, which a POSITA would have understood to simply mean compilation of client-requested data received into the centralized firewall as part of the fulfillment of the client request, as compared to “just-in-time” compiling at the client itself. *Id.*, 2 (noting 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, ¶192. In sum, Kimera’s architecture provides mandatory safety enforcement, simple security management, and efficient memory and processing of Java clients.

Ex.1008, 2; Ex.1002, ¶¶187-192.

4. Fowlow (Ex.1009)

U.S. Pat. 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 applets 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, ¶¶193-201.

C. Ground 1: Claims 1 and 9 are Unpatentable Under 35 U.S.C. §103(a) as Obvious over Johnson

Claims 1 and 9 are obvious over Johnson. Ex.1002, ¶205.

1. Independent Claims 1 and 9

- a. [Ipre] A method operating on a computer system for managing requests to a server computer for applets in a client server environment wherein each request for an applet specifies one form of the applet out of a plurality forms of the applet, comprising:*⁴

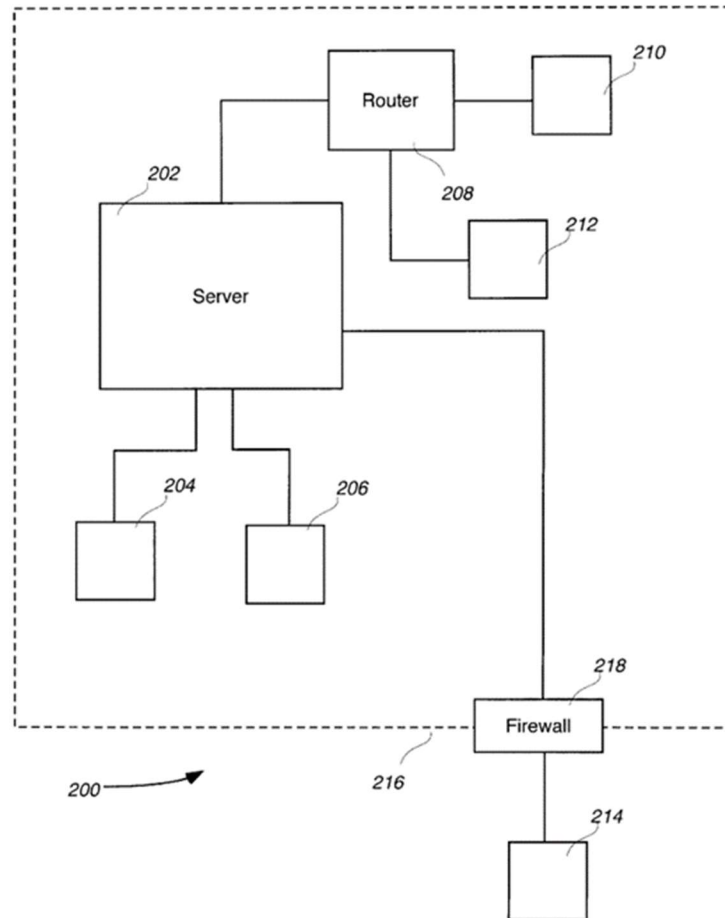
To the extent the preamble is limiting, it is taught by Johnson. Ex.1002, ¶207.

Johnson discloses a network “for installing and operating selected software applications on a client computer that is in communication with a server computer on a computer network.” Ex.1011, Abstract, 3:15-25, 6:6-9, 12:3-7. The USPTO already found Johnson to anticipate claim 1, which included this preamble. Ex.1022; EX.1023, 3-4.

Figure 2 illustrates an intranet having a server 202 coupled to client computers 204 and 206. Ex.1011, 5:57-6:5. The server 202 provides data and software, including applets compatible with the client computer, to the client computers 204 and 206. *Id.*, 5:57-60, 6:6-24 (“The term ‘applet’ as used herein refers to a software

⁴ The preamble of claim 9 is rendered obvious by Johnson for these same reasons. Ex.1002, ¶234.

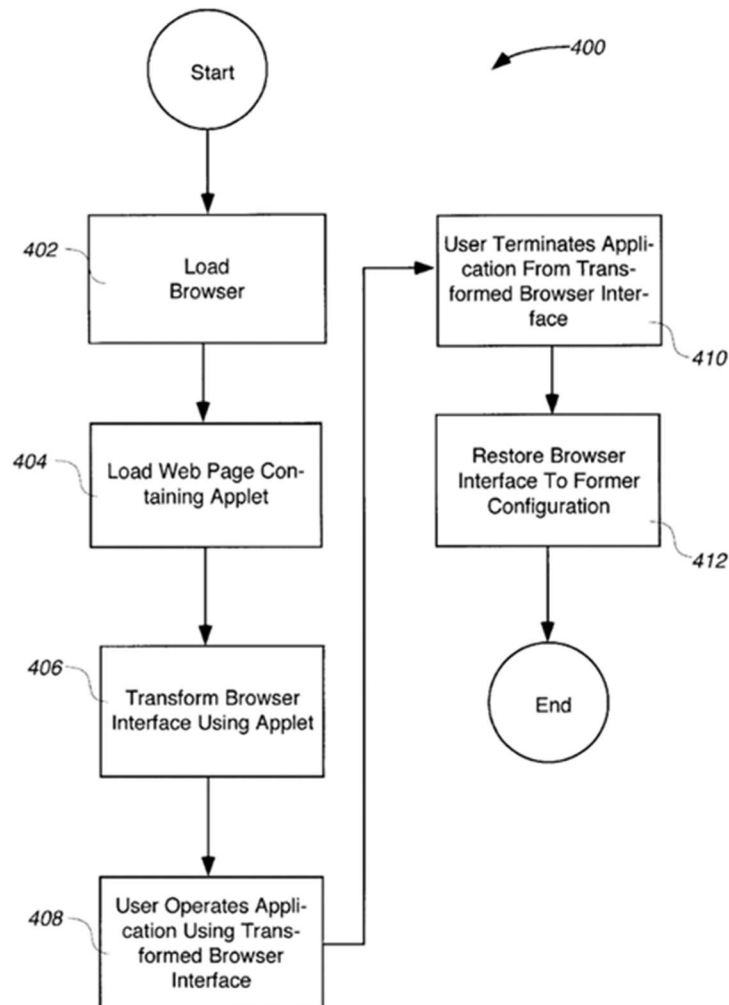
code that is executed in the context of software already running on the remote computer.”).



Id., Fig 2; Ex.1002, ¶¶208-210.

Using Johnson’s network, “users can select software from central sources and execute that software on their local machines as if that software was installed originally on those local machines.” Ex.1011, 12:3-14, 3:15-25. A user first loads browser software at step 402. *Id.*, 8:13-18, Fig. 4 (reproduced below). Once the browser software is loaded on the client computer, the user executes the browser software from the client. *Id.* “[T]he user then loads a web page which contains the

desired applet to be executed at step 404.” *Id.*, 8:24-27, Fig. 4. This disclosure demonstrates that Johnson manages requests (*i.e.*, user software selections) to a server computer for applets in a client server environment. Ex.1002, ¶210.



Ex.1011, Fig. 2.

When a user loads a web page, the browser (located on the client) loads the desired applet from the remote source. *Id.*, 8:24-27, 6:6-21. The applet, which is software code executed at the browser of the client, provides additional

functionalities that the browser itself is not configured to perform. *Id.*, 6:22-32; Ex.1002, ¶209.

Further, Johnson highlights “the need to support software that runs on different computer platforms, such as UNIX, Macintosh, and Windows,” recognizing that software for one client platform will not work on a different platform. Ex.1011, 2:40-54. Likewise, a POSITA would have known that applets for one browser—*e.g.*, Netscape Navigator, Microsoft Explorer, or JavaSoft HotJava—would not be compatible with another browser, and thus the request would specify the correct form of the applet configured to run on the requesting browser. *Id.*, 7:15-19; Ex.1002, ¶¶211-212. A POSITA would have understood that the request for the desired software (*i.e.*, requests to the server computer) specifies the form (*i.e.*, the type or version of the software needed to run on the requesting client and browser) of the software. *Id.*, ¶¶211-217.

To the extent the preamble is limiting, a POSITA would have found it obvious in view of Johnson. *Id.*, ¶218; Ex.1023, 3-4.

- b. *[1a] a) receiving on said server computer a request from a client computer for an applet in a form selected from a plurality forms;*⁵

This limitation would have been obvious over Johnson. Ex.1002, ¶¶219-221. The USPTO already found Johnson discloses this element. Ex.1023, 4-5. Johnson discloses a network “for installing and operating selected software applications on a client computer that is in communication with a server computer on a computer network.” Ex.1011, Abstract.

The intranet includes a server 202 coupled to client computers 204 and 206. Ex.1011, 5:57-60, Fig. 2. “[T]he server and clients are configured to exchange not only data but computer software applications and, more particularly, software in the form of ‘applets.’” *Id.*, 6:17-24. Using Johnson’s network, “users [at clients] can *select* software from central sources [servers] and execute that software on their local machines.” *Id.*, 12:3-7 (emphasis added)⁶, 3:15-25. When a user selects a desired applet, the client computer sends a request to the server for the specified applet, and the server receives that request. *Id.*, 12:3-14; 3:15-25, 6:6-24; Ex.1002, ¶¶219-220.

⁵ Element [9a] is rendered obvious by Johnson for these same reasons. Ex.1002, ¶235.

⁶ All emphasis herein is added unless stated otherwise.

Further, as explained in Section VI.C.1.a, the user request specifies the form of the applet from of a plurality of forms, namely the form of the applet compatible with the requesting client and browser. Ex.1011, 2:40-54, 7:15-19; Ex.1002, ¶¶211-217.

Johnson renders obvious this limitation. Ex.1002, ¶221; Ex.1023, 4-5.

- c. [1b] b) compiling said applet into said selected form from a local resource comprising at least one source module and one compiler which acts on said source module to produce said selected form; and⁷*

Johnson discloses compiling the selected applet into a selected form using local resources. Ex.1002, ¶¶222-223. The USPTO already found Johnson discloses

⁷ A POSITA would have understood that “an intermediate form program module,” as recited in limitation [9b], to be rendered obvious by platform-independent software, such as Java applets, which Johnson teaches is a possible selection. Ex.1011, 6:42-49 (“Java applets or an applet written in the Java programming language, can function in a platform-independent manner”), 12:66-13:3; Ex.1002, ¶¶236-238; *see also* Ex.1001, 3:38-39 (“The intermediate byte-code need not necessarily be Java byte-code.”), 2:43-46). Thus, compiling an applet into an intermediate form, as recited in element [9b], is similarly rendered obvious by Johnson. Ex.1002, ¶239.

this element. Ex.1023, 5-6. In Johnson, users select software that is then executed on the local machine. Ex.1011, 12:3-7, 3:15-25. “[T]he transfer of the selected software from the remote computer is effective to transmute the user interface of the browser into the user interface of the software application.” *Id.*, 4:28-34. A POSITA would have understood this transmuting (or transforming) code is compiling from one form to another, namely, from source code to executable code. *Id.*; Ex.1002, ¶224.

Johnson further teaches the client can either directly execute software, or it can “respond more like a terminal than a stand-alone computing platform.” Ex.1011, 8:16-23. When the client directly executes software, a POSITA would have understood the server would ensure the correct version of the applet was sent to the requesting client. A POSITA would have found it obvious to compile source code into browser-specific, executable code at the server so the browser could execute that executable code directly. *Id.* (“In one embodiment, the browser software is installed on the local client machine and is executed by the user directly from the client.”); Ex.1002, ¶¶225-227. Alternatively, when the client acts like a terminal, a POSITA would have understood that the server compiles applets into a server-compatible form that is executable on the server and *executes* those applets using resources local to the server. Ex.1011, 8:16-23; Ex.1002, ¶226.

Johnson renders obvious this limitation. Ex.1002, ¶228; Ex.1023, 5-6.

*d. [1c] c) transmitting said applet in said selected form to said client computer.*⁸

Johnson discloses transmitting the applet in the selected form to the client computer. Ex.1002, ¶¶229-230. The USPTO has already found Johnson to disclose this element. Ex.1023, 6-7.

Johnson discloses “[m]ethods, systems, and software for installing and operating selected software applications on a client computer that is in communication with a server computer on a computer network....” Ex.1011, Abstract. This includes a “server [that] provides data and software that is accessible to the various clients which are in communication with the server, either directly or through a device such as a router.” *Id.*, 6:6-24. “In particular, however, the server 208 will be configured to provide data that is compatible with browser software” including software applications and applets. *Id.*

Johnson renders obvious this limitation. Ex.1002, ¶231; Ex.1023, 6-7.

Johnson, therefore, renders obvious claim 1. Ex.1002, ¶232.⁹

⁸ Element [9c] is rendered obvious by Johnson for these same reasons. Ex.1002, ¶¶241-242.

⁹ Claim 9 is similarly rendered obvious Johnson. Ex.1002, ¶¶233-242.

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

Claims 1-13 are obvious over Parthasarathy in view of Kimera. Ex.1002, ¶243.

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, ¶¶244-246. 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 ’248 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, ¶247. A POSITA would have recognized that the inefficiencies arising from this overhead taking place on various local computers, which would have led a POSITA to explore the benefits of centralizing the module to increase efficiency. Ex.1002, ¶¶247-249; *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, ¶250. 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, ¶¶249-250. 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, ¶251. 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, ¶¶252-258.

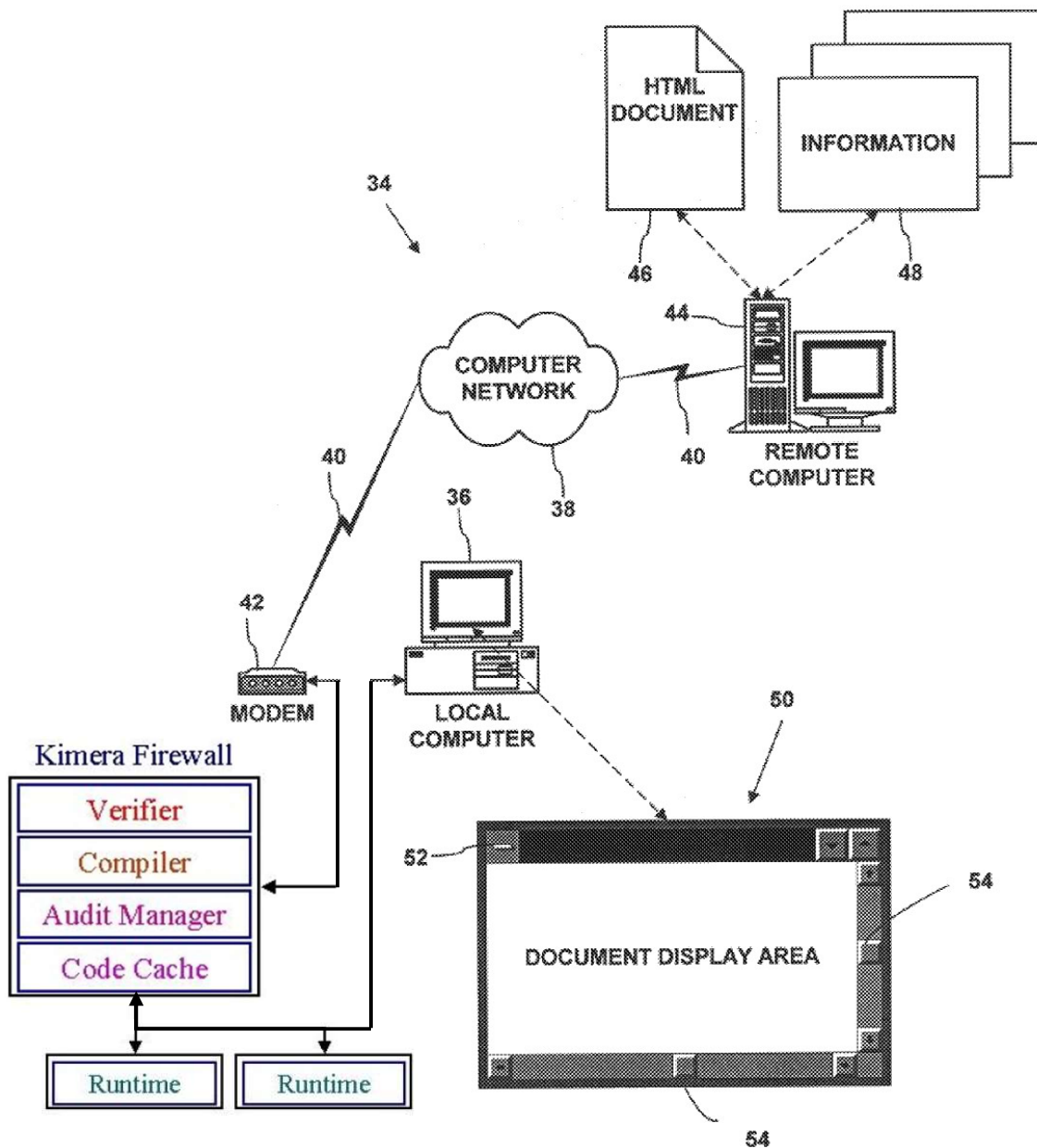


FIG. 2

Ex.1007, Fig. 2 (annotated); Ex.1008, 1; Ex.1002, ¶251.

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, ¶¶252-255.

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, ¶256. 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 addressing concerns raised in both references. *Id.*, ¶¶252-257. 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. It would also improve speed by holding processed software components in cache memory. *Id.*, ¶¶246, 253-254. Further, this combination would allow a POSITA to eliminate a JIT compiler, verifier, and interpreter from the local computers, if desired, thereby making the system cheaper. *Id.*, ¶257; Ex.1008, 2.

A POSITA would have been motivated to combine Parthasarathy and Kimera to arrive at the claimed method. Ex.1002, ¶¶244-258.

2. Independent Claim 1

- a. *[Ipre] A method operating on a computer system for managing requests to a server computer for applets in a client server environment wherein each request for an applet specifies one form of the applet out of a plurality forms of the applet, comprising:*

To the extent the preamble is limiting, Parthasarathy-Kimera teaches it. Parthasarathy discloses “network browsing environment” that operates to manage requests for applets in a client-server environment. Ex.1007, 8:19-31. This “network browsing environment” includes a local, client and remote, server computer. *Id.*, 6:44-50, 7:3-8, Fig. 2; Ex.1002, ¶¶259-261.

In one example, “a request is made [from the local, client computer] to download a desired software component stored on a remote computer...” Ex.1007, 8:19-20. The “network browsing environment” manages this request to the server computer based on whether the “desired software component is already installed on the local computer.” *Id.*, 8:19-23. If the desired software component is not locally available, it is downloaded from a remote source. *Id.*, 8:26-31.

As detailed in Section VI.D.1, a POSITA would have been motivated to modify Parthasarathy’s networking environment to centralize software locating, downloading, and verifying functionality as taught by Kimera. Ex.1002, ¶¶249-250, 261-262. 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, 1. The combined system would include Kimera’s centralized server located between the local computer and Internet interface, as depicted below:

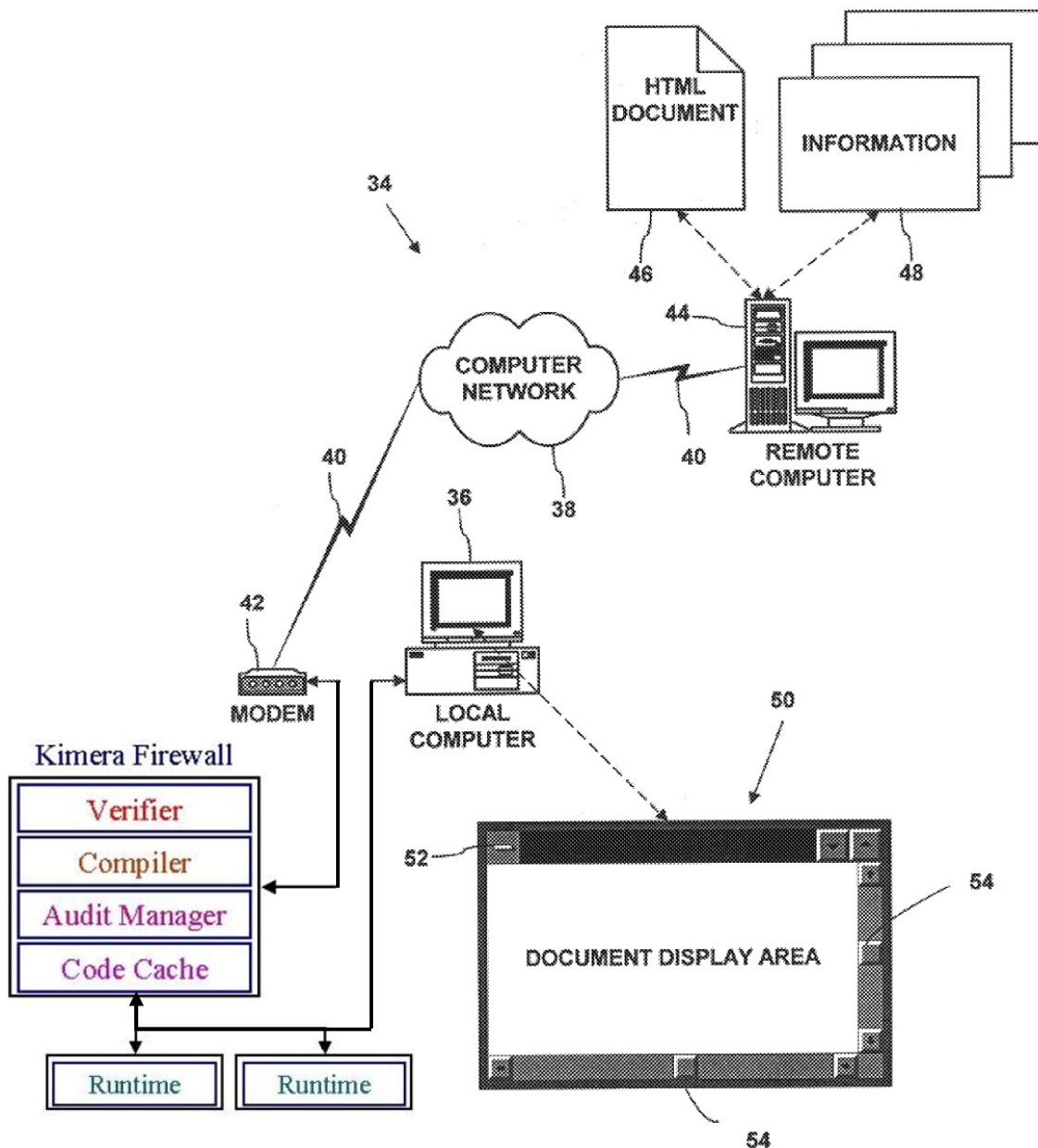


FIG. 2

Ex.1007, Fig. 2 (annotated); Ex.1008, 1; Ex.1002, ¶251, 261-264.

Parthasarathy further teaches that the request for a software component (*i.e.*, applet) specifies the form of the applet from a plurality of forms. Ex.1002, ¶265. For example, Parthasarathy provides two separate mechanisms to achieve platform independence: using initialization files and HTTP headers. Ex.1007, 17:61-67, 18:64-19:3. A POSITA would have understood that platform independence was achieved by storing a plurality of forms of an applet corresponding to a plurality of operating systems and client computer hardware parameters and selecting the appropriate version to send to the requesting client. *See id.*, 18:35-57; *see also* Ex.1008, 2 (noting that Java bytecode is compiled “into native machine code at the centralized server” and that recent compilations are stored in cache memory at the centralized server); Ex.1002, ¶¶265-270.

When a client computer requests a software component, the request uses a designator, such as a URL or a network search path to locate a control file, which in turn is used to locate the requested software component in memory. Ex.1007, 9:9-21. The control file may be an initialization file, which includes instructions to install the correct files for the client system via “the %opersys% and %cpu% attribute values [that] will *specify the operating system and CPU* for the desired platform desired software components will be executed on.” Ex.1007, 17:61-67, 18:35-57; Ex.1002, ¶¶266-268. By specifying the operating system and CPU of the system that will execute the software component, a POSITA would have understood

that the request identifies an application (*i.e.*, the software component) and identifies a form of the application (*i.e.*, a form of the code compatible with the “desired platform”) from a plurality of forms (*i.e.*, software components specific to for various operating system/CPU combinations), such as win32, mac, and other operating systems using x86 or PowerPC CPU architectures.

Valid values for %opersys%	Meaning
win32	32-bit Windows ® operating systems (Windows95 or Windows NT)
mac	Macintosh ® operating system
<other>	will be defined as necessary
Valid values for %cpu%	Meaning
x86	Intel ® x86 family of processors
ppc	Motorola ® PowerPC architecture
mips	MIPS ® architecture processors
alpha	DEC ® Alpha architecture

Ex.1007, 18:45-57; Ex.1002, ¶268.

In another example, an HTTP ACCEPT header MIME request specifies the form of the software. *Id.*, 18:64-19:3 (“The MIME request type *specifies which platform the code is to run on*, thus allowing platform independence when the CODEBASE attribute or the network search path is use.”). A POSITA would have understood that the HTTP request specifies a platform-specific applet from a plurality of applets. Ex.1002, ¶269.

Finally, both requests for software components by control file and HTTP request would both route through Kimera’s centralized server in order to determine

whether the code cache already stored the desired software component. *See* Ex.1007, 8:20-24 (noting that before requests are sent to the remote computer, the system checks to see whether the software is already installed locally); Ex.1008, 2 (noting “results of recent compilations” are cached in the server’s code cache); Ex.1002, ¶¶265-270. The centralized server would receive the request and serve the specified applet from a plurality of applets if that applet was stored in the code cache. Ex.1007, 8:20-24; Ex.1008, 2. If the specific applet was not available, or a more recent version as available on the Web, the request would be passed on to the remote computer for fulfillment. *See* Ex.1007, 8:26-30; Ex.1002, ¶¶264, 270.

To the extent the preamble is limiting, a POSITA would have found it taught by Parthasarathy-Kimera. Ex.1002, ¶271.

- b. [1a] a) receiving on said server computer a request from a client computer for an applet in a form selected from a plurality forms;*

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 in Section VI.D.1, 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, ¶¶272-274.

Parthasarathy further teaches that locating the desired software utilizes an initialization file or HTTP header that “*specifies the operating system and CPU* for the desired platform desired software components will be executed on.” Ex.1007, 17:45-67, 18:35-57, 18:64-19:3. In this way, requests for software components specify a form of the applet from a plurality of forms (*e.g.*, the same software for other operating systems or CPU architectures). *Id.*, 18:15-57, 18:64-19:3. A POSITA would have understood that the request received at the server (*i.e.*, Kimera’s centralized server) identifies an application (*i.e.*, the software component) and identifies a form of the application (*i.e.*, a form of the code compatible with the “desired platform”) selected from a plurality of forms of the application (*i.e.*, versions of the code for other operating systems or CPUs). Ex.1002, ¶¶275-276.

Parthasarathy-Kimera, therefore, teaches this limitation. Ex.1002, ¶277.

- c. [1b] b) *compiling said applet into said selected form from a local resource comprising at least one source module and one compiler which acts on said source module to produce said selected form; and*

Kimera discloses a discrete compiler located at the centralized server, noting “[c]ompilation from Java bytecode into native machine code is another service that

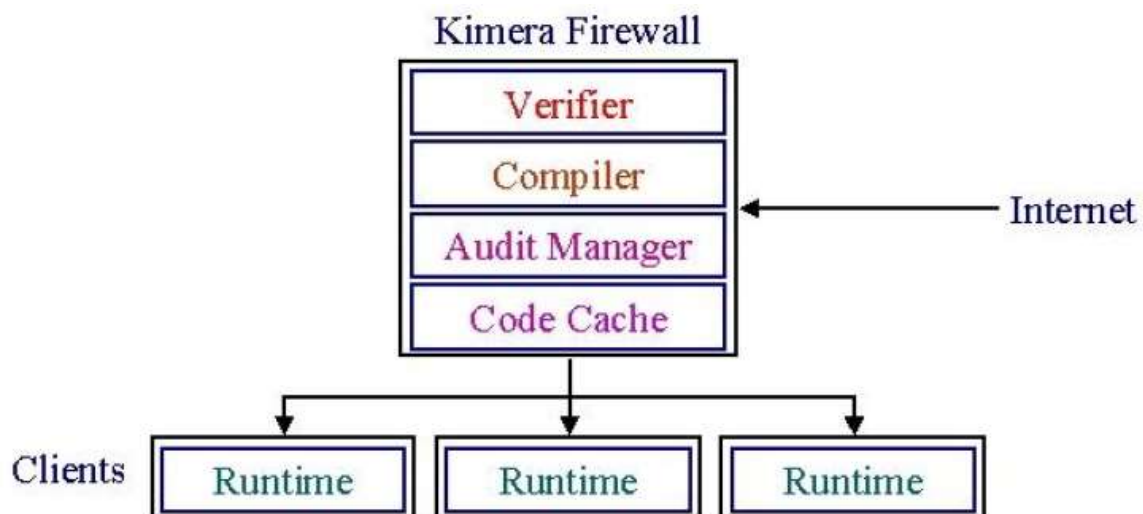
is best performed at a central location.” Ex.1008, 1, 2. Kimera teaches “results of recent compilations” are cached in the central server “to further reduce the latency of binary translation.” *Id.*, 2. A POSITA would have understood that the compiler of Kimera compiles source module (*e.g.*, Java bytecode and other cached recent compiled modules) to produce the selected form (*i.e.*, native machine code). *Id.*; *see also* Ex.1007, 18:35-57 (listing potential values for the %opersys% and %cpu% attributes that are used to “*specify the operating system and CPU* for the desired platform desired software components will be executed on.”); Ex.1002, ¶¶266-268, 278-280.

This disclosure comports with the disclosure of the ’248 patent, which explains that “[l]ocal resources 26 are comprised of program modules 32 (applets in source form, not the requested form) and compilers 30.” Ex.1001, 4:46-48, 5:23-26 (“A target compiler 52 compiles intermediate applet code 44 into an applet 54 in a processor specific format (binary) suitable for running natively on a given computer platform.”). A POSITA would have been motivated to combine Parthasarathy and Kimera for the reasons enumerated above. *See* Section VI.D.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, ¶281 VI.D.2.a. Parthasarathy renders obvious this limitation. Ex.1002, ¶282.

d. [1c] c) transmitting said applet in said selected form to said client computer.

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. The returned software component would be the specific form needed on the requesting client computer. *Id.*, 17:45-67, 18:35-57, 18:64-19:3; Ex.1002, ¶¶283-284.

In the context of the combination with Kimera, the software would be sent to the client from the centralized server, as depicted by the arrows from the Kimera Firewall to the client computers.



Ex.1008, 1; Ex.1002, ¶283.

Parthasarathy-Kimera, therefore, teaches this limitation. Ex.1002, ¶285.

Thus, Parthasarathy-Kimera renders obvious claim 1. Ex.1002, ¶286.

3. Dependent Claims 2-4

- a. *2. The method of claim 1, further comprising the step of: copying said applet in said selected form to a local cache after compiling said applet from said local resource if said cache does not contain a copy of said applet in said selected form.*

Parthasarathy discloses a permanent cache store for downloaded code, along with a registry of downloaded code as a shortcut to determine whether a file needs to be overwritten because it is an out-of-date version. Ex.1007, 27:15-27. In other words, the registry is used to determine whether the downloaded code is already in the permanent cache (*i.e.*, local cache) of the client computer. *Id.*; Ex.1002, ¶¶287-288. If the downloaded applet is not stored in cache memory, Kimera teaches that the centralized server will compile the downloaded code and then “cach[e] the results of *recent compilations* in the central compilation server.” Ex.1008, 2.

A POSITA would have understood “caching the results of recent compilations” teaches copying the selected applet to a local cache. Ex.1002, ¶¶289-290. This claim is obvious in view of Parthasarathy and Kimera. Ex.1002, ¶291.

- b. 3. *The method of claim 2, further comprising the step of: transmitting a request to an external resource for said applet in said selected form if a copy of said applet is not stored in said local cache and said applet can not be compiled from said local resource, and directing said external resource to transmit said applet in said selected form to said server computer.*

Parthasarathy manages requests for software depending on whether a copy of the selected form is stored in local memory. Ex.1007, 8:19-30, 13:42-45. “If the desired software component is not registered and installed on the local computer, or if the version number is incorrect, then the desired software component is obtained from the remote computer 44.” *Id.*, 13:45-48, 8:26-30. A POSITA would have understood that the code cache of the local, centralized server would also be checked for the desired software component before the request was directed to the remote computer. Ex.1002, ¶¶292-294.

If the selected form of the desired software component was stored in the centralized code cache, a POSITA would have understood that the software component would be returned to the client from cache memory. Ex.1002, ¶294; *see also* Ex.1008, 2 (noting caching is used “to further **reduce the latency** of binary translation”). If the code is not stored in Kimera’s code cache, “then the desired software component is obtained from the remote computer 44.” Ex.1007, 13:45-48, 8:26-30. To obtain the desired form of the software component, the request is

transmitted to the remote computer in order to locate the software on the remote computer. *See id.*, 9:9-21; Ex.1002, ¶¶295-296.

The process of locating software on the remote computer utilizes a “control file including 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. Both the remote computer, and the other remote computers that the remote computer can rely on, are external resources that can transmit the applet in the selected form to the centralized server and client. Ex.1002, ¶¶297-300; *see also* Ex.1007, 7:15-26, 1:48-51.

This claim is obvious in view of Parthasarathy and Kimera. Ex.1002, ¶301.

- c. 4. *The method of claim 1, further comprising the step of: transmitting a request to an external resource for supplying said applet in said selected form if said applet can not be compiled from said local resource and directing said external resource to transmit said applet in said selected form to said server computer.*

Parthasarathy-Kimera renders this claim obvious as described in Section VI.D.3.b. Ex.1007, 1:48-51, 7:15-26, 8:19-30, 9:9-21, 13:42-48, 32:40-44; Ex.1008, 2; Ex.1002, ¶¶292-302. A POSITA would have understood the compiler on Kimera’s centralized server would compile cached applets, if necessary, to fulfill client computer requests for applets. Ex.1002, ¶303. If the compiler could not

compile the applet from code stored in cache memory, the centralized server would seek the necessary code from external resources as taught by Parthasarathy. *See* Ex.1007, 32:40-44 (noting software components can be stored “on one or more other remote computers”); *see also* Ex.1002, ¶303.

Thus, this claim is obvious in view of Parthasarathy-Kimera. Ex.1002, ¶304.

4. Independent Claim 5

- a. *[5pre] A method operating on a computer system for managing requests to a server computer for applets in a client server environment wherein each request for an applet specifies one form of said applet out of a plurality of forms of said applet, comprising:*

To the extent the preamble is limiting, Parthasarathy-Kimera renders it obvious as discussed in Section VI.D.2.a. Ex.1007, 6:44-50, 7:3-8, 8:19-34, 9:9-21, 17:61-67, 18:35-57; 18:64-19:3, Fig. 2; Ex.1008, 1, 2; Ex.1002, ¶¶305, 259-271.

- b. *[5a] a) receiving on said server computer a request from a client computer for an applet in a specified form selected from a plurality of forms;*

Parthasarathy-Kimera renders this limitation obvious as discussed in Section VI.D.2.b. Ex.1007, 2:65-3:21, 8:20-24, 8:66-9:8, 9:33-42, 17:45-67, 18:15-57, 18:64-19:3, Fig. 4; Ex.1008, 1-2; Ex.1002, ¶¶306, 272-277.

- c. [5b] b) determining whether said applet is stored in said specified form in a local cache and, if so, transmitting said applet in said specified form to said client computer;*

Parthasarathy-Kimera renders this limitation obvious as discussed in Section VI.D.3.b. Ex.1007, 1:48-51, 7:15-26, 8:19-30, 9:9-21, 13:42-48, 32:40-44; Ex.1008, 2; Ex.1002, ¶¶307, 292-301. Specifically, Parthasarathy teaches managing requests for software depending on whether a copy of the selected form is stored in local memory. Ex.1007, 8:19-30, 13:42-48. This would include checking both the client computer's memory, and the centralized server's code cache. Ex.1002, ¶¶308-309.

If the selected form of the desired software component was stored in the centralized server's code cache, a POSITA would have understood that the software component would be returned to the client from cache memory. Ex.1002, ¶309; *see also* Ex.1008, 2 (noting caching is used “to further **reduce the latency** of binary translation”).

Parthasarathy-Kimera, therefore, teaches this limitation. Ex.1002, ¶310.

- d. [5c] c) if said applet is not stored in said selected form in said local cache,*

As discussed in Section VI.D.3.b, Parthasarathy-Kimera teaches managing requests for software depending on whether a copy of the selected form is stored in local memory or cache memory. Ex.1007, 8:19-30, 13:42-45; Ex.1008, 2; Ex.1002, ¶¶311, 297-298. “If the desired software component is not registered and installed

on the local computer, or if the version number is incorrect, then the desired software component is obtained from the remote computer 44.” Ex.1007, 13:45-48, 8:26-30; Ex.1002, ¶312.

Parthasarathy-Kimera, therefore, teaches this limitation. Ex.1002, ¶313.

- e. [5d] compiling said applet into said selected form from a local resource comprising at least one source module and one compiler which acts on said source module to produce said selected form and*

Parthasarathy-Kimera renders this limitation obvious as discussed in Section VI.D.2.c. Ex.1007, 18:35-57; Ex.1008, 1, 2; Ex.1002, ¶¶314, 278-282.

- f. [5e] transmitting said applet in said selected form to said client computer.*

Parthasarathy-Kimera renders this limitation obvious as discussed in Section VI.D.2.d. Ex.1007, 8:19-20, 17:45-67, 18:35-57, 18:64-19:3, 29:13-20; Ex.1008, 1; Ex.1002, ¶¶315, 283-285.

Thus, Parthasarathy-Kimera renders obvious claim 5. Ex.1002, ¶316.

5. Dependent Claims 6-8

- a. 6. The method of claim 5, further comprising the step of: copying said applet in said selected form to said cache after compiling said applet from said local resource if said cache does not contain a copy of said applet in said form.*

Parthasarathy-Kimera renders this claim obvious as described in Section VI.D.3.a. Ex.1007, 27:15-27; Ex.1008, 2; Ex.1002, ¶¶317, 287-291.

- b. 7. The method of claim 6, further comprising the step of: transmitting a request to an external resource for said applet in said selected form if a copy of said applet is not stored in said local cache and said applet can not be compiled from said local resource, and directing said external resource to transmit said applet in said form to said server computer.*

Parthasarathy-Kimera renders this claim obvious as described in Section VI.D.3.b. Ex.1007, 1:48-51, 7:15-26, 8:19-30, 9:9-21, 13:42-48, 32:40-44; Ex.1008, 2; Ex.1002, ¶¶318, 292-301.

- c. 8. The method of claim 5, further comprising the step of: transmitting a request to an external resource for said applet in said selected form if a copy of said applet is not stored in said local cache and said applet can not be compiled from said local resources, and directing said external resource to transmit said applet in said selected form to said server computer.*

Parthasarathy-Kimera renders this claim obvious as described in Section VI.D.3.c. Ex.1007, 1:48-51, 7:15-26, 8:19-30, 9:9-21, 13:42-48, 32:40-44; Ex.1008, 2; Ex.1002, ¶¶319, 292-304.

6. Independent Claim 9

- a. [9pre] A method operating on a computer system for generating an applet in response to a request by a client computer wherein each request for an applet specifies one form of the applet out of a plurality forms of the applet, comprising:*

To the extent the preamble is limiting, Parthasarathy-Kimera renders it obvious as discussed in Section VI.D.2.a. Ex.1007, 6:44-50, 7:3-8, 8:19-34, 9:9-21, 17:61-67, 18:35-57; 18:64-19:3, Fig. 2; Ex.1008, 1, 2; Ex.1002, ¶¶320, 259-271.

- b. [9a] a) receiving on a server computer a request from a client computer for an applet in a form selected from a plurality forms;*

Parthasarathy-Kimera renders this limitation obvious as discussed in Section VI.D.2.b. Ex.1007, 2:65-3:21, 8:20-24, 8:66-9:8, 9:33-42, 17:45-67, 18:15-57, 18:64-19:3, Fig. 4; Ex.1008, 1-2; Ex.1002, ¶¶321, 272-277.

- c. [9b] b) compiling an applet program module into an intermediate form program module;*

Parthasarathy-Kimera renders this limitation obvious as discussed in Section VI.D.2.c. Ex.1007, 18:35-57; Ex.1008, 1, 2; Ex.1002, ¶¶322, 278-282. A POSITA would have understood that “an intermediate form program module” would include Java bytecode, which Kimera expressly teaches compiling into native machine code at the centralized server. Ex.1008, 2; Ex.1002, ¶323; *see also* Ex.1001, 3:38-39 (“The intermediate byte-code need not necessarily be Java byte-code.”), 2:43-46. In

addition to Java bytecode, other intermediate forms of code would have been known to a POSITA. Ex.1002, ¶¶323-324.

Parthasarathy-Kimera renders obvious this limitation. Ex.1002, ¶325.

d. [9c] c) transmitting said applet in said selected form to said client computer.

Parthasarathy-Kimera renders this limitation obvious as discussed in Section VI.D.2.d. Ex.1007, 8:19-20, 17:45-67, 18:35-57, 18:64-19:3, 29:13-20; Ex.1008, 1; Ex.1002, ¶¶326, 283-286.

Thus, Parthasarathy-Kimera renders obvious claim 9. Ex.1002, ¶327.

7. Dependent Claims 10-13

a. 10. The method of claim 9, further comprising the step of: transforming said intermediate form program module into a transformed intermediate form program module with at least one transformer program.

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, ¶¶328-330. The centralized server

would also apply a tamper-proof signature after the code had been verified. Ex.1008, 2. A POSITA would have understood the verified and signed code would be a transformed intermediate program module, ready to be returned to the client device. *Id.*; Ex.1002, ¶¶330-331.

Parthasarathy also discloses a “cabinet file [which] contains one or more files, all of which are downloaded together in a compressed ‘cabinet.’” Ex.1007, 15:66-16:9. Transforming an intermediate form program module via compression was, therefore, well known. Ex.1002, ¶¶332-333. 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, ¶333.

The Parthasarathy-Kimera system also teaches the transforming intermediate form program modules by optimizing those programs. 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 at the server in order to achieve faster services performed on the most inexpensive clients possible. Ex.1002, ¶334.

Kimera further teaches that “[c]ompilation from Java bytecode into native machine code is another service that is best performed at a central location.” Ex.1008, 2. A POSITA would have understood that the compiler is a transformer program that acts on Java bytecode to produce a transformed intermediate program module, such as native machine code. *Id.*; Ex.1002, ¶335.

Finally, a POSITA would have been familiar with a wide range of multi-stage transformer operations for a proxy device that services a variety of operating systems and architectures. Ex.1002, ¶¶335, 136-147. This knowledge would have further rendered claim 10 obvious. *Id.*

Parthasarathy-Kimera, therefore, renders this claim obvious. Ex.1002, ¶336.

- b. 11. The method of claim 10, wherein said at least one transformer program is selected from the group consisting of verifying computer programs, optimizing computer programs, compressing computer programs, debugging computer programs, usage monitoring computer programs and encrypting computer programs.*

Parthasarathy-Kimera renders this claim obvious as described in Section VI.D.7.a. Ex.1007, Abstract, 1:11-16, 2:65-67, 15:66-16:9; Ex.1008, 1, 2; Ex.1002, ¶¶328-336, 119, 134-147. Specifically, Parthasarathy-Kimera teaches and suggests

various programs that transform the intermediate program modules, including the following: a “verification service” that verifies programs (Ex.1008, 2); optimization programs to take advantage of reduced computation and memory requirements in client computers (*Id.*, 1, 2); compression programs, including Lempel-Ziv compression (Ex.1007,15:66-16:9), and encryption computer programs, including the application of cryptographically strong hash functions such as MD5 (Ex.1008, 2). Ex.1002, ¶¶338-342, 136-147.

Any one of the transformer programs taught in Parthasarathy-Kimera are sufficient to find this claim obvious over the prior art. *Fresenius USA, Inc. v. Baxter Int’l, Inc.*, 582 F.3d 1288, 1298 (Fed. Cir. 2009) (“Element (a) is written in Markush form, such that the entire element is disclosed by the prior art if one alternative in the Markush group is in the prior art.”).

Parthasarathy-Kimera, therefore, renders this claim obvious. Ex.1002, ¶343.

- c. 12. The method of claim 11, further comprising the step of: compiling said transformed intermediate form program module into machine specific binary code with a target compiler.*

As discussed in Section VI.D.2.a, Parthasarathy discloses client requests that specify the form of application based on the operating system and CPU. Ex.1007, 17:61-67, 18:35-19:3; Ex.1002, ¶¶343, 266-268. Once requested, a POSITA would have understood that the compiler of Kimera is a target compiler because it would

transform the code based on the request of Parthasarathy into machine-specific binary code. Ex.1008, 2; Ex.1002, ¶¶344-345; *see also* Ex.1001, 5:11-26 (“A target compiler 52 compiles intermediate applet code 44 into an applet 54 in a processor specific format (binary) suitable for running natively on a given computing platform.”). Kimera expressly explains that “[c]ompilation from Java bytecode *into native machine code* is another service that is best performed at a central location.” Ex.1008, 2.

Parthasarathy-Kimera, therefore, renders this claim obvious. Ex.1002, ¶346.

- d. 13. The method of claim 10, further comprising the step of: compiling said transformed intermediate form program module into machine specific binary code with a target compiler.*

Parthasarathy-Kimera renders this claim obvious as described in Section VI.D.7.c. Ex.1007, 17:61-67, 18:35-19:3; Ex.1008, 2; Ex.1001, 5:11-26; Ex.1002, ¶¶344-347.

E. Ground 3: Claims 1-13 are Unpatentable Under 35 U.S.C. §103(a) as Obvious over Fowlow in view of Kimera

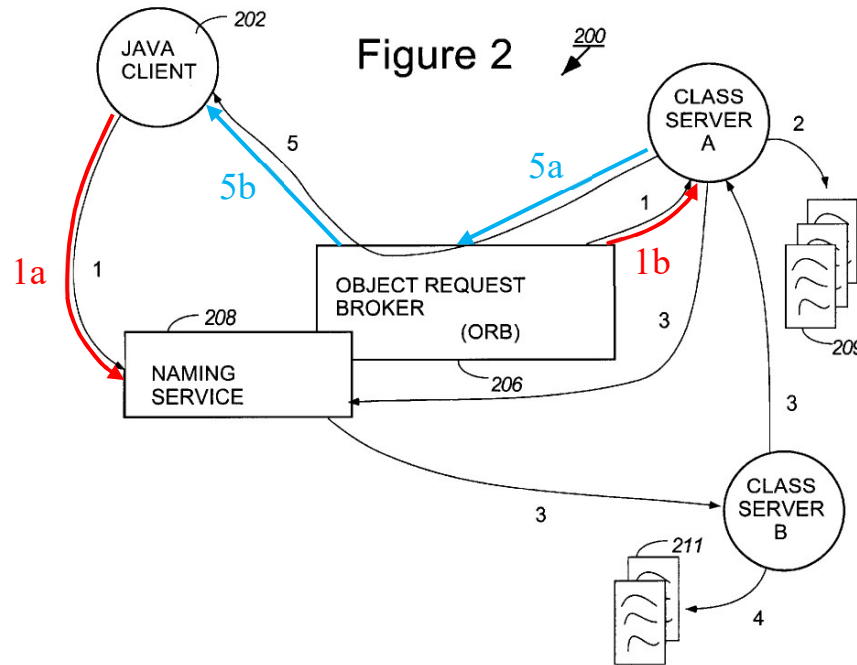
Claims 1-13 are obvious over Fowlow in view of Kimera. Ex.1002, ¶348.

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,

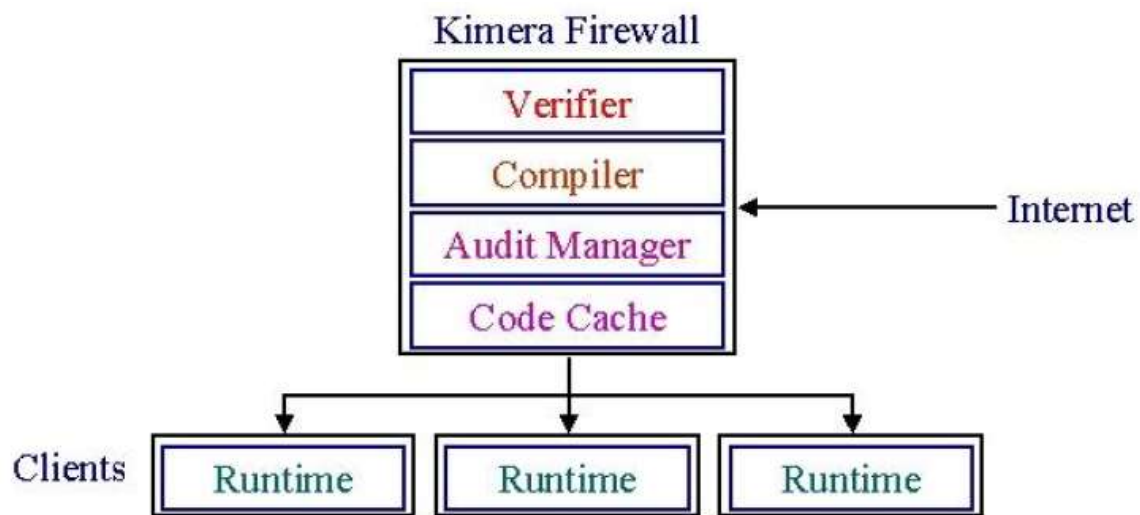
relatedly, describes acquiring software applications from the Internet via a “centralized server” that includes a verification service, compiler, audit manager, and code cache. Ex.1008, 1-2. Fowlow and Kimera are both directed to acquiring software capable of running on client devices from a remote network and are analogous art to the ’248 patent and to each other. Ex.1002, ¶¶349-353.

Fowlow teaches a loop involving the client device, an ORB, and one or more servers, which 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, ¶¶354-355.

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, ¶356.

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 (as taught by Kimera) to discover and resolve any unresolved dependencies (as taught by Fowlow), 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, ¶¶356-358.

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

translation” in response to future requests for the same code. Ex.1008, 2; Ex.1002, ¶¶360-361. 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, ¶362. 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 expected such a combination to be successful without overly-burdensome experimentation. Ex.1002, ¶362.

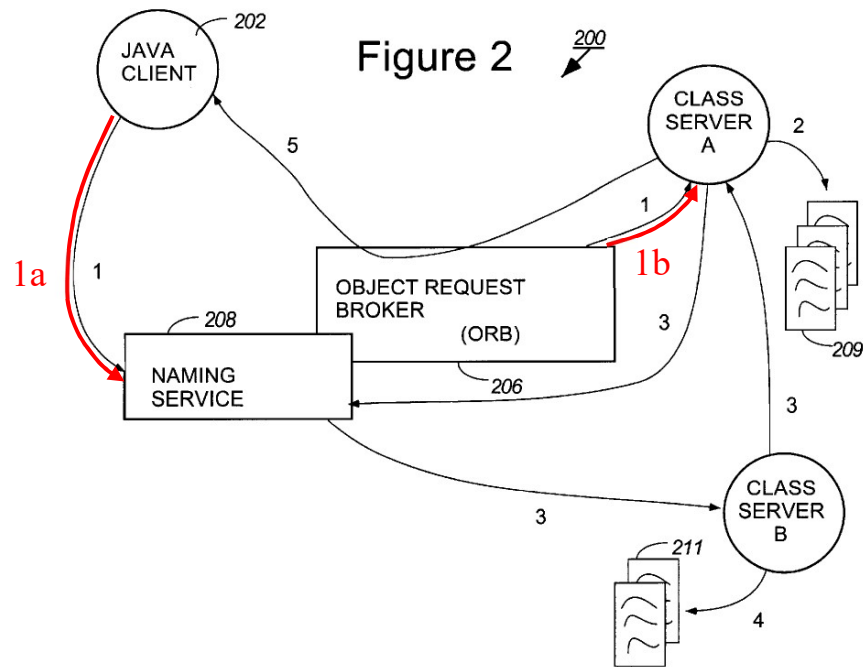
A POSITA would have been motivated to combine Fowlow and Kimera. *Id.*, ¶¶349-362.

2. Independent Claim 1

- a. *[Ipre] A method operating on a computer system for managing requests to a server computer for applets in a client server environment wherein each request for an applet specifies one form of the applet out of a plurality forms of the applet, comprising:*

To the extent the preamble is limiting, it is taught by Fowlow. Ex.1002, ¶363. 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. To accomplish this, Fowlow teaches an intermediary Object Request Broker (“ORB”), located at a server computer. *Id.*, 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); Ex.1002, ¶¶364-365. 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.”).

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. Client code requests all flow through the ORB, which in turn manages all applet requests that are sent to the connected servers. *Id.*, 10:18-34, Fig. 2.



Id., Fig. 2 (annotated); Ex.1002, ¶366.

The request specifies not only the application requested but the form of the application. For example, the request of Fowlow may be for a specific Java applet (*e.g.*, requesting an application) made up of Java class files stored in one or more servers. Ex.1009, 11:21-30, 1:44-57. 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.

Fowlow explains that “[an] ‘applet’ may refer to *a wide variety* of types of execution code,” such that a POSITA would have understood that it would be necessary to identify the type of execution code (*i.e.*, a form of the application) required by the client from the wide variety of types of code (*i.e.*, plurality of forms). *Id.*, 10:49-50; Ex.1002, ¶367.

Kimera further discloses that Java bytecode is compiled into native machine code at the centralized server. Ex.1008, 2. A POSITA would have understood that the programming language used by the client would dictate the form of the execution code needed on the requesting client machine. Ex.1002, ¶¶368-369; *see also* Ex.1009, 7:11-52 (“the form of the request will depend to a large extent upon the nature of the programming language used to create the client.”), 10:28-34. For example, the code necessary to run on one client developed using one programming language may differ from the code necessary to run on another client using a different language. Ex.1002, ¶369 (citing Ex.1001, 3:27-29 (“Binary format refers to processor specific machine instructions suitable for running natively on a given computing platform....”)).

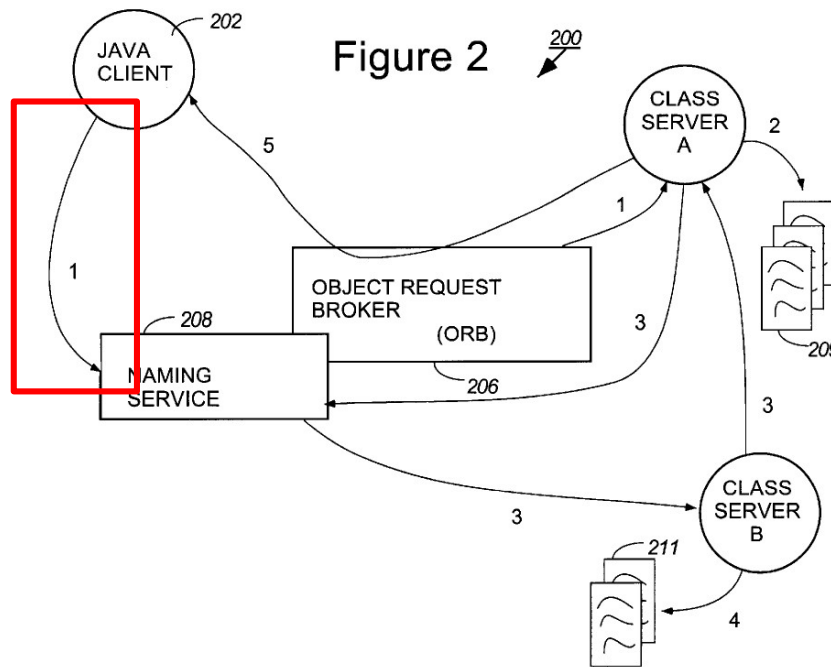
A POSITA would have understood that the form of the applet required by the client must be identified so the correct version of the applet would be served to the client device. Ex.1009, 3:38-42, 7:11-23; *see also* Ex.1008, 2 (noting different

native machine code binaries are compiled by the centralized compiler and the results are cached); Ex.1002, ¶369.

To the extent the preamble is limiting, a POSITA would have found it taught by Fowlow-Kimera. Ex.1002, ¶370.

- b. [1a] a) receiving on said server computer a request from a client computer for an applet in a form selected from a plurality forms;*

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, Abstract, 4:3-15 (describing the server receiving a “client request[] [for] additional applet execution code”), 10:18-34, Fig. 2.



Ex.1009, Fig. 2 (annotated); Ex.1002, ¶371.

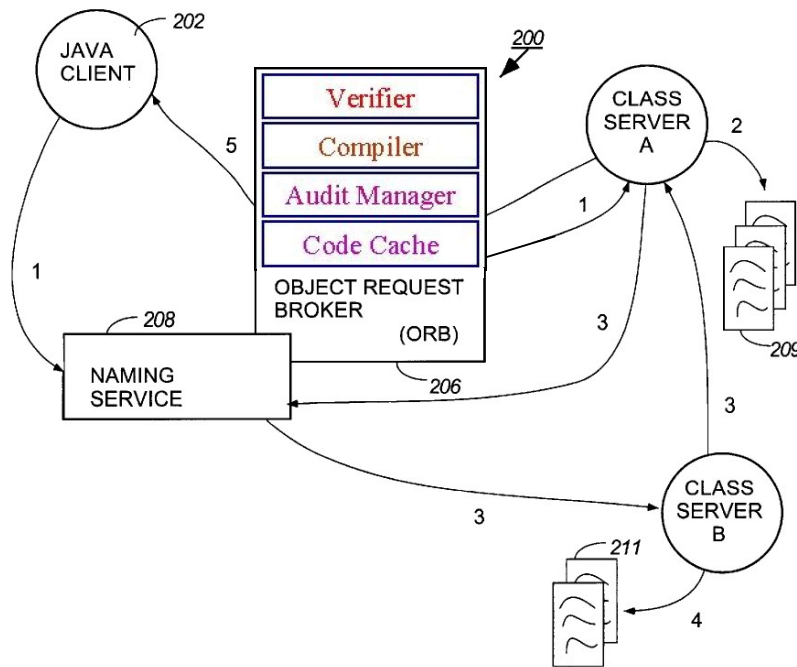
Further, as introduced in Section VI.E.2.a, the request specifies the form of the applet from of a plurality of forms, such as an intermediate form Java applet or an applet in “native machine code” for the requesting client computer. Ex.1009, 1:44-57, 7:11-52 (describing using a method table to locate and download the applet in the “‘native’ language of the client process”), 10:28-34, 10:49-50, 11:21-30; Ex.1002, ¶¶372, 367-369.

Fowlow-Kimera, therefore, renders this limitation obvious. Ex.1002, ¶373.

- c. [1b] b) compiling said applet into said selected form from a local resource comprising at least one source module and one compiler which acts on said source module to produce said selected form; and*

Fowlow discloses applets made up of “byte codes representing classes that are executed by a remote interpreter, such as Java classes...may also be compiled before execution....” Ex.1009, 10:50-56. Additionally, a POSITA would have understood that the “collection of byte codes representing classes” would be a collection of discrete files that would be compiled into a single applet. *Id.*, 10:45-65; Ex.1002, ¶¶374-377.

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.



Ex.1009, Fig. 2 (annotated); Ex.1008, 1; Ex.1002, ¶¶378-379.

A POSITA would have understood that the compiler of Kimera acts on a source module (*e.g.*, Java bytecode) to produce the selected form (*e.g.*, native machine code). Ex.1008, 2; Ex.1009, 10:49-50 (“[an] ‘applet’ may refer to *a wide variety* of types of execution code”); Ex.1002, ¶380. This disclosure comports with the ’248 patent, which explains that “[l]ocal resources 26 are comprised of program modules 32 (applets in source form, not the requested form) and compilers 30.” Ex.1001, 4:46-48, 5:23-26. A POSITA would have been motivated to combine Fowlow with the teaching of Kimera for the reasons identified in Section VI.E.1.

Additionally, the Kimera Firewall mirrors the server computer of the ’248 patent, which similarly compiles code before transmitting such code to the client.

Ex.1001, 3:49-54, Fig. 1; Ex.1002, ¶¶381-382. Both serve as an interface between client computers and external networks, such as the Internet. *Compare* Ex.1008, 1 (“[L]ocating verification, access control, auditing, and compilation services in centralized servers enables the same services to be performed faster, cheaper, and without introducing security vulnerabilities” and allows client devices to “be constructed from cheaper components”) *with* Ex.1001, 3:49-54 (noting the applet server computer verifies, optimizes, and compiles code before sending it to clients which “reduces the amount of work the client computers 12 and 14 have to do”); *see also* Ex.1002, ¶¶381-382.

The Fowlow-Kimera combination renders obvious this limitation. Ex.1002, ¶383.

d. [1c] c) transmitting said applet in said selected form to said client computer.

Finally, the Fowlow-Kimera combination teaches transmitting the transformed form of the application to the client computer. Fowlow discloses the “class code corresponding to the class name desired by the Java client is finally returned via the ORB.” Ex.1009, 12:6-9, 3:38-42, 7:11-23, Fig. 2 (arrow 5). “Once th[e] execution code for class name has been retrieved, it is delivered to the network class loader within the Java client in step 410. Next, in step 412, the NCL passes this execution code to the Java interpreter.” *Id.*, 13:43-46.

A POSITA would have been motivated to move the Java interpreter into the ORB as modified by Kimera as discussed in Section VI.E.1. After the requested applet was compiled and transformed (via the modified ORB), the compiled and transformed applet would be sent to the client computer. Ex.1002, ¶¶384-387. Specifically, as Kimera teaches, eliminating the “JIT compiler, verifier, [and] interpreter” from the client device and moving those processes to the modified ORB allows the “Java end-point [to] consist of only a minimal runtime environment.” Ex.1008, 2. This minimal runtime environment would necessitate the server sending the transformed form of the application to the client computer, so the client computer could display or enable the transformed form of the application for a user. Ex.1002, ¶387. This limitation is obvious in view of Fowlow-Kimera. *Id.*, ¶388.

Thus, Fowlow-Kimera renders this claim obvious. *Id.*, ¶389.

3. Dependent Claims 2-4

- a. *2. The method of claim 1, further comprising the step of: copying said applet in said selected form to a local cache after compiling said applet from said local resource if said cache does not contain a copy of said applet in said selected form.*

Claim 2 would have been obvious over the Fowlow-Kimera combination. Ex.1002, ¶¶390-393. As introduced in Section VI.E.2.c, Kimera’s architecture compiles Java bytecode (*i.e.*, the source module) to produce the selected form (*i.e.*, the native machine code). Ex.1008, 2. Kimera’s architecture further includes a

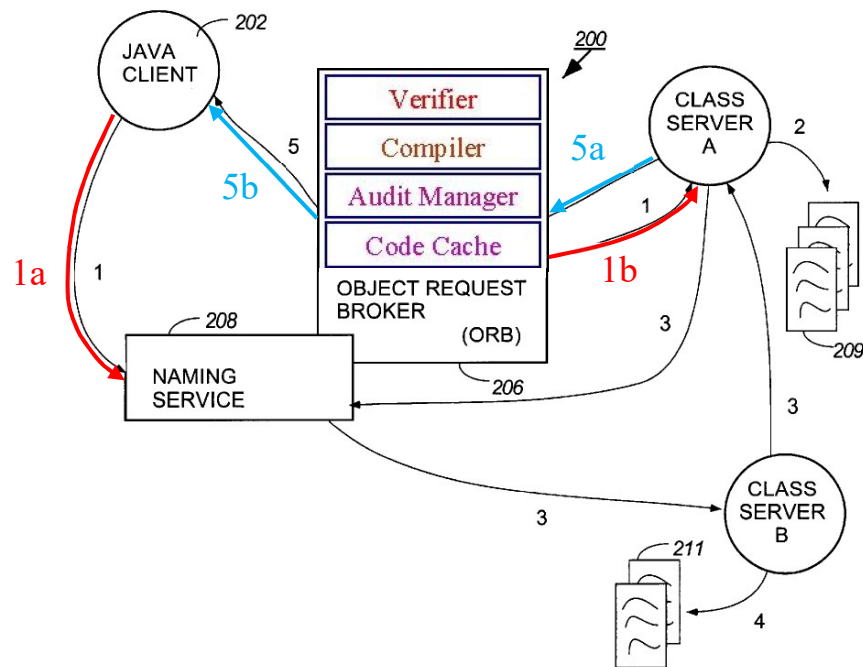
cache which “cach[es] the results of *recent compilations* in the central compilation server.” *Id.* If the compiled applet is not stored in cache memory, a POSITA would have understood Kimera’s disclosure of “caching the results of recent compilations” teaches copying the selected applet to a local cache. Ex.1002, ¶¶391-392.

This claim is obvious in view of Fowlow-Kimera. Ex.1002, ¶393.

- b. 3. The method of claim 2, further comprising the step of: transmitting a request to an external resource for said applet in said selected form if a copy of said applet is not stored in said local cache and said applet can not be compiled from said local resource, and directing said external resource to transmit said applet in said selected form to said server computer.*

As introduced in Section VI.E.1, a POSITA would have understood that the Fowlow-Kimera combination discloses copying the selected applet to a local cache because the compilation would be stored at Fowlow’s ORB in response to recent client requests. Ex.1009, 10:26-34; Ex.1008, 2; Ex.1002, ¶¶394-395, 391. If the applet was stored in the modified ORB’s code cache, it would be returned to the client (arrow 5b below) from the code cache. Ex.1009, Fig. 2; Ex.1008, 2; Ex.1002, ¶¶396-397. To the extent that a selected applet is not a recent compilation stored in the local cache, a POSITA would have understood that the Fowlow-Kimera system transmits a request to an external resource (arrow 1b below) for an applet in a selected form. Ex.1002, ¶¶398-400. The external resource (*i.e.*, applet server)

transmits the applet in the selected form to the server computer (arrow 5a below) before it is sent to the client.



Ex.1009, Fig. 2 (annotated); Ex.1008, 1; Ex.1002, ¶398.

This claim is obvious in view of Fowlow-Kimera. Ex.1002, ¶401.

- c. 4. The method of claim 1, further comprising the step of: transmitting a request to an external resource for supplying said applet in said selected form if said applet can not be compiled from said local resource and directing said external resource to transmit said applet in said selected form to said server computer.

Fowlow-Kimera renders this claim obvious as described in Section VI.E.3.b.

Ex.1009, 10:26-34, Fig. 2; Ex.1008, 2; Ex.1002, ¶¶391, 394-402. A POSITA would have understood the compiler on Kimera's centralized server would compile cached applets, if necessary, to fulfill client computer requests for applets. Ex.1002, ¶402.

If the compiler could not compile the applet from code stored in cache memory, the centralized server would seek the necessary code from external resources as taught by Fowlow. *See* Ex.1009, 10:45-56 (noting an “‘applet’ may refer to a wide variety of types of execution code” including “a collection of byte codes representing classes”), 12:51-54 (noting “a downloaded class [may] use[] other classes that are not currently known or defined within the Java client, [meaning] these other classes must be found and loaded (‘resolved’).”), Abstract, 13:60-67; *see also* Ex.1002, ¶402.

Thus, this claim is obvious in view of Fowlow-Kimera. Ex.1002, ¶403.

4. Independent Claim 5

- a. [5pre] A method operating on a computer system for managing requests to a server computer for applets in a client server environment wherein each request for an applet specifies one form of said applet out of a plurality of forms of said applet, comprising:*

Fowlow-Kimera renders this claim obvious as discussed in Section VI.E.2.a. Ex.1009, Abstract, 1:44-57, 3:38-42, 5:7-13, 5:34-36, 7:11-52, 10:15-34, 10:45-56, 11:21-30, Fig. 2; Ex.1008, 2; Ex.1001, 3:27-29; Ex.1002, ¶¶404, 363-370.

- b. [5a] a) receiving on said server computer a request from a client computer for an applet in a specified form selected from a plurality of forms;*

Fowlow-Kimera renders this claim obvious as discussed in Section VI.E.2.b. Ex.1009, Abstract, 1:44-57, 3:27-42, 4:3-15, 7:11-52, 10:18-34, 10:49-50, 11:21-30, Fig. 2; Ex.1002, ¶¶405, 371-373.

- c. [5b] b) determining whether said applet is stored in said specified form in a local cache and, if so, transmitting said applet in said specified form to said client computer;*

Fowlow-Kimera renders this limitation obvious as discussed in Section VI.E.3.b. Ex.1009, 10:26-34, Fig. 2; Ex.1008, 2; Ex.1002, ¶¶406, 394-401. Specifically, Kimera’s architecture “cach[es] the results of recent compilations in the central compilation server [which] can be used to further reduce the latency of binary translation.” Ex.1008, 2. A POSITA would have understood that latency would be reduced because compiled and cached applet code could be transmitted directly to the client computer. *Id.*; Ex.1002, ¶¶407-408.

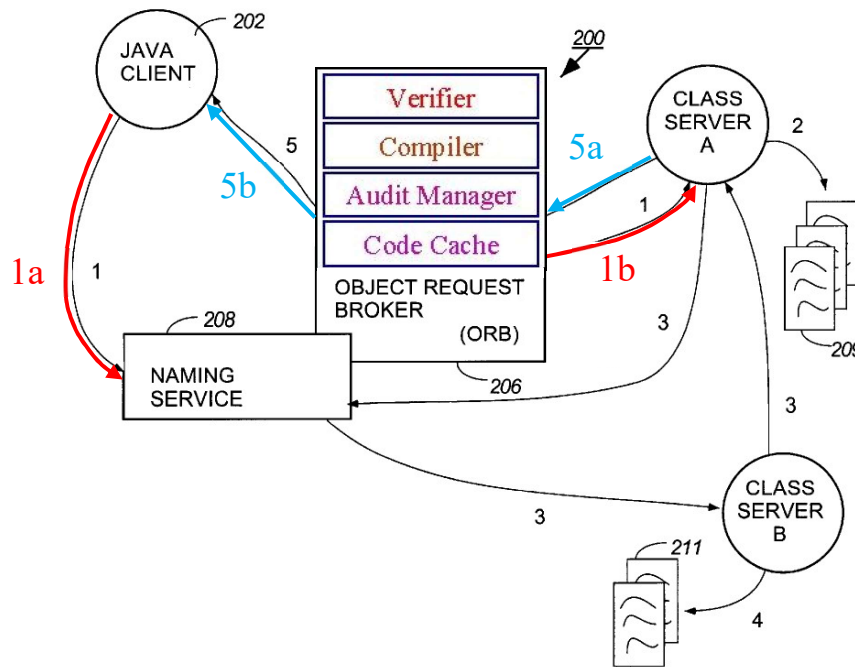
Fowlow-Kimera, therefore, teaches this limitation. Ex.1002, ¶409.

- d. [5c] c) if said applet is not stored in said selected form in said local cache,*

As discussed in Section VI.E.3.b, Fowlow-Kimera teaches managing requests for software depending on whether a copy of the selected form is stored in the modified ORB’s code cache. Ex.1008, 2; Ex.1002, ¶¶410, 394-401. If the applet

was stored in the modified ORB's code cache, it would be returned to the client (arrow 5b) from the code cache. Ex.1008, 2. To the extent that a selected applet is not a "recent compilation" stored in the local cache, a POSITA would have understood that the Fowlow-Kimera system would determine whether the software components or classes were stored in the code cache, but were not compiled into the specific, requested applet, the compiler on the modified ORB would compile the applet from local resources, "further reduc[ing] the latency of binary translation" by processing the applet locally. *Id.*; Ex.1002, ¶408; *see also* Ex.1009, 10:49-53 (noting "an applet may be a collection of byte codes representing classes that are executed by a remote interpreter, such as Java classes that are executed by a Java language interpreter").

If the request could not be fulfilled with applet classes stored in the code cache, the modified ORB would transmit a request to an external resource (arrow 1b) for an applet or applet classes in the selected form. Ex.1009, 11:57-12:11, 3:31-42; Ex.1002, ¶411. The external resource (*i.e.*, applet server) would transmit the applet to the server computer (arrow 5a), which would compile the applet before it is sent to the client. Ex.1009, 12:6-12:9, 3:38-42; Ex.1008, 2.



Ex.1009, Fig. 2 (annotated); Ex.1008, 1; Ex.1002, ¶411.

Fowlow-Kimera, therefore, teaches this limitation. Ex.1002, ¶412.

- e. [5d] compiling said applet into said selected form from a local resource comprising at least one source module and one compiler which acts on said source module to produce said selected form and*

Fowlow-Kimera renders this limitation obvious as discussed in Section VI.E.2.c. Ex.1009, 10:45-65, Fig. 2; Ex.1008, 1, 2; Ex.1001, 3:49-54; 4:46-48, 5:23-26, Fig. 1; Ex.1002, ¶¶413, 374-383.

- f. [5e] transmitting said applet in said selected form to said client computer.*

Fowlow-Kimera renders this limitation obvious as discussed in Section VI.E.2.d. Ex.1009, 3:38-42, 7:11-23, 12:6-9, 13:43-46, Fig. 2; Ex.1008, 2; Ex.1002, ¶¶414, 384-388.

5. Dependent Claims 6-8

- a. 6. The method of claim 5, further comprising the step of: copying said applet in said selected form to said cache after compiling said applet from said local resource if said cache does not contain a copy of said applet in said form.*

Fowlow-Kimera renders this claim obvious as described in Section VI.E.3.a.

Ex.1008, 2; Ex.1002, ¶¶415, 390-393.

- b. 7. The method of claim 6, further comprising the step of: transmitting a request to an external resource for said applet in said selected form if a copy of said applet is not stored in said local cache and said applet can not be compiled from said local resource, and directing said external resource to transmit said applet in said form to said server computer.*

Fowlow-Kimera renders this claim obvious as described in Section VI.E.3.b.

Ex.1009, 10:26-34, Fig. 2; Ex.1008, 2; Ex.1002, ¶¶416, 394-401.

- c. 8. The method of claim 5, further comprising the step of: transmitting a request to an external resource for said applet in said selected form if a copy of said applet is not stored in said local cache and said applet can not be compiled from said local resources, and directing said external resource to transmit said applet in said selected form to said server computer.*

Fowlow-Kimera renders this claim obvious as described in Section VI.E.3.c.

Ex.1009, Abstract, 10:26-34, 10:45-56, 12:51-54, 13:60-67, Fig. 2; Ex.1008, 2;
Ex.1002, ¶¶417, 394-403.

6. Independent Claim 9

- a. [9pre] A method operating on a computer system for generating an applet in response to a request by a client computer wherein each request for an applet specifies one form of the applet out of a plurality forms of the applet, comprising:*

Fowlow-Kimera renders this limitation obvious as described in Section VI.E.2.a. Ex.1009, Abstract, 1:44-57, 3:38-42, 5:7-13, 5:34-36, 7:11-52, 10:15-34, 10:45-56, 11:21-30, Fig. 2; Ex.1008, 2; Ex.1001, 3:27-29; Ex.1002, ¶¶418, 363-370.

- b. [9a] a) receiving on a server computer a request from a client computer for an applet in a form selected from a plurality forms;*

Fowlow-Kimera renders this limitation obvious as described in Section VI.E.2.b. Ex.1009, Abstract, 1:44-57, 3:27-42, 4:3-15, 7:11-52, 10:18-34, 10:49-50, 11:21-30, Fig. 2; Ex.1002, ¶¶419, 371-373.

- c. [9b] b) compiling an applet program module into an intermediate form program module;*

Fowlow-Kimera renders this limitation obvious as described in Section VI.E.2.c. Ex.1009, 10:45-65, Fig. 2; Ex.1008, 1, 2; Ex.1001, 3:49-54; 4:46-48, 5:23-26, Fig. 1; Ex.1002, ¶¶420, 374-383. A POSITA would have understood that “an intermediate form program module” would include Java bytecode, which Kimera expressly teaches compiling into native machine code at the centralized server. Ex.1008, 2; Ex.1009, 10:49-56 (explaining “an applet may be a collection of byte codes representing classes,” including Java classes, that “may also be compiled

before execution”); Ex.1002, ¶421; *see also* Ex.1001, 3:38-39 (“The intermediate byte-code need not necessarily be Java byte-code.”), 2:43-46. In addition to Java bytecode, other intermediate forms of code would have been known to a POSITA. Ex.1002, ¶¶421-422.

Fowlow-Kimera renders obvious this limitation. Ex.1002, ¶423.

- d. [9c] c) transmitting said applet in said selected form to said client computer.*

Fowlow-Kimera renders this limitation obvious as described in Section VI.E.2.d. Ex.1009, 3:38-42, 7:11-23, 12:6-9, 13:43-46, Fig. 2; Ex.1008, 2; Ex.1002, ¶¶424, 384-388.

Thus, Fowlow-Kimera renders obvious claim 9. Ex.1002, ¶425.

7. Dependent Claims 10-13

- a. 10. The method of claim 9, further comprising the step of: transforming said intermediate form program module into a transformed intermediate form program module with at least one transformer program.*

Fowlow describes processing applets 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. Transforming an intermediate form program module via compression was, therefore, well known. Ex.1002, ¶¶426-427. 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, ¶427.

The encryption and debugging functionality of Fowlow’s filter 40 further work to verify code segments. 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. Ex.1002, ¶¶428, 431-432. 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; Ex.1002, ¶¶433-434.

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, ¶433. The centralized server would also apply a tamper-proof signature after the code had been verified. Ex.1008, 2. A POSITA would have understood the verified and signed code would be a transformed intermediate program module, ready to be returned to the client device. *Id.*; Ex.1002, ¶¶433-434. The debugged, encrypted applet with a tamper-

proof signature applied is a transformed intermediate form program module as recited in claim 10. This is consistent with the teachings of the '248 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, 5:38-40; Ex.1002, ¶428.

The Fowlow-Kimera system also teaches the transforming intermediate form program modules by optimizing those programs. Specifically, a POSITA would have understood that Fowlow's filter 40's tracing and debugging functions suggest optimization because both functions have long been used to analyze programs and source code for the purpose of optimization. Ex.1002, ¶¶429-430, 119, 134-135. Relatedly, 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 at the server in order to achieve faster services performed on the most inexpensive clients possible. Ex.1002, ¶427.

Kimera further teaches that “[c]ompilation from Java bytecode into native machine code is another service that is best performed at a central location.” Ex.1008, 2. A POSITA would have understood that the compiler is a transformer

program that acts on Java bytecode to produce a transformed intermediate program module, such as native machine code. *Id.*, 2; Ex.1002, ¶435.

Finally, a POSITA would have been familiar with a wide range of multi-stage transformer operations for a proxy device that services a variety of operating systems and architectures. Ex.1002, ¶¶435, 136-147. This knowledge would have further rendered claim 10 obvious. *Id.*

Fowlow-Kimera, therefore, renders this claim obvious. Ex.1002, ¶436.

- b. *11. The method of claim 10, wherein said at least one transformer program is selected from the group consisting of verifying computer programs, optimizing computer programs, compressing computer programs, debugging computer programs, usage monitoring computer programs and encrypting computer programs.*

Fowlow-Kimera renders this claim obvious as described in Section VI.E.7.a. Ex.1009, 6:5-8; Ex.1008, 1, 2; Ex.1001, 5:38-40; Ex.1002, ¶¶119, 134-147, 426-437. Specifically, Fowlow-Kimera teaches and suggests various programs that transform the intermediate program modules, including the following: a “verification service” that verifies programs (Ex.1008, 2); optimization programs to take advantage of reduced computation and memory requirements in client computers (*Id.*, 1, 2); compression programs (Ex.1009, 6:5-8), and encryption computer programs, including the application of cryptographically strong hash functions such as MD5 (*Id.*; Ex.1008, 2). Ex.1002, ¶¶438-442.

Any one of the transformer programs taught in Fowlow-Kimera are sufficient to find this claim obvious over the prior art. *Fresenius*, 582 F.3d at 1298.

Fowlow-Kimera, therefore, renders this claim obvious. Ex.1002, ¶443.

- c. 12. The method of claim 11, further comprising the step of: compiling said transformed intermediate form program module into machine specific binary code with a target compiler.*

As discussed in Section VI.E.2.a, Fowlow discloses client requests that specify the form of application. Ex.1009, 11:21-30, 1:44-57, 10:45-56; Ex.1002, ¶¶444, 367-369. Once requested, a POSITA would have understood that the compiler of Kimera is a target compiler because it would transform the code based on the request of Fowlow into machine-specific binary code. Ex.1008, 2; Ex.1002, ¶¶444-445; *see also* Ex.1001, 5:11-26 (“A target compiler 52 compiles intermediate applet code 44 into an applet 54 in a processor specific format (binary) suitable for running natively on a given computing platform.”). Kimera’s compiler would be a target compiler because, as Kimera explains, “[c]ompilation from Java bytecode into native machine code is another service that is best performed at a central location.” Ex.1008, 2.

Fowlow-Kimera, therefore, renders this claim obvious. Ex.1002, ¶446.

- d. 13. The method of claim 10, further comprising the step of: compiling said transformed intermediate form program module into machine specific binary code with a target compiler.

Fowlow-Kimera renders this claim obvious as described in Section VI.E.7.c. Ex.1009, 11:21-30, 1:44-57, 10:45-56; Ex.1008, 2; Ex.1001, 5:11-26; Ex.1002, ¶¶367-369, 444-447.

F. The Petition Should Not Be Discretionarily Denied

1. Advanced Bionics

Kimera was cited in an IDS submitted to the USPTO but did not form the basis of the sole rejection in the prosecution of the '248 patent. Ex.1004, 31-34, 43-49. 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).

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).

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.

Second, there is no trial date presently set for any litigation involving the '248 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.¹⁰

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.

¹⁰ 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).

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.

VII. CONCLUSION

The Petition demonstrates a reasonable likelihood that at least one claim is unpatentable. Ex.1002, ¶448. Accordingly, all grounds in the Petition should be instituted. *SAS Institute Inc. v. Iancu*, 138 S.Ct. 1348, 1359-60 (2018).

Respectfully submitted by

O'MELVENY & MYERS LLP,

By: /s/ Cameron W. Westin
Cameron W. Westin
Reg. No. 66,188

I. CLAIM APPENDIX OF THE CHALLENGED CLAIMS

- [1pre] A method operating on a computer system for managing requests to a server computer for applets in a client server environment wherein each request for an applet specifies one form of the applet out of a plurality forms of the applet, comprising:
- [1a] a) receiving on said server computer a request from a client computer for an applet in a form selected from a plurality forms;
- [1b] b) compiling said applet into said selected form from a local resource comprising at least one source module and one compiler which acts on said source module to produce said selected form; and
- [1c] c) transmitting said applet in said selected form to said client computer.
- [2] The method of claim 1, further comprising the step of: copying said applet in said selected form to a local cache after compiling said applet from said local resource if said cache does not contain a copy of said applet in said selected form.
- [3] The method of claim 2, further comprising the step of: transmitting a request to an external resource for said applet in said selected form if a copy of said applet is not stored in said local cache and said applet can not be compiled from said local resource, and directing said external resource to transmit said applet in said selected form to said server computer.

- [4] The method of claim 1, further comprising the step of: transmitting a request to an external resource for supplying said applet in said selected form if said applet can not be compiled from said local resource and directing said external resource to transmit said applet in said selected form to said server computer
- [5pre] A method operating on a computer system for managing requests to a server computer for applets in a client server environment wherein each request for an applet specifies one form of said applet out of a plurality of forms of said applet, comprising:
- [5a] a) receiving on said server computer a request from a client computer for an applet in a specified form selected from a plurality of forms;
- [5b] b) determining whether said applet is stored in said specified form in a local cache and, if so, transmitting said applet in said specified form to said client computer;
- [5c] c) if said applet is not stored in said selected form in said local cache, compiling said applet into said selected form from a local resource comprising at least one source module and one compiler which acts on said source module to produce said selected form and transmitting said applet in said selected form to said client computer.
- [6] The method of claim 5, further comprising the step of: copying said applet in said selected form to said cache after compiling said applet from said local resource if said cache does not contain a copy of said applet in said form.

- [7] The method of claim 6, further comprising the step of: transmitting a request to an external resource for said applet in said selected form if a copy of said applet is not stored in said local cache and said applet can not be compiled from said local resource, and directing said external resource to transmit said applet in said form to said server computer.
- [8] The method of claim 5, further comprising the step of: transmitting a request to an external resource for said applet in said selected form if a copy of said applet is not stored in said local cache and said applet can not be compiled from said local resources, and directing said external resource to transmit said applet in said selected form to said server computer.
- [9] A method operating on a computer system for generating an applet in response to a request by a client computer wherein each request for an applet specifies one form of the applet out of a plurality forms of the applet, comprising:
- [9a] a) receiving on a server computer a request from a client computer for an applet in a form selected from a plurality forms;
- [9b] b) compiling an applet program module into an intermediate form program module;
- [9c] c) transmitting said applet in said selected form to said client computer.
- [10] The method of claim 9, further comprising the step of: transforming said intermediate form program module into a transformed intermediate form program module with at least one transformer program.

[12] The method of claim 11, further comprising the step of: compiling said transformed intermediate form program module into machine specific binary code with a target compiler.

[13] The method of claim 10, further comprising the step of: compiling said transformed intermediate form program module into machine specific binary code with a target compiler.

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 5th day of April, 2024:

Kowert, Hood, Munyon, Rankin, & Goetzel, PC
Jeffrey Hood
1120 South Capital of Texas Highway
Building 2, Suite 300
Austin, Texas 78746
United States

With an electronic courtesy copy to litigation counsel:

James M. Lennon
jlennon@devlinlawfirm.com
Timothy Devlin
tdevlin@devlinlawfirm.com
1526 Gilpin Ave
Wilmington, Delaware 19806
Devlin Law Firm, LLC

By: /s/ Cameron W. Westin
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

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 13,388 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: /s/ Cameron W. Westin
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