

UNITED STATES PATENT AND TRADEMARK OFFICE

PATENT TRIAL AND APPEAL BOARD

Capital One, National Association,
Petitioner

v.

Implicit, LLC,
Patent Owner

CASE: IPR2024-00878
U.S. PATENT NO. 7,774,740

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)	vii
I. GROUNDS FOR STANDING (37 C.F.R. §42.104(a)).....	1
II. STATEMENT OF PRECISE RELIEF REQUEST AND REASONS THEREFORE	1
III. SUMMARY OF THE '740 PATENT	2
A. Background of the Technology	2
B. Prosecution History	4
IV. CLAIM CONSTRUCTION	5
V. ARGUMENTS	7
A. Level of Ordinary Skill in the Art.....	7
B. Prior Art.....	7
1. Fowlow (Ex. 1009)	7
2. Kimera (Ex. 1008).....	9
3. Parthasarathy (Ex. 1007).....	12
4. Angles (Ex. 1010)	14
C. Ground 1: Claims 11-19 are Unpatentable Under 35 U.S.C. §103(a) as Obvious over Fowlow	15
1. Independent Claim 11	15
2. Dependent Claims 12-19.....	24
D. Ground 2: Claims 11-21 are Unpatentable Under 35 U.S.C. §103(a) as Obvious over Fowlow in view of Kimera.....	30
1. Motivation to Combine	30
2. Independent Claim 11	35
3. Dependent Claims 12-21.....	41

E.	Ground 3: Claims 11-12, 14-21 are Unpatentable Under 35 U.S.C. §103(a) as Obvious over Parthasarathy in view of Kimera	48
1.	Motivation to Combine	48
2.	Independent Claim 11	53
3.	Dependent Claims 12, 14-21.....	62
F.	Ground 4: Claim 13 is Unpatentable Under 35 U.S.C. §103(a) as Obvious over Parthasarathy in view of Kimera and Angles.....	70
1.	Motivation to Combine	70
2.	Dependent Claim 13	72
G.	The Petition Should Not Be Discretionarily Denied.....	73
1.	<i>Advanced Bionics</i>	73
2.	<i>Fintiv</i>	73
VI.	CONCLUSION.....	75
VII.	CLAIM APPENDIX OF THE CHALLENGED CLAIMS	76

LIST OF EXHIBITS

Exhibit No.	Description
1001	U.S. Patent No. 7,774,740 (the “’740 patent”)
1002	Declaration of Erez Zadok, Ph.D.
1003	Curriculum Vitae of Erez Zadok, Ph.D.
1004	Prosecution History of U.S. Patent Application No. 11/241,985
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	<i>Reserved</i>
1012	Affidavit of Nathaniel E. Frank-White, dated 2/9/2024
1013	<i>Reserved</i>
1014	Microsoft Computer Dictionary, 3rd ed., 1997
1015	Hypertext Transfer Protocol – HTTP/1.1, RFC 2068, The Internet Engineering Task Force (IETF), January 1997. [Download from https://www.rfc-editor.org/rfc/pdf/rfc2068.txt.pdf]

Exhibit No.	Description
1016	HTML 4.0 Specification, The World Wide Web Consortium (W3C) Recommendation, December 18, 1997. [Download from https://www.w3.org/TR/REC-html40-971218/html40.pdf]
1017	HTTP State Management Mechanism, RFC 2109, The Internet Engineering Task Force (IETF), February 1997. [Download from https://www.rfc-editor.org/rfc/pdf/rfc2109.txt.pdf]
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. [https://www.rfc-editor.org/rfc/pdf/rfc2187.txt.pdf]
1020	GNU C Manual Page documentation, 1993
1021	Persistent Client State HTTP Cookies, October 1996
1022	<i>Reserved</i>
1023	<i>Reserved</i>
1024	<i>Reserved</i>
1025	<i>Implicit, LLC v. Trend Micro, Inc.</i> , No. 6:16-cv-00080-JRG, Dkt. No. 115, Markman Order (Mar. 29, 2017)

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

A. Real Parties-In-Interest

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

- Capital One, National Association

B. Related Matters

Patent Owner, Implicit, LLC, asserted U.S. Patent No. 7,774,740 (“’740 patent”) in the following litigations:

- *Implicit LLC v. Capital One Financial Corp et al.*, (E.D. Tex.) Case No. 4:23-cv-00770-SDJ;
- *Implicit, LLC v. Hulu, LLC*, (W.D. Tex.) Case No. 6:23-cv-00513-ADA-DTG;
- *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; and
- *Implicit Networks, Inc. v. Hewlett-Packard Company* (N.D. Cal.) Case No. 3:10-cv-03746.

The ’740 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. 6,976,248;
- 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 has also challenged U.S. Patent No. 8,056,075 in IPR2024-00643 and U.S. Patent No. 6,976,248 in IPR2024-00787.

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

Lead Counsel	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 to the email addresses provided above.

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

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

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

Petitioner certifies that the '740 patent is available for IPR and that Petitioner is not barred or estopped from requesting IPR.

II. STATEMENT OF PRECISE RELIEF REQUEST AND REASONS THEREFORE

Petitioner requests the institution of IPR and the cancellation of the following claims (collectively, the “Challenged Claims”) on the following Grounds:

Ground	Basis	Relied-On References	Claim(s)
1	§103 ¹	U.S. Pat. No. 6,260,078 to Fowlow et al. (“Fowlow”) (Ex. 1009)	11-19
2	§103	Fowlow in view of Emin Gün Sirer & Sean McDermid & Brian Bershad, <u>Kimera: A System Architecture for Networked Computers</u> , Univ. of Wash., 1997 (“Kimera”) (Ex. 1008)	11-21
3	§103	U.S. Pat. No. 6,347,398 to Parthasarathy et al. (“Parthasarathy”) (Ex. 1007) and Kimera	11-12, 14-21
4	§103	Parthasarathy, Kimera, and U.S. Patent No. 5,933,811 to Angles et al. (“Angles”) (Ex. 1010)	13

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

The Petition is accompanied by the declaration of Dr. Erez Zadok, an expert in the field of the '740 patent, who opines each of the Challenged Claims are obvious over the prior art. Ex. 1002, ¶¶1-44, 201-202, 495.

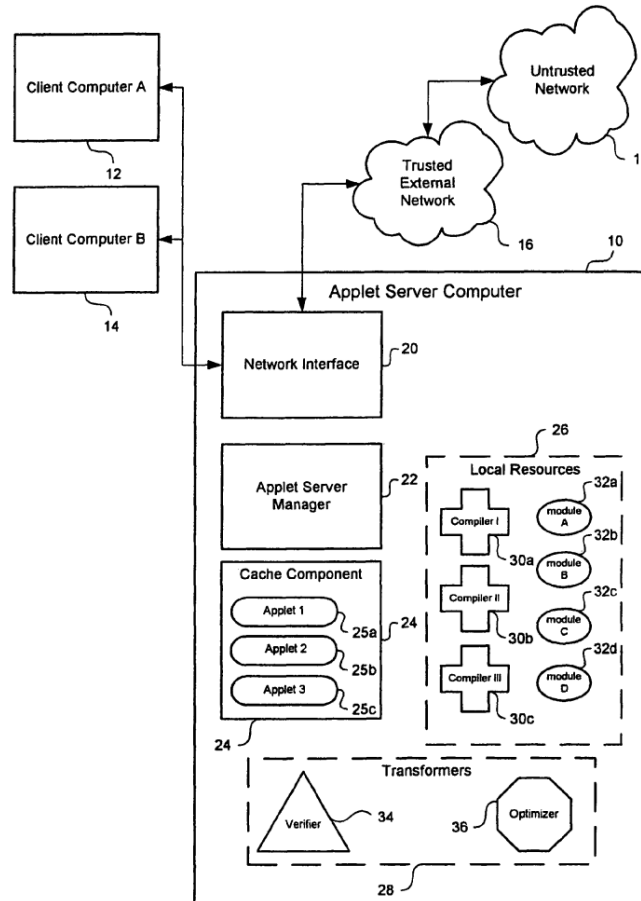
III. SUMMARY OF THE '740 PATENT

A. Background of the Technology

The '740 patent discloses server architecture that delivers applications requested by a client computer. Ex. 1001, 2:16-29, 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, 2:16-29, 3:46-4:10. The compiled applet is then transmitted to the

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, 3:13-18.

requesting client computer. *Id.* Figure 1 of the '740 patent depicts an applet server computer 10 that sits between client computers 12, 14 and external networks 16, 18.



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

³ “Verification, optimization and compression are examples of types of transformation operations.” Ex. 1001, 3:53-55.

date of the '740 patent as shown in the references described herein. Ex. 1002, ¶¶45-156 (describing the background of general computer operations, web technologies, caching technologies, programming languages, compilers, interpreters, data and file transformation, and an overview of the '740 patent).

B. Prosecution History

The '740 patent is part of a long chain of continuation patents that has involved extensive prosecution. Issued claim 11 was added part-way through prosecution, but was rejected as obvious over U.S. Patent No. 6,253,228 to Ferris ("Ferris"). Specifically, the examiner found Ferris teaches, "at the server computer, receiving a request from the client computer, the request identifying a form of the application, the identified form being one of a plurality of available forms." Ex. 1004, 86-87. The examiner continued that Ferris teaches "in response to receiving the request: generating the identified form of the application from an intermediate [] source form of the application" and "transforming the application into a transformed version of the generated form of the application; and sending the transformed form of the application to the client computer...." *Id.*, 87-89; Ex. 1002, ¶¶157-159.

The applicant argued Ferris was directed to a system involving applets already running on a client, and therefore did not teach the allegedly claimed "deliver[ing] applets to a client *from* a server to a client...." *Id.*, 100-101. Additionally, the applicant argued Ferris taught simply receiving a request, performing that request,

and terminating the connection without any transforming taking place as a result of the server receiving the request. *Id.*, 102-103; Ex. 1002, ¶¶160-167. The claims were then allowed.

IV. CLAIM CONSTRUCTION

For purposes of this proceeding, Petitioner believes no terms need to be construed as the prior art utilizes substantially the same language such that, under any construction of the terms of the Challenged Claims, the claims are unpatentable. *Nidec Motor Corp. v. Zhongshan Broad Ocean Motor Co. Ltd.*, 868 F.3d 1013, 1017 (Fed. Cir. 2017) citing *Vivid Techs., Inc. v. Am. Sci. & Eng'g, Inc.*, 200 F.3d 795, 803 (Fed. Cir. 1999). While the Board need not construe any terms to find the Challenged Claims unpatentable, three terms warrant additional context.

First, claim element [11a] recites a “form of the application,” which was clarified in prosecution to mean “whether [the application] is source code, binary code or intermediate code.” Ex. 1004, 101 (“Applicant surmises that the Examiner has confused the term ‘form’ of the application in the present invention, i.e., whether it is source code, binary code or intermediate code, with the FORM element of HTML discussed in *Ferris*.”). To the extent this term is construed, it should be construed consistent with the applicant’s representation during prosecution: to differentiate the steps in the code compilation progression.

Second, regarding the term “transforming” as that term appears in element [11c], the ultimate parent in the ’740 patent family included the term “transforming operation,” which was construed in *Implicit, LLC v. Trend Micro, Inc.*, No. 6:16-cv-00080-JRG, to mean “operation that modifies code.” Ex. 1025, 43-45. To the extent this term is construed, it should be construed consistent with the previously construed “transforming operation” term. *See TQP Development, LLC v. Intuit Inc.*, No. 2:12-cv-180, 2014 WL 2810016, at *6 (E.D. Tex. June 20, 2014); Ex. 1002, ¶¶222-223.

Third, the term “wherein transforming comprises execution and compression” as it appears in element [11c], based on the specification refers to encasing unsafe code that prevents execution at the client, modifying unsafe code to make it safe, and/or flagging to a user that a code fragment is unsafe. Ex. 1001, 5:34-40; Ex. 1002, ¶¶224-229. While the specification includes other references to “execution” and “executing,” they all refer to a *client* executing a received application, and not an action undertaken by the server as part of “transforming.” *See* Ex. 1001, 1:37-38, 2:6-8, 2:17-19, 6:30-31, 6:43-46. Thus, to the extent this term is construed, it should be construed to at least encompass encasing unsafe code to prevent the client from executing that unsafe code, modifying unsafe code to make it safe, or flagging the risk of executing the unsafe code at the client. Ex. 1002, ¶229.

The Challenged Claims are unpatentable under any construction of the above as the prior art relied upon herein describes the claimed processes using the same language.

V. ARGUMENTS

A. Level of Ordinary Skill in the Art

A POSITA with respect to this patent would have had a bachelor's degree in computer science, computer engineering, or equivalent degree, and approximately two years of experience working in the computer science or engineering field. A POSITA would have had knowledge of design considerations known in the industry and would have been familiar with then-existing products and solutions, including compilers, programming languages, networking, and web systems and applications. Additional work experience could make up for less education and vice-versa. Ex. 1002, ¶¶168-170.

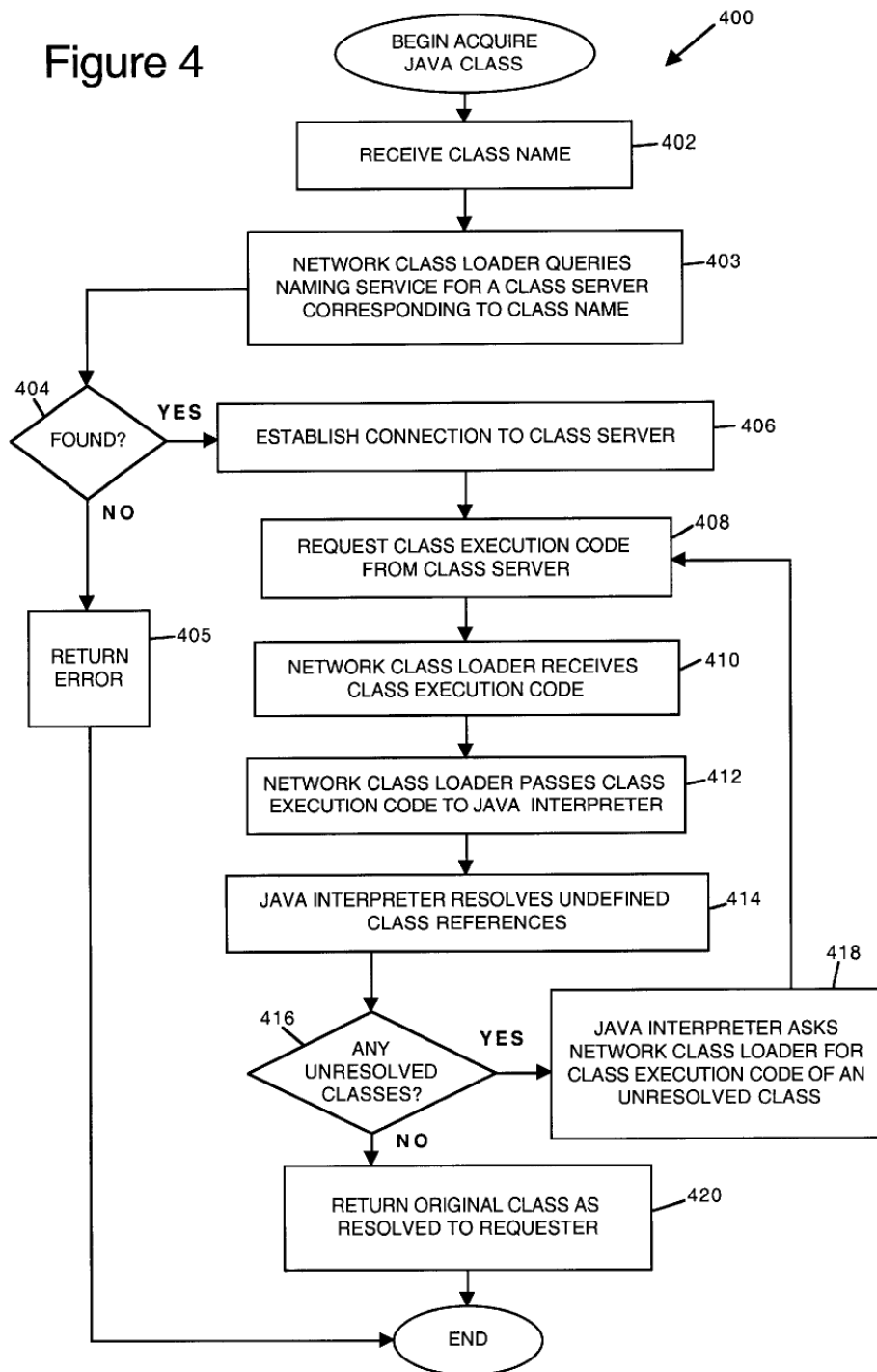
B. Prior Art

1. 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

applet servers to locate and return the desired code. *Id.*, 3:29-36. The applet servers respond to the request by retrieving and returning the “desired applet execution code to the client by way of the object request broker.” *Id.*, 3:36-42. A Java interpreter on the client determines whether the returned applet execution code contains any “unresolved classes,” *e.g.*, missing dependencies, and if so, sends another request to the object request broker to retrieve the unresolved classes from the server(s). *Id.*, Abstract, 13:33-67. The applet download process is depicted below:

Figure 4



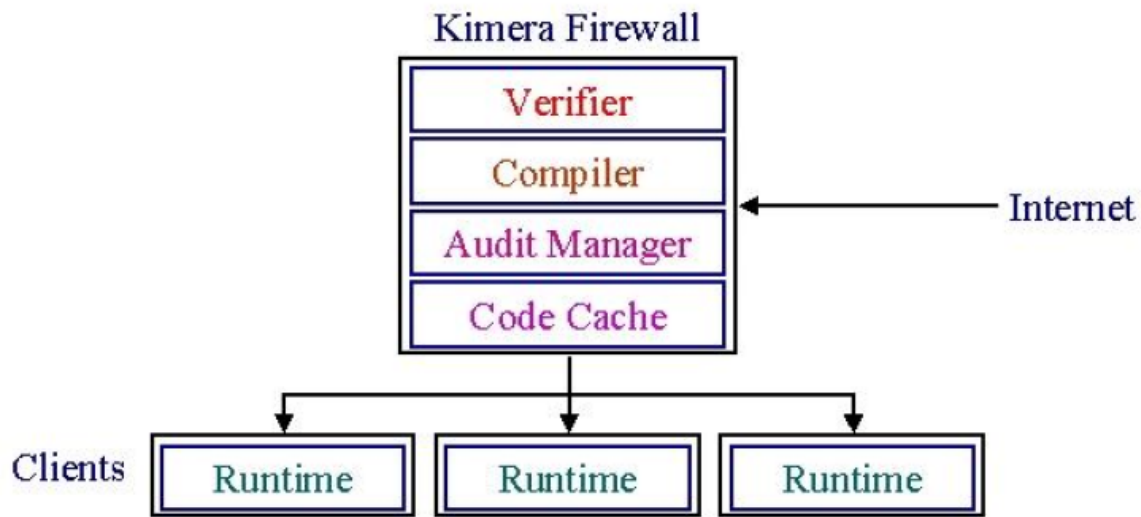
Id., Fig. 4; *see also id.*, 11:57-12:11, 13:18-67; Ex. 1002, ¶¶171-179.

2. 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 '740 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.



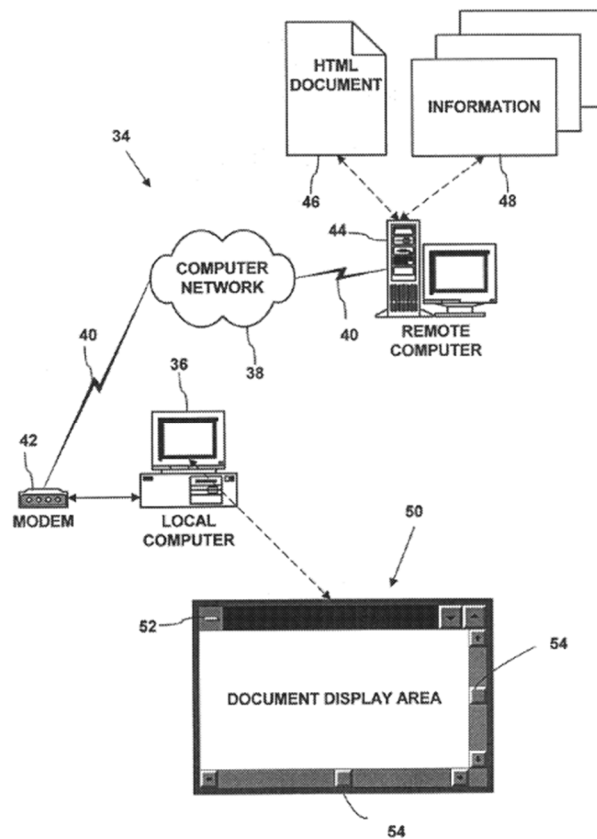
Id.

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, ¶186. 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, ¶¶180-186.

3. Parthasarathy (Ex. 1007)

U.S. Patent No. 6,347,398 to Parthasarathy et al. (“Parthasarathy”) was filed on November 8, 1999, claims priority and is a continuation of 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, ¶¶187-194.

4. Angles (Ex. 1010)

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

Angles explains the “small piece of information which a web server (via a CGI script) can store with a web browser...is useful for having the browser remember some specific information across several pages.” Ex. 1010, 11:5-12. The example given includes remembering items added to a shopping cart in a “virtual shopping mall” “so that the consumer can pay for all the items [] once he has finished shopping.” *Id.* Angles then describes how to set a cookie using a CGI script. *Id.*, 11:13-49; Ex. 1002, ¶¶195-199.

C. Ground 1: Claims 11-19 are Unpatentable Under 35 U.S.C. §103(a) as Obvious over Fowlow

Claims 11-19 are obvious over Fowlow. Ex. 1002, ¶203.

1. Independent Claim 11

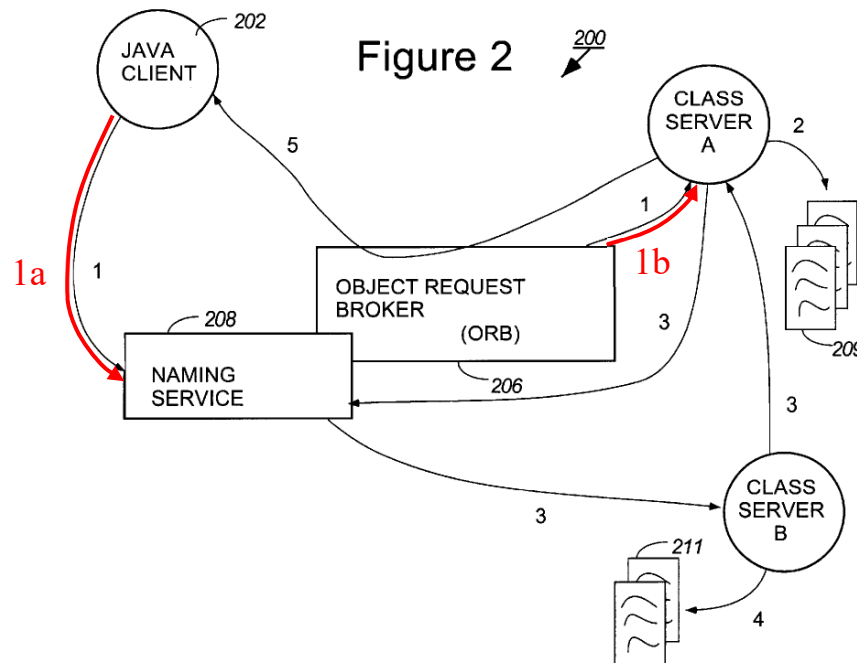
- a. [11pre] A method operating on a computer system, having a client computer and a server computer, for managing requests to the server computer, the method comprising:*

To the extent the preamble is limiting, it is taught by Fowlow. Ex. 1002, ¶204.

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, ¶¶205-206. 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 application requests that are sent to the connected servers. *Id.*, 10:18-34, Fig. 2.

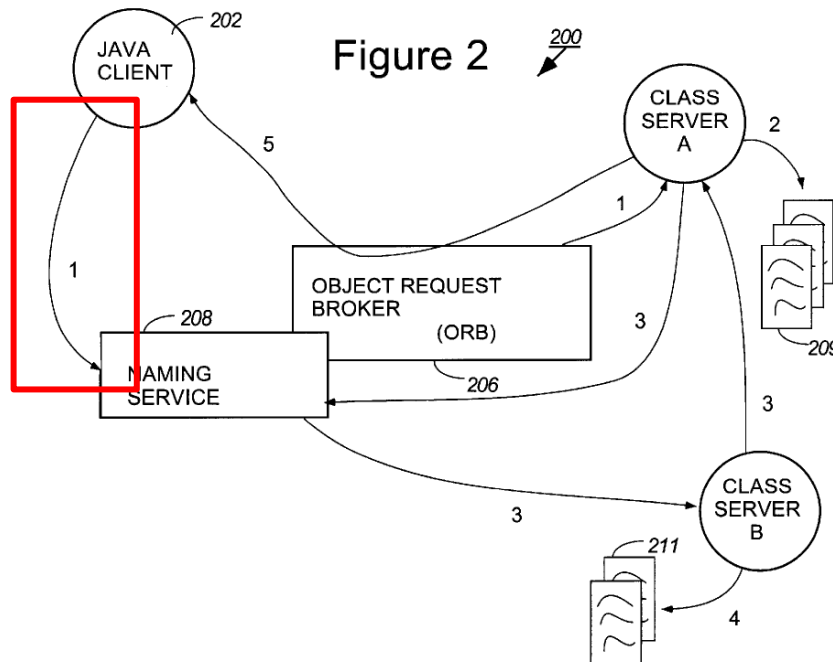


Id., Fig. 2 (annotated); Ex. 1002, ¶207.

To the extent the preamble is limiting, a POSITA would have found it taught by Fowlow. Ex. 1002, ¶208.

- b. [11a] at the server computer, receiving a request from the client computer, the request identifying an application and identifying a form of the application; and

Fowlow describes a client request-server response interaction, which includes an ORB receiving the at least one request. For example, Fowlow teaches that a client requests an application 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, ¶209.

The request identifies not only the requested application but also identifies a form of the application from a plurality of forms. 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, ¶210.

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, ¶¶210-211; *see also* Ex. 1009, 7:11-20 (“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, ¶211 (citing Ex. 1001, 3:19-23 (“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 application required by the client must be identified so the correct version of the application would be served to the client device. Ex. 1009, 3:38-42, 7:11-23 (describing how the binary required by the client is dictated by “the nature of the programming language used to create the client,” which would include Java applications), 10:28-34; Ex. 1002, ¶¶211-212.

Fowlow, therefore, renders this limitation obvious. Ex. 1002, ¶213.

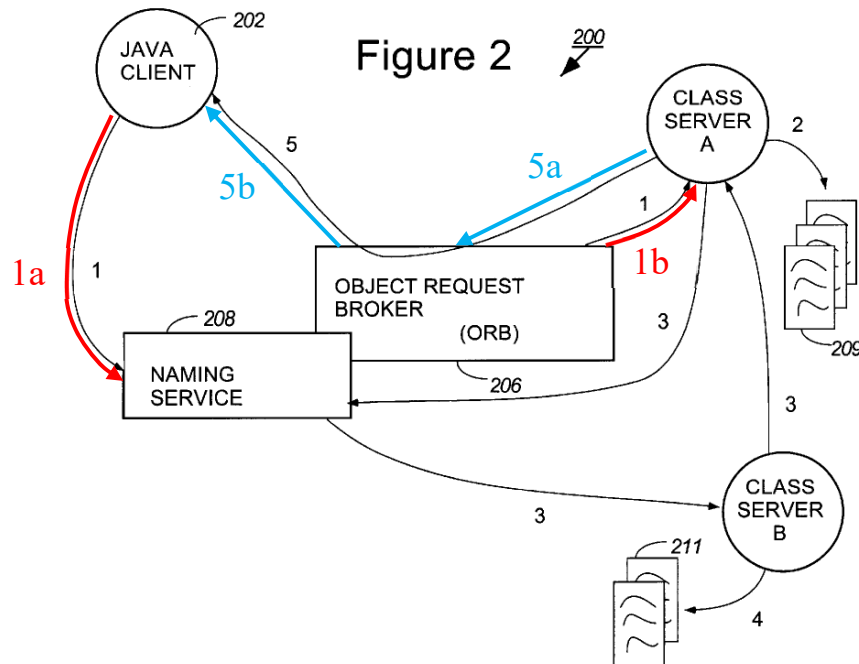
c. [11b] in response to receiving the request: compiling the application into a compiled form;

Fowlow describes a client request-server response interaction, where the server performs certain actions in response to a client request. *See* Ex. 1009, Abstract, 3:27-42; Ex. 1002, ¶¶219-220. In response to this request, Fowlow’s ORB performs a number of actions: (1) locating and sending code to the client from memory local to the server (e.g., cached data), (2) seeking the requested software from another server in the distributed network and returning the files to the client once received from the second server, and (3) searching recursively until the desired code is found. Ex. 1009, 3:32-4:2; *see also* Ex. 1009, Abstract, 11:57-12:9.

The requested applications are 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. A POSITA would have

understood that the byte code of the requested application would need to be compiled, as Fowlow teaches, in order to produce an executable form of the application. *Id.*; Ex. 1002, ¶¶215-216.

Thus, a POSITA would have understood that in addition to locating software across one or more networks, the ORB would also be used to compile applications in response to a request (arrow 1a below) and before the application was returned to the client for execution (arrow 5b below).



Ex. 1009, Fig. 2 (annotated); Ex. 1002, ¶¶217-220. Because a system could not pre-compile every piece of code on the Internet, a POSITA would have understood that compilation would occur on an as-needed basis, *i.e.*, in response to a request for that application. Ex. 1002, ¶¶214-220.

Fowlow, therefore, teaches this limitation. Ex. 1002, ¶221.

- d. *[11c] transforming the compiled application into a transformed form of the compiled form of the application, wherein transforming comprises execution and compression of the compiled form;*

Fowlow describes transforming the compiled applications at the ORB prior to sending them to the requesting client using “filter 40” on ORB 11, which “may perform a variety of tasks, such as compression, encryption, tracing, or debugging, that are applied to communications to and from an object.” Ex. 1009, 6:5-8. Fowlow’s filter 40 expressly discloses transforming the compiled application by compression, which happens after the application is compiled (as a POSITA would understand the transmitted application would already be compiled, such that Fowlow compresses the compiled, not pre-compiled, code). *Id.*; Ex. 1002, ¶¶230, 237.

Fowlow’s filter 40 also discloses tracing and debugging, which a POSITA would have understood required executing the compiled application in order to locate and remove bugs in the compiled code. Ex. 1002, ¶¶224-229, 233-236, 134 (“A ‘debugger’ is ‘[a] program designed to aid in debugging another program by allowing the programmer to *step through the program*, examine the data, and monitor conditions such as the values of variables.’”), 135 (“Tracing is sometimes called “monitoring” or “logging.” Programmers have used such techniques to *analyze source code and running programs*, looking for bottlenecks, which in turn

allow them to find and fix bugs, as well as to optimize the code (*e.g.*, to run faster, to consume less memory).”).

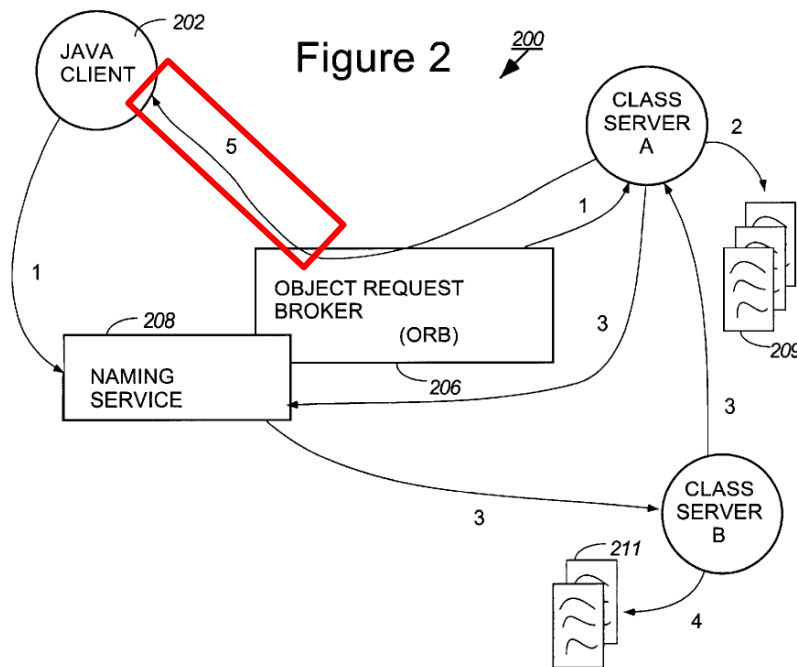
In addition, Fowlow discloses applets made up of “byte codes representing classes” are “executed by a remote interpreter” and that these classes are “compiled before execution.” Ex. 1009, 10:50-56. A POSITA would have understood the classes that are “compiled before execution” and then “executed by a remote interpreter” renders obvious the claimed “transforming the compiled application..., wherein transforming comprises execution....” *Id.*; Ex. 1002, ¶¶231-233.

Finally, a POSITA would have understood filter 40 and the remote interpreter would be used to transform the application after it is compiled but before the application is sent to the client. Ex. 1002, ¶¶238-240. For example, if portions of the application were compressed prior to compilation, those portions would need to be uncompressed to compile the application. *Id.*, ¶¶241-242. Similarly, executing only portions of the application prior to compilation may not catch errors or issues with executing the whole, compiled application. *Id.* Thus, a POSITA would have understood the transforming operations take place after compilation but before being sent to the client. *Id.*, ¶¶240-242.

Fowlow, therefore, teaches this limitation. Ex. 1002, ¶243.

- e. [11d] sending the transformed form of the application to the client computer.

Fowlow discloses sending the requested application from the ORB to the client. For example, Fowlow explains that once the applet server retrieves the desired applet execution code it is “able to return this portion of applet execution code to the client *by way of the object request broker.*” Ex. 1009, 3:38-42. Similarly, as shown in Figure 2, Fowlow teaches that the “class code corresponding to the class name desired by the Java client is finally returned [to the client] via the ORB.” Ex. 1009, 12:6-9.



Id., Fig. 2 (annotated); Ex. 1002, ¶244. A POSITA would have understood the application sent to the client could be the same application that is transformed by the ORB, for example using the transforming operations of filter 40. Ex. 1002, ¶245;

Ex. 1009, 6:5-8, 10:50-56, 11:21-35. Fowlow, therefore, teaches this limitation. Ex. 1002, ¶246.

This claim is rendered obvious by Fowlow. Ex. 1002, ¶247.

2. Dependent Claims 12-19

- a. 12. The method of claim 11 wherein the identified form of the application is specified using a Universal Resource Locator.*

Fowlow discloses that it was known that “the location of [the] Java classes [that comprise an application and that are located] within the computer network are identified using a Universal Resource Locator (URL) address.” Ex. 1009, 2:13-32. Further, the “HTML page data that a user may acquire through [a browser] contains references to applet classes that may be used to execute small programs in parts of the page” and the “code for the applet...is found by reference to a specific URL address that identifies a particular computer.” *Id.*, 2:37-54. A POSITA would have understood the request would identify the “hard-wired” file name of the requested application using a URL. *Id.*; Ex. 1002, ¶248.

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

- b. 13. The method of claim 11 wherein the identified form of the application is specified using a cookie.*

Fowlow teaches “the form of the request [for an application] will depend to a large extent upon the nature of the programming language used to create the client.”

Ex. 1009, 7:11-23, 10:28-34. Fowlow also teaches an implementation repository on the ORB that “is used to store information relating to object servers” including “information such as...any arguments to the [server] program, and any environment variables to pass to the program, etc.” *Id.*, 6:61-67. A POSITA would have understood the arguments and environment variables stored in the implementation repository of the ORB, which are used to identify the requested application, and more specifically the form of the application, would be a form of cookie. Ex. 1002, ¶¶250-255, 66-71; Ex. 1017, 1; Ex. 1014, 119.

Fowlow renders obvious this claim. Ex. 1002, ¶256.

c. 14. The method of claim 11 wherein the identified form of the application is specified in a HTTP header.

Fowlow discloses that “a browser [client] typically uses the hypertext transfer or ‘http’ protocol” to request data, such as Java classes, stored on servers. Ex. 1009, 2:22-25, 7:11-23, 10:49-50; Ex. 1002, ¶¶257-258. Because, as Fowlow teaches, clients “typically use[] the” HTTP protocol, a POSITA would have understood that it would have been obvious for the requested application to be identified in the HTTP header. Ex. 1009, 2:22-25; Ex. 1002, ¶¶63-65, 257-258; Ex. 1014, 305.

Fowlow renders obvious this claim. Ex. 1002, ¶259.

- d. 15. The method of claim 11 wherein the identified form of the application is a web page.*

Fowlow describes a method of locating and downloading applications for use on a client machine. Ex. 1009, Abstract, 10:15-65. These applets, according to Fowlow, “may refer to a wide variety of types of execution code.” *Id.*, 10:45-56. One such example of a type of execution code is a set of Java classes that “can be embedded in another *application such as an HTML document* in order to provide interactive, executable content on a *Web page*.” *Id.*, 10:56-62. A POSITA would have understood that the requested application could be an HTML document, *i.e.*, a web page, that may additionally include embedded applets. *Id.*, 2:34-40; Ex. 1002, ¶¶260-261, 72-77.

Fowlow renders obvious this claim. Ex. 1002, ¶262.

- e. 16. The method of claim 11 Wherein the identified form of the application is a portion of a web page.*

Fowlow discloses “the HTML page data that a user may acquire through Netscape contains references to *applet classes* that may be used to execute small programs *in parts of the page*. Thus, Netscape would be directed to locate and download the code for the applet to run in the frame that the page defines.” Ex. 1009, 2:37-47, 4:3-15. Fowlow also states “applet may mean in particular a small Java program that can be embedded in another application such as an HTML document in order to provide interactive, executable content on a Web page.... The

presentation of a particular object may be used to run an executable program in a small window within a larger window.” *Id.*, 10:59-62. A POSITA would have understood “a portion of a web page” to include a small Java applet program embedded in part of a web page. Ex. 1002, ¶¶263-264, 72-80.

Fowlow renders obvious this claim. Ex. 1002, ¶265.

f. 17. The method of claim 11 wherein transforming further comprises verification of the compiled form.

Fowlow describes transforming the applications at the ORB prior to sending them to the requesting client using “filter 40” on ORB 11, which “may perform a variety of tasks, such as compression, encryption, tracing, or debugging, that are applied to communications to and from an object.” Ex. 1009, 6:5-8. Fowlow teaches verification of the compiled form because a POSITA would have understood encryption and debugging, performed on the compiled form of the application, to be a form of verification. Ex. 1002, ¶¶266-268, 128-134. For example, a POSITA would have recognized that the filter would analyze the compiled application code to ensure it meets the requisite level of encryption and was error free, and therefore safe. *Id.* This is consistent with the teachings of the ’740 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.” Ex. 1001, 5:29-31; Ex. 1002, ¶268.

While filter 40 and transport layer 38 are detailed in relation to Figure 1a, Fowlow expressly confirms the “object request broker, or ORB, [of Figure 2] may be implemented as described above with reference to FIGS. 1a, 1b, and 1c.” *Id.*, 11:21-35. After being transformed by the ORB, Fowlow explains that the application is “finally ***returned*** via the ORB 206 ***to the Java client 202***” as shown by Arrow 5 in Figure 2. Ex. 1009, 11:63-12:9; Ex. 1002, ¶269. Thus, a POSITA would have understood the transformation, including verification, takes place before the application is sent to the client computer. Ex. 1002, ¶269.

Fowlow, therefore, teaches this claim. Ex. 1002, ¶270.

- g. 18. The method of claim 11 wherein transforming further comprises optimization of the compiled form.*

Fowlow describes transforming the applications at the ORB prior to sending them to the requesting client using “filter 40” on ORB 11, which “may perform a variety of tasks, such as compression, encryption, tracing, or debugging, that are applied to communications to and from an object.” Ex. 1009, 6:5-8. Fowlow teaches optimization of the compiled form because a POSITA would have understood that tracing and debugging suggest optimizing the application received from the network at least because both functions have long been used to analyze programs and source code for the purpose of optimization. Ex. 1002, ¶¶271-274, 119, 134-135. The ’740 patent describes “optimizers” as “transformer programs [that] analyze code, making

improvements to well known code fragments by substituting in optimized but equivalent code fragments.” Ex. 1001, 6:2-5. In tracing and debugging the compiled application code, a POSITA would have understood this suggests making improvements to the code, for example, to eliminate excessive or incorrect class calls. Ex. 1002, ¶¶274-275. A POSITA would have been motivated to include the filter on the ORB in Figure 2 to provide efficient, secure, and reliable applications to the client. Ex. 1009, 6:5-8; Ex. 1002, ¶¶275, 125, 134.

A POSITA would have understood the transformation, including optimization, takes place before the application is sent to the client computer. Ex. 1009, 11:63-12:9; Ex. 1002, ¶¶275, 241 (“any transforming actions would occur at the ORB after the application has been compiled, but before the application was sent to the client”).

Fowlow renders obvious this claim. Ex. 1002, ¶276.

- h. 19. The method of claim 11 wherein the transformed form of the application is sent to the client computer in an HTTP response.*

As discussed in Section V.C.2.c, Fowlow discloses HTTP requests. Ex. 1009, 2:22-25, 7:11-23, 10:49-50; Ex. 1002, ¶¶257-259. A POSITA would have understood that the http exchange, and in particular, “[t]he http server...send[ing] classes...back to the requesting browser for execution” involves sending the

requested, transformed application to the client in an HTTP response. *Id.*; Ex. 1002, ¶¶277-278, 55, 62.

Fowlow renders obvious this claim. Ex. 1002, ¶279.

D. Ground 2: Claims 11-21 are Unpatentable Under 35 U.S.C. §103(a) as Obvious over Fowlow in view of Kimera

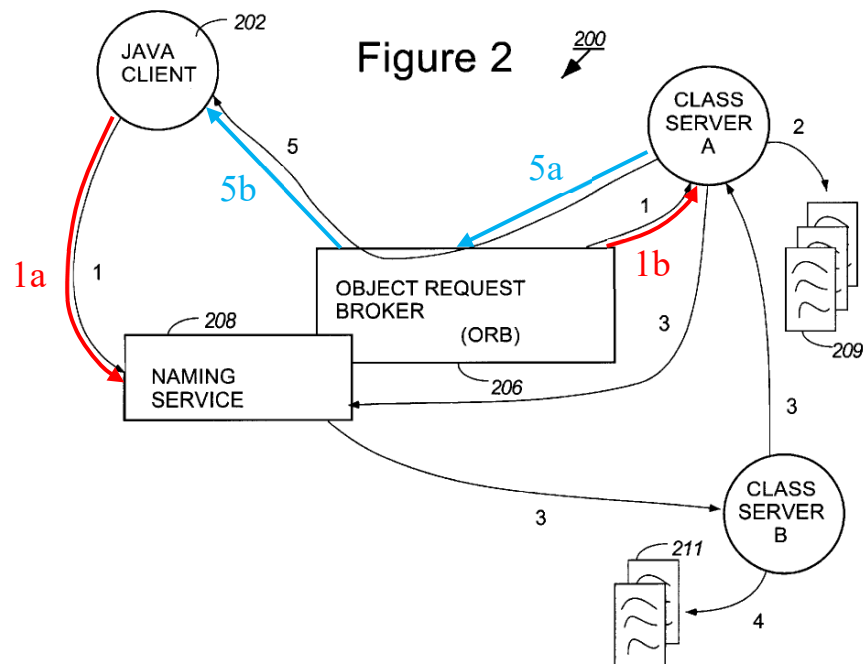
As detailed in Sections V.C.1 and V.C.2, Fowlow renders claims 11-19 obvious. For example, a POSITA would have understood Fowlow's ORB would have processing capability to compile and transform applications, as recited in claim 11. Ex. 1009, Abstract, 3:27-4:2, 6:5-8, 10:50-56, 11:57-12:9, Fig. 2; Ex. 1002, ¶¶214-243, 134, 135. To the extent the Board finds Fowlow's disclosure insufficient as to any of the application processing limitations, a POSITA would have readily looked to combine Fowlow with Kimera, which render obvious claims 11-21. Ex. 1002, ¶280.

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

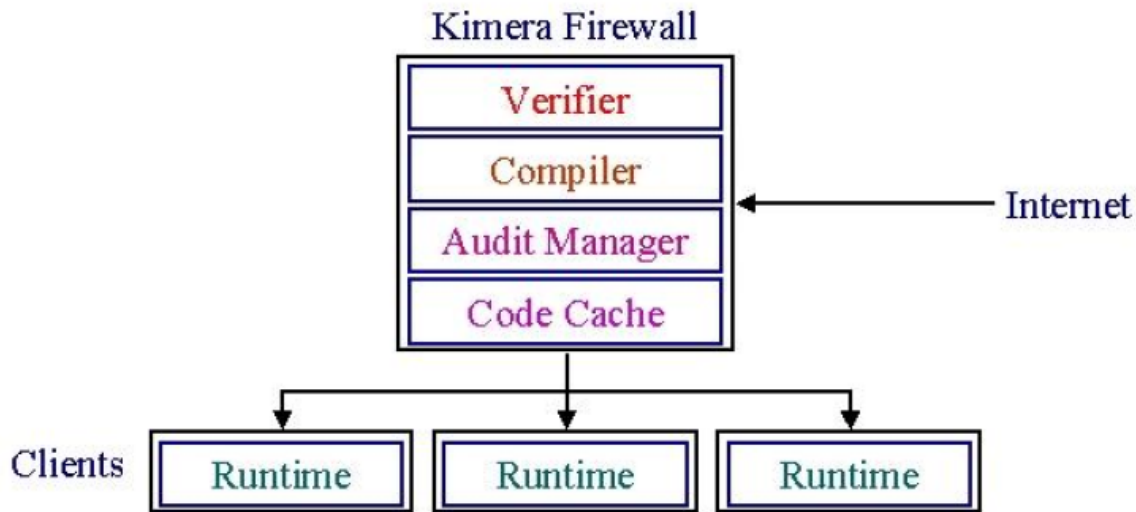
software capable of running on client devices from a remote network and are analogous art to the '740 patent and to each other. Ex. 1002, ¶¶281-285.

Fowlow teaches a loop involving the client device, an ORB, and one or more servers; the loop 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, ¶¶286-287.

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.

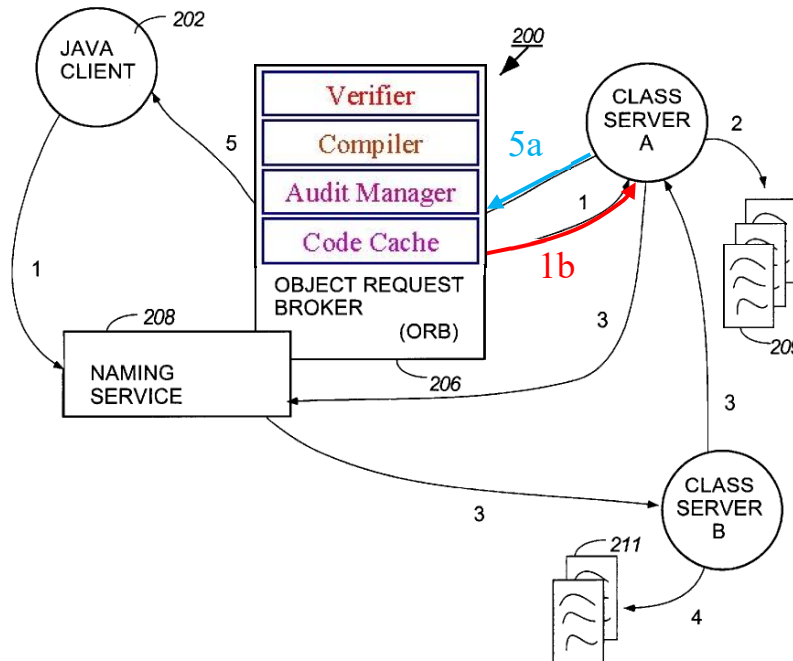


Ex. 1008, 1; Ex. 1002, ¶288.

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, ¶¶288-290.

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



Ex. 1009, Fig. 2 (annotated); Ex. 1008, 1; Ex. 1002, ¶291. This eliminates extra communication steps between the client and server and minimizes the need for compilers and verifiers on client devices, which a POSITA would have understood would make the overall system faster, cheaper, and more secure. Ex. 1002, ¶¶290-291 (citing Ex. 1008, 1-2).

Locating a verifier in the modified ORB further advances the goals outlined in Fowlow, namely, that the client “knows a base class this it wishes to load but is mercifully unaware of all the machinations behind the scenes required to find other classes used or needed by this base class.” Ex. 1009, 10:26-34. Once the complete applet has been compiled at the modified ORB, the result would be cached in the “Code Cache” of the modified ORB to “further reduce the latency of binary translation” in response to future requests for the same code. Ex. 1008, 2; Ex. 1002, ¶¶292-293. This desire to reduce latency further would contribute to the optimization of the cached code. Ex. 1002, ¶¶292-293.

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, ¶294. 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, particularly 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, ¶294.

A POSITA would have been motivated to combine Fowlow and Kimera. *Id.*, ¶¶281-294.

2. Independent Claim 11

a. [11pre]

To the extent the preamble is limiting, Fowlow in view of Kimera teaches it as discussed in Section V.C.1.a. Ex. 1009, Abstract, 5:7-13, 5:34-69, 10:15-34, Fig. 2; Ex. 1002, ¶¶295, 205-208.

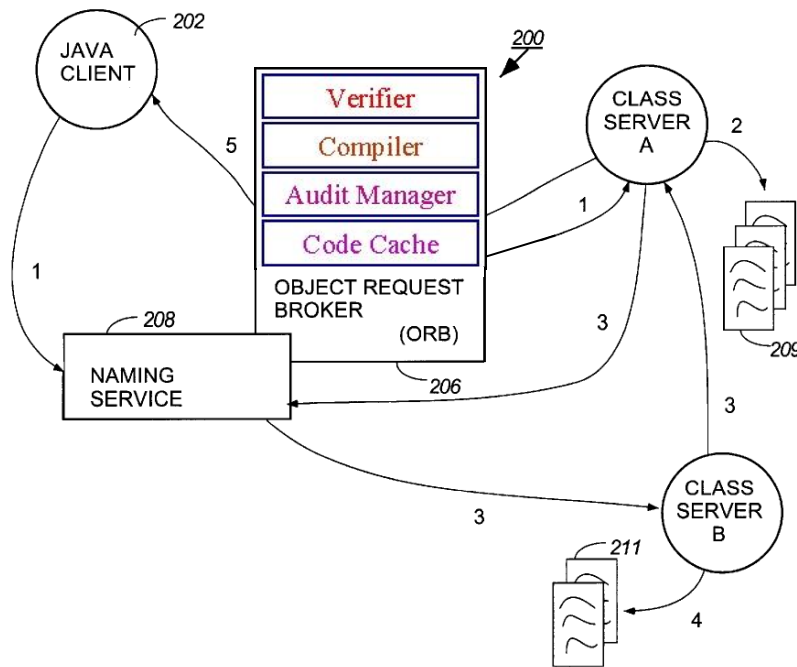
b. [11a]

Fowlow in view of Kimera teaches this limitation as discussed in Section V.C.1.b. Ex. 1009, Abstract, 1:44-57, 3:27-42, 4:3-15, 7:11-23, 10:18-34, 10:45-56, 11:21-30, Fig. 2; Ex. 1001, 3:19-23; Ex. 1002, ¶¶296-302, 209-213.

c. [11b]

Fowlow in view of Kimera teaches this limitation as discussed in Section V.C.1.c. Ex. 1009, Abstract, 3:27-4:2, 10:50-56, 11:57-12:9, Fig. 2; Ex. 1002, ¶¶303, 214-221. 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, ¶¶304-305.

A POSITA would have understood that the compiler of Kimera acts on a source module (*e.g.*, Java bytecode) to produce the identified form (*e.g.*, binary or “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, ¶306. This disclosure comports with the ’740 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, 4:37-39, 5:14-17. A POSITA would have been motivated to combine Fowlow with the teaching of Kimera for the reasons identified in Section V.D.1.

Kimera also includes a code cache, which “cach[es] the results of recent compilations in the central compilation server.” Ex. 1008, 2. A POSITA would have understood the *recent* compilation results would be the results of a compilation made in response to recent client requests. Ex. 1002, ¶306. For example, a system would not pre-compile every piece of code on the Internet and store that pre-compiled code in a cache. *Id.* Instead, a POSITA would understand that the system would, in response to the incoming request that identifies the form of the application, compile the application into that form. *Id.*, ¶¶306-307; Ex. 1008, 1, 2. Therefore, a POSITA would have understood that in this arrangement, the only practical use of Kimera involves a client requesting code (*e.g.*, Java applets described in Fowlow) located on the Internet, receiving the code into the Kimera centralized server, compiling the code with the centralized compiler (as well as transforming the code as discussed below), and transmitting the requested application to the client. *Id.* Once compiling is complete, Kimera teaches “caching the results of [these] recent compilations in the central compilation server.” Ex. 1008, 2.

Additionally, the Kimera firewall mirrors the server computer of the ’740 patent, which similarly compiles code before transmitting such code to the client.

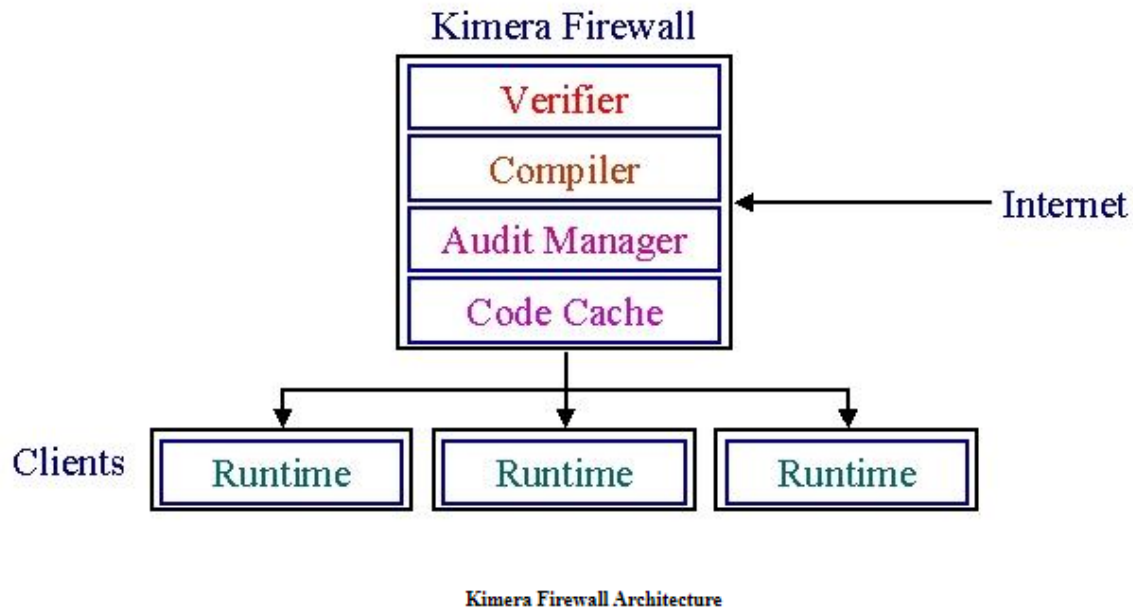
Ex. 1001, 3:40-45, Fig. 1; Ex. 1002, ¶¶308-309. Both serve as an interface between client computers and external networks, such as the Internet. *Compare* Ex. 1008, 1 (“[L]ocating verification, access control, auditing, and compilation services in centralized servers enables the same services to be performed faster, cheaper, and without introducing security vulnerabilities” and allows client devices to “be constructed from cheaper components”) *with* Ex. 1001, 3:40-45 (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, ¶¶308-309.

Fowlow-Kimera renders obvious this limitation. Ex. 1002, ¶310.

d. [11c]

Fowlow in view of Kimera teaches this limitation as discussed in Section V.C.1.d. Ex. 1009, 6:5-8, 10:50-56; Ex. 1002, ¶¶311-312, 222-243, 134, 135.

Adding the functionality of Kimera’s firewall to Fowlow’s ORB enhances the ORB’s processing of the applets by bringing additional application transforming 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.



Id., 1; Ex. 1002, ¶¶313-315, 326.

The Kimera architecture includes several components specifically designed to transform applications received from the Internet. For example, Kimera’s “approach to Java system structure is to break the trusted computing base into small, secure components and locate them in security firewalls.” Ex. 1008, 1. To consolidate resources and reduce client requirements, Kimera discusses consolidating processing functionality into a centralized server, which allows clients to exist as a runtime environment. *Id.*, 2. To avoid excessive communication overhead between client and server, a POSITA would have found it obvious to include compression on all applet files sent to the client, which would reduce the size of communications flowing between client and server. *Id.*; Ex. 1002, ¶¶320-322.

Kimera also teaches a “verification service” within the modified ORB that “performs checking of incoming code for typesafety and conformance to interfaces.” Ex. 1008, 2. A POSITA would have understood Kimera’s verification service could have executed the “incoming code” to ensure the code did not introduce security vulnerabilities. *Id.*, 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”), 2; Ex. 1002, ¶¶316-318, 323-325, 133. Kimera also teaches an auditing service on the centralized server. A POSITA would have understood auditing the application code could require executing the application code. Ex. 1002, ¶¶319, 135 (“Tracing is sometimes called ‘monitoring’ or ‘logging.’ Programmers have used such techniques to analyze source code and running programs, looking for bottlenecks, which in turn allow them to find and fix bugs, as well as to optimize the code (*e.g.*, to run faster, to consume less memory).”). These additional safety features of Kimera are further examples of the obviousness of the claimed “transforming comprises execution...” element.

Fowlow-Kimera renders obvious this limitation. Ex. 1002, ¶327.

e. [11d]

Fowlow in view of Kimera teaches this limitation as discussed in Section V.C.1.e. Ex. 1009, 3:38-42, 6:5-8, 10:50-56, 11:21-35, 12:6-9, Fig. 2; Ex. 1002, ¶¶328-329, 244-247.

This claim is rendered obvious by Fowlow-Kimera. Ex. 1002, ¶330.

3. Dependent Claims 12-21

a. Claim 12.

Fowlow in view of Kimera renders this claim obvious as discussed in Section V.C.2.a. Ex. 1009, 2:13-32, 2:37-54; Ex. 1002, ¶¶248-249, 331.

b. Claim 13.

Fowlow in view of Kimera renders this claim obvious as discussed in Section V.C.2.b. Ex. 1009, 6:61-67, 7:11-23, 10:28-34; Ex. 1002, ¶¶66-71, 250-256, 332.

c. Claim 14.

Fowlow in view of Kimera renders this claim obvious as discussed in Section V.C.2.c. Ex. 1009, 2:22-25, 7:11-23, 10:28-34, 10:49-50; Ex. 1002, ¶¶63-65, 257-259, 333.

d. Claim 15.

Fowlow in view of Kimera renders this claim obvious as discussed in Section V.C.2.d. Ex. 1009, Abstract, 2:34-40, 10:15-65; Ex. 1002, ¶¶72-77, 260-262, 334.

e. Claim 16.

Fowlow in view of Kimera renders this claim obvious as discussed in Section V.C.2.e. Ex. 1009, 2:37-47, 4:3-15; Ex. 1002, ¶¶72-80, 263-265, 335.

f. Claim 17.

Fowlow in view of Kimera renders this claim obvious as discussed in Section V.C.2.f. Ex. 1009, 6:5-8, 11:21-35, 11:63-12:9; Ex. 1001, 5:29-31; Ex. 1002, ¶¶128-134, 266-270, 336. Kimera also teaches a “verification service” within the modified ORB that “performs checking of incoming code for typesafety and conformance to interfaces.” Ex. 1008, 2. A POSITA would have understood transforming the compiled form would take advantage of Kimera’s verification service, which would perform verification of the compiled form of the application. Ex. 1002, ¶¶337-339.

Fowlow-Kimera, therefore, teaches this claim. Ex. 1002, ¶340.

g. Claim 18.

Fowlow in view of Kimera renders this claim obvious as discussed in Section V.C.2.g. Ex. 1009, 6:5-8; Ex. 1001, 6:2-5; Ex. 1002, ¶¶119, 125, 134-135, 241, 271-276, 341. Kimera further explains the benefits of the centralized architecture utilized in both Fowlow and Kimera, which is that it “enables the same services [*e.g.*, verification, access control, auditing, and compilation services] to be performed *faster*, cheaper, and without introducing security vulnerabilities.” Ex. 1008, 1. Kimera continues, describing additional benefits include how the centralized server

“makes a sizable reduction in computation and memory requirements of the endpoints” and that “Java enabled network computers [clients] could [] be constructed from cheaper components.” *Id.*, 2; Ex. 1002, ¶343.

A POSITA would have understood that this is a form of optimizing the applets from the Java bytecode before sending the one or more applets to the one or more client computers because it results in faster services performed on the most inexpensive clients possible. Ex. 1002, ¶¶342, 344, 114-119; Ex. 1014, 343 (defining “optimization” as “1. In programming, the process of producing more efficient (smaller or faster) programs through selection and design of data structures, algorithms, and instruction sequences. 2. The process of a compiler or assembler in producing efficient executable code.”). Additionally, compiling and caching applets on the centralized server would reduce communication overhead, optimizing the applet retrieval process before applets are sent to client computers. Ex. 1002, ¶345.

Fowlow-Kimera, therefore, teaches this claim. Ex. 1002, ¶346.

h. Claim 19.

Fowlow in view of Kimera renders this claim obvious as discussed in Section V.C.2.h. Ex. 1009, 2:22-25, 7:11-23, 10:49-50; Ex. 1002, ¶¶55, 62, 257-259, 277-278, 347.

i. 20. The method of claim 11, further comprising:

- i. [20a] storing a plurality of available forms of the application;*

Kimera's architecture compiles Java bytecode (*i.e.*, the source module) to produce the identified form of an application (*i.e.*, the native machine code). Ex. 1008, 2; Ex. 1002, ¶349. 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 storing a plurality of available forms [*e.g.*, recent compilations] of the application. Ex. 1002, ¶¶348-350.

Fowlow-Kimera, therefore, teaches this limitation. Ex. 1002, ¶351.

- ii. [20b] the server, in response to receiving the request, determining if the identified form of the application is one of the stored plurality of available forms of the application; and*

If a request identified an application and a form of the application, and that application in the identified form was stored in the modified ORB's code cache, a POSITA would have understood that it would be returned to the client (arrow 5b below) from the code cache. Ex. 1009, Fig. 2; Ex. 1008, 2; Ex. 1002, ¶¶352-353. 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

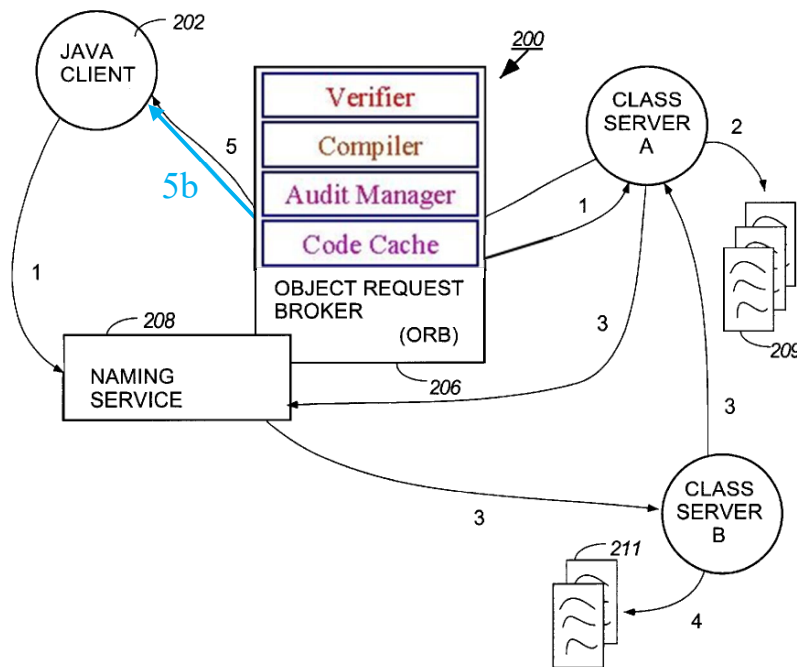
binary translation.” Ex. 1008, 2. A POSITA would have understood that latency would be reduced because compiled and cached application code could be transmitted directly to the client computer. *Id.*; Ex. 1002, ¶353.

To the extent that an identified form of an application is not a recent compilation or otherwise not stored in the local cache, a POSITA would have understood that the Fowlow-Kimera system transmits a request to an external resource on the broader distributed object system for the application in the identified form. Ex. 1009, 3:32-42, 10:21-34, 11:63-12:6; Ex. 1008, 2; Ex. 1002, ¶353.

Fowlow-Kimera, therefore, teaches this limitation. Ex. 1002, ¶354.

- iii. [20c] if the identified form of the application is one of the stored plurality of available forms, sending the identified form of the application to the client computer without compiling and transforming.

If the identified form of the requested application was stored in the modified ORB’s code cache, it would be returned to the client from the code cache (arrow 5b below) without any additional processing.



Ex. 1002, ¶¶352, 355-357; *see also* Ex. 1009, Fig. 2; Ex. 1008, 2. 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 application code could be transmitted directly to the client computer without compiling and transforming that application. Ex. 1002, ¶¶357-360.

Fowlow-Kimera, therefore, teaches this limitation and each limitation of this claim. Ex. 1002, ¶¶361-362.

j. 21. The method of claim 11, further comprising:

- i. [21a] storing a plurality of available forms of the application;

Fowlow in view of Kimera teaches this element as described in Section V.D.3.i.i. Ex. 1008, 2; Ex. 1002, ¶¶348-351, 363.

- ii. [21b] the server, in response to receiving the request, determining if the identified form of the application is one of the stored plurality of available forms of the application; and

Fowlow in view of Kimera teaches this element as described in Section V.D.3.i.ii. Ex. 1009, 3:32-42, 10:21-34, 11:63-12:6, Fig. 2; Ex. 1008, 2; Ex. 1002, ¶¶352-354, 364.

- iii. [21c] if the identified form of the application is one of the stored plurality of available forms, sending the identified form of the application to the client computer without compiling.

Fowlow in view of Kimera teaches this element as described in Section V.D.3.i.iii, which claims “sending the identified form of the application to the client computer without compiling and transforming.” Ex. 1009, Fig. 2; Ex. 1008, 2; Ex. 1002, ¶¶352, 355-362, 365-366.

Fowlow-Kimera, therefore, renders obvious claim 21. Ex. 1002, ¶367.

E. Ground 3: Claims 11-12, 14-21 are Unpatentable Under 35 U.S.C. §103(a) as Obvious over Parthasarathy in view of Kimera

Claims 11-12, 14-21 are obvious in view of the combination of Parthasarathy and Kimera. Ex. 1002, ¶368.

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, ¶¶369-371. 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 ’740 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, ¶372. 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, ¶¶372-374; *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, ¶375. 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, ¶¶374-375. 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, ¶376. Thus, the Parthasarathy-Kimera combination provides a centralized server, centralized as to all local computers in the network, 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, ¶¶377-382.

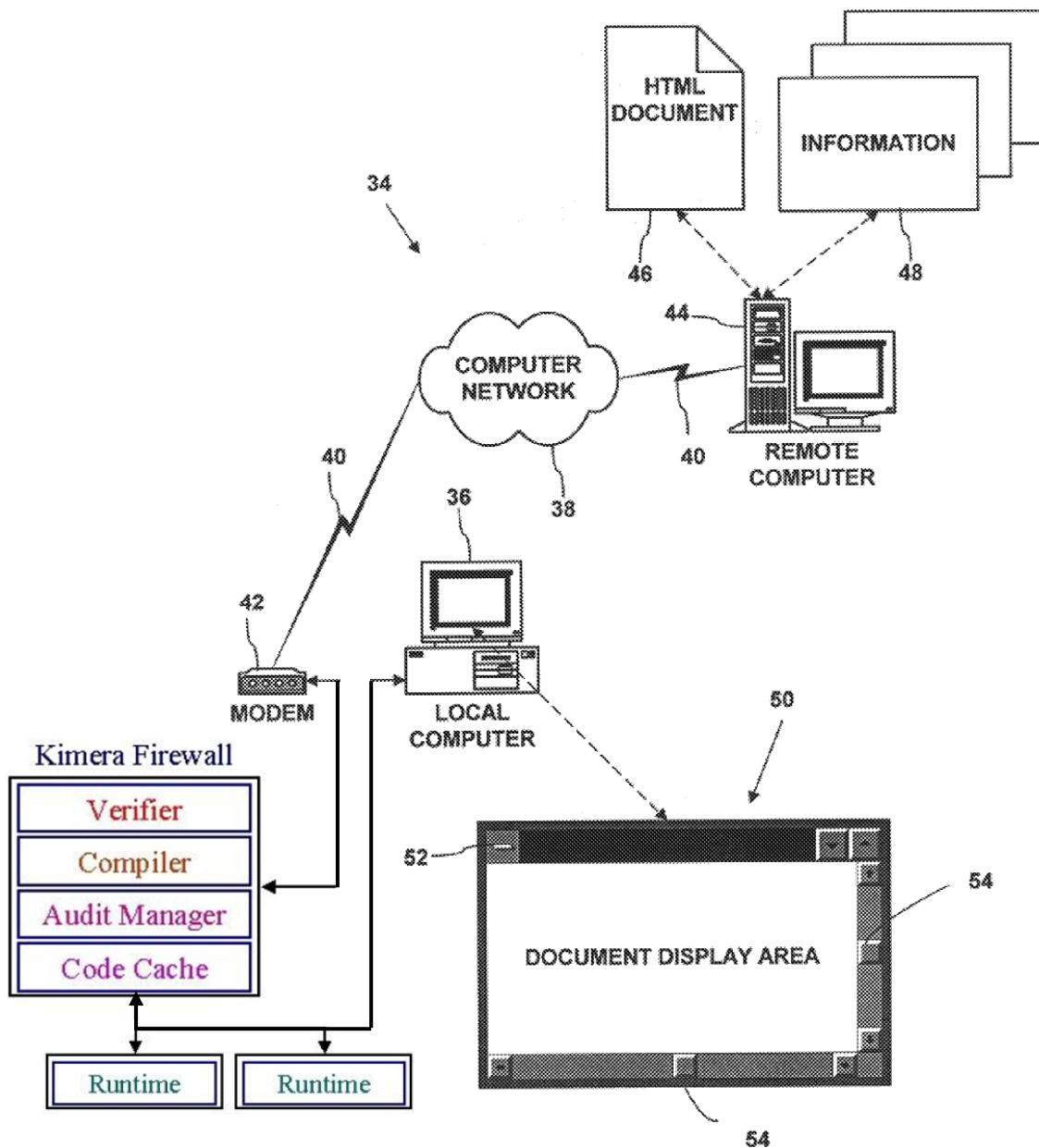


FIG. 2

Ex. 1007, Fig. 2 (annotated); Ex. 1008, 1; Ex. 1002, ¶376.

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 a 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, ¶¶377-380.

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, ¶381. 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.*, ¶¶377-382. 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.*, ¶¶371, 378-379. 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.*, ¶382; Ex. 1008, 2.

A POSITA would have been motivated to combine Parthasarathy and Kimera to arrive at the claimed method. Ex. 1002, ¶¶369-383.

2. Independent Claim 11

a. [11pre]

To the extent the preamble is limiting, Parthasarathy-Kimera teaches it. Parthasarathy discloses “network browsing environment” that operates to manage requests for applications 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, ¶¶384-386.

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 V.E.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, ¶¶374-375, 386-388. 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:

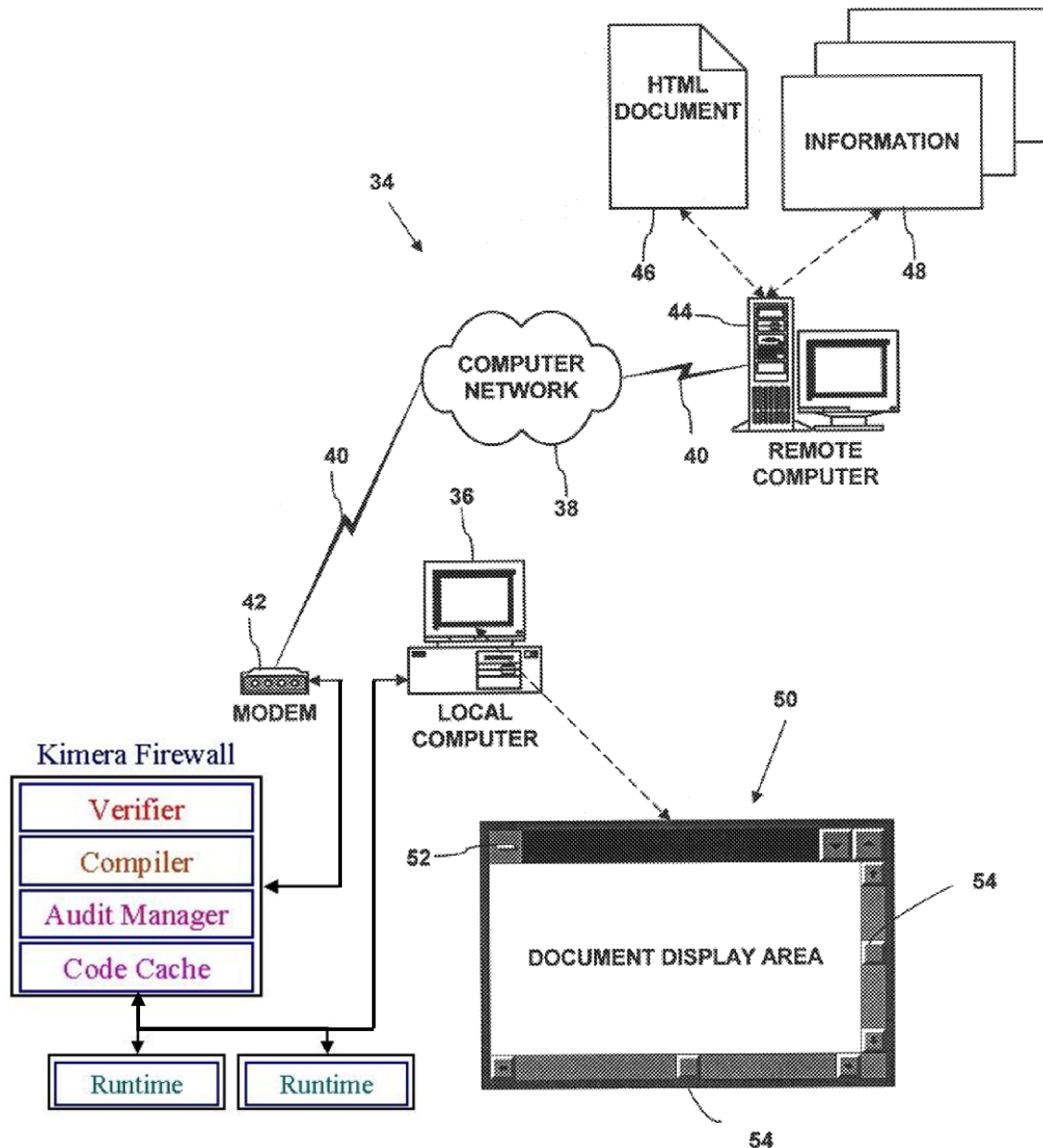


FIG. 2

Ex. 1007, Fig. 2 (annotated); Ex. 1008, 1; Ex. 1002, ¶¶376, 387-388. A POSITA would have understood this centralized server would have managed all requests for

applications, at least in order to determine whether the centralized server's 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, ¶¶385-388.

To the extent the preamble is limiting, a POSITA would have found it taught by Parthasarathy-Kimera. Ex. 1002, ¶389.

b. [11a]

Parthasarathy-Kimera teaches receiving a request for an application 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 V.E.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, ¶¶390-393, 377.

Parthasarathy further teaches that the request for a software component (*i.e.*, application) identifies the form of the application from a plurality of available forms. Ex. 1002, ¶394. 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 application 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, ¶¶394-398.

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, ¶¶394-397. 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, ¶397.

In another example, an HTTP ACCEPT header MIME request identifies 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 identifies a platform-specific application from a plurality of available application forms. Ex. 1002, ¶398.

Parthasarathy-Kimera, therefore, teaches this limitation. Ex. 1002, ¶399.

c. [11b]

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. *Id.*, 2. A POSITA would have understood the *recent* compilation results would be the results of a compilation made in response to recent client requests because the system would not pre-compile every piece of code on the Internet and store that pre-compiled code in a cache. Ex. 1002, ¶¶400-401.

A POSITA would have further understood that the compiler of Kimera compiles source module (*e.g.*, Java bytecode and other cached recent compiled modules) to produce the identified form of the application (*i.e.*, binary or “native machine code”). Ex. 1008, 2; *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, ¶400. Because the operating system and CPU of the requesting client machine (or as Kimera describes the “native machine code”) would have been known, the act of compiling the application into a compiled form could not occur before the request was made identifying the requesting client by OS and CPU. Ex. 1002, ¶¶400-401.

A POSITA would have understood that the identified form of the application would have been compiled into a compiled form in response to receiving a request because the request would define the form of the application needed for a particular operating system and CPU combination (Ex. 1007, 18:35-57) or to run natively on the client (Ex. 1008, 2). Ex. 1002, ¶400. Thus, receipt of the request is a prerequisite to compilation. *Id.*, ¶¶400-401.

This disclosure comports with the disclosure of the '740 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, 4:37-39, 5:14-17 (“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.”); Ex. 1002, ¶402.

A POSITA would have been motivated to combine Parthasarathy and Kimera for the reasons enumerated in Section V.E.1. Ex. 1007, Abstract, 2:67-3:14, 8:13-31, 8:66-9:8, 9:33-42, 23:39-56, Figs. 2, 4; Ex. 1008, 1, 2; Ex. 1002, ¶¶369-383.

Parthasarathy-Kimera, therefore, teaches this limitation. Ex. 1002, ¶403.

d. [11c]

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. A POSITA would have understood Kimera’s verification and auditing services could have executed the “incoming code” to ensure the code did not introduce security vulnerabilities. *Id.*, 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”), 2; Ex. 1002, ¶¶404-410, 133 (“When programs run, they can verify their own safety; the operating system and system libraries often perform additional verifications.”), 135 (“Tracing is sometimes called ‘monitoring’ or ‘logging.’ Programmers have used such techniques to analyze source code and running programs, looking for bottlenecks, which in turn allow them to find and fix bugs, as well as to optimize the code (*e.g.*, to run faster, to consume less memory).”).

Parthasarathy in view of Kimera also teaches transforming compiled applications through compression. Ex. 1002, ¶¶411-413. Parthasarathy discloses a “cabinet file [which] contains one or more files, all of which are downloaded together in a *compressed* ‘cabinet.’” Ex. 1007, 15:65-16:1. Compressing the one or more applications before sending them to the client was, therefore, well known. Ex. 1002, ¶¶411, 126-127; Ex. 1014, 107 (compression), 130 (data compression).

Application 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 centralized server 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, ¶412. Compiling and caching applications on the centralized server would also reduce communication overhead, further optimizing the application retrieval process. Ex. 1002, ¶¶413-417, 116; Ex. 1018, 43; Ex. 1014, 343.

Parthasarathy-Kimera, therefore, teaches this limitation. Ex. 1002, ¶418.

e. [11d]

The local computer of Parthasarathy can request “to download a desired software component stored on [the] remote computer 44.” Ex. 1007, 8:19-20. In response to receiving a request, the software component is downloaded to the local computer and the “software components are returned 78 to the client application (*e.g.*, the network browser) which made the request for the computer software components.” *Id.*, 29:13-20. In the context of the combination with Kimera, the

software would be sent to the client from the centralized server, as described above with reference to elements [1a] and [1e]. Ex. 1002, ¶¶419-420.

Parthasarathy-Kimera, therefore, teaches this limitation. Ex. 1002, ¶421.

Thus, Parthasarathy-Kimera renders obvious claim 11. Ex. 1002, ¶422.

3. Dependent Claims 12, 14-21

a. Claim 12.

Parthasarathy discloses systems and methods to “locate, download, verify, install, register and display computer software components from a computer network.” Ex. 1007, Abstract. Initially, “a request is made to download a desired software component stored on a remote computer 44.” *Id.*, 8:19-20. The request “us[es] one of two designators: a URL or a network search path” to identify and locate the computer software component via a control file. *Id.*, 9:9-21.

Parthasarathy also claims a “control file [that] includes an initialization file that includes one or more universal resource locators pointing to one or more computer software components...being specified for the operating system or the CPU of the local computer.” *Id.*, cl. 57; Ex. 1002, ¶¶423-426.

Parthasarathy-Kimera, therefore, renders obvious this claim. Ex. 1002, ¶427.

b. Claim 14.

Parthasarathy discloses systems and methods to “locate, download, verify, install, register and display computer software components from a computer

network.” Ex. 1007, Abstract. In the request, the local computer may identify the form of the application in an HTTP header. *Id.*, 14:65-15:13; Ex. 1002, ¶¶428-429. For example, “an Object Store on the network search path is an HTTP server that services requests for a desired control file.” *Id.*, 14:65-67. The control file would be located from the various Object Stores in the search path and “[s]pecifically, an Object Store will receive an HTTP POST request with data in the format below:

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

Id., 14:67-15:7. “The Object Store parses this information, checks an internal database, and either fails, or redirects the HTTP request to the control file on the next remote computer in the network search path.” *Id.*, 15:8-13; Ex. 1002, ¶430.

Parthasarathy also discloses queries to Object Stores that “include HTTP headers for ACCEPT (MIME type) and ACCEPT-LANGUAGE, thus specifying the desired platforms and language-localized implementation for a [software] component.” *Id.*, 15:30-38. When HTTP headers are added to HTTP requests, Object Stores can serve different code implementations based on differing platforms or even different languages. *Id.*; Ex. 1002, ¶431.

Parthasarathy-Kimera, therefore, renders obvious this claim. Ex. 1002, ¶432.

c. Claim 15.

“Documents for browsing with the illustrated browser can reside as files of a file system stored in the computer’s secondary storage 32 (FIG. 1), or reside as resources at a remote computer 44 (also referred to as a ‘site’ or ‘store’) connected to the computer network 38, such as a World Wide Web site on the Internet.” Ex. 1007 at 7:3-8. A POSITA would have understood the “document for browsing with the illustrated browser” would be HTML documents, also known as web pages. Ex. 1002, ¶433; *see also* Ex. 1007, 9:3-8.

Parthasarathy-Kimera, therefore, renders obvious this claim. Ex. 1002, ¶434.

d. Claim 16.

Parthasarathy’s local computer browses for “files of a file system stored in the computer’s secondary storage 32 (FIG. 1), or reside as resources at a remote computer 44 (also referred to as a ‘site’ or ‘store’) connected to the computer network 38, such as a World Wide Web site on the Internet.” Ex. 1007, 7:3-8. “Almost any type of computer software components can be obtained from a local or remote computer using system 56 and method 66 for many types of files or documents.” *Id.*, 9:3-8. A POSITA would have understood the “document for browsing with the illustrated browser” and additional “files or documents” would include HTML documents defining an entire web page, or a portion of a web page. Ex. 1002, ¶435; *see also* Ex. 1007, 9:3-8.

Further, the web page, may include embedded HTML <OBJECT> tags for dynamic or interactive multimedia. *Id.*, 9:63-67. The HTML <OBJECT> tags include multiple attributes, including, for example, the CODEBASE attribute, which “represent[s] the desired version of a software component on a remote server.” *Id.*, 10:4-27. By requesting software corresponding to an embedded HTML <OBJECT> tag, a POSITA would have understood that the identified form is a portion of the web page in which the tag is embedded. Ex. 1002, ¶436.

Parthasarathy-Kimera, therefore, renders obvious this claim. Ex. 1002, ¶437.

e. Claim 17.

Parthasarathy discloses systems and methods for “downloading, **verifying**, installing, registering and displaying computer software components.” Ex. 1007, Abstract. Parthasarathy discloses a “code verification module” to “verif[y] that the downloaded computer software components are safe to install on a local computer by verifying a digital signature and digital certificate sent with the downloaded computer software component.” *Id.*, 3:9-13. 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, ¶¶438-440.

Once verified, the downloaded computer software component is installed on the local computer by a code installation module. Ex. 1007, 8:35-46. A POSITA would have understood that the installed version of the application would be the transformed, *i.e.*, verified form. Ex. 1002, ¶441.

Parthasarathy-Kimera, therefore, renders obvious this claim. Ex. 1002, ¶442.

f. Claim 18.

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. To most effectively take advantage of the “sizeable reduction” to the end-points, a POSITA would have known to include well-known optimization techniques to reduce the computation and memory requirements of the requested application. *Id.*; Ex. 1002, ¶¶443-448. Put differently, a POSITA would have understood that these benefits suggest optimizing the requested applications, in its compiled form, before sending the application to the client computer in order to achieve faster services performed on the most inexpensive clients possible. Ex. 1002, ¶¶446-449, 114-119, 125.

Parthasarathy-Kimera, therefore, renders obvious this claim. Ex. 1002, ¶450.

g. Claim 19.

Parthasarathy discloses a method of downloading computer software components from a computer network using HTTP requests. Ex. 1007, 8:66-9:3,

14:65-15:13, 15:39-44. A POSITA would have understood that in the HTTP protocol, data returned to the client is sent in an HTTP response. *Id.*, 1:45-46, 15:39-44; Ex. 1002, ¶¶451-453, 55, 62.

Parthasarathy-Kimera, therefore, renders obvious this claim. Ex. 1002, ¶454.

h. Claim 20.

i. [20a]

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, ¶¶455-456. If the downloaded application 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; Ex. 1002, ¶457.

A POSITA would have understood “caching the results of recent compilations” teaches storing a plurality of available forms of the application. Ex. 1002, ¶¶457-459.

This claim is obvious in view of Parthasarathy and Kimera. Ex. 1002, ¶460.

ii. [20b]

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, ¶¶461-463. Thus, in response to receiving a request for an application in an identified form, the server would determine if the identified form is among the stored plurality of available forms of the application. *Id.*, ¶¶463-466.

Parthasarathy-Kimera, therefore, teaches this limitation. Ex. 1002, ¶467.

iii. [20c]

As mentioned above, 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, ¶¶462-464, 468-469.

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 without compiling

or transforming that application. Ex. 1002, ¶470; *see also* Ex. 1008, 2 (noting caching is used “to further *reduce the latency* of binary translation”).

Parthasarathy-Kimera, therefore, teaches this limitation and each limitation of this claim. Ex. 1002, ¶¶471-472.

i. Claim 21

i. [21a]

Parthasarathy-Kimera renders this claim obvious as described in Section V.E.3.h.i. Ex. 1007, 8:19-30, 13:42-48, 27:15-27; Ex. 1008, 2; Ex. 1002, ¶¶455-460, 473.

ii. [21b]

Parthasarathy-Kimera renders this claim obvious as described in Section V.E.3.h.ii. Ex. 1007, 8:19-30, 13:42-48; Ex. 1002, ¶¶461-467, 474.

iii. [21c]

Parthasarathy-Kimera renders this claim obvious as described in Section V.E.3.h.iii, which claims “sending the identified form of the application to the client computer without compiling and transforming.” Ex. 1007, 8:19-30, 13:42-48; Ex. 1008, 2; Ex. 1002, ¶¶462-464, 468-471, 474-476.

Parthasarathy-Kimera, therefore, teaches each limitation of this claim. Ex. 1002, ¶477.

F. Ground 4: Claim 13 is Unpatentable Under 35 U.S.C. §103(a) as Obvious over Parthasarathy in view of Kimera and Angles

Claim 13 is obvious in view of the combination of Parthasarathy, Kimera, and Angles. Ex. 1002, ¶478.

1. Motivation to Combine

Parthasarathy and Kimera are both directed to requesting software capable of running on client devices from a remote network, such as the Internet, and delivering the requested software to the client, as described in Section V.E.1. Ex. 1007, Abstract, 2:67-3:14, 8:13-31, 8:66-9:8, 9:33-42, 23:39-56, Figs. 2, 4; Ex. 1008, 1, 2; Ex. 1002, ¶¶369-383, 479-480. The requests utilize “HTTP headers for ACCEPT (MIME type) and ACCEPT-LANGUAGE [that] specify[] the desired platforms and language-localized implementation for a [software] component.” *See* Ex. 1007, 15:30-38; Ex. 1002, ¶¶481-482.

Angles similarly describes “a system and method for delivering customized electronic advertisements in an interactive communication system.” Ex. 1010, Abstract. Angles describes assigning a consumer member code used for tracking consumer registration with an advertisement provider service. *Id.*, 10:60-11:2. This member code is stored as a cookie. *Id.*, 11:2-4. The cookie is a part of the “HTTP header line” that is transmitted with all HTTP requests. *Id.*, 11:13-15; Ex. 1002, ¶¶481-484, 66-71.

To the extent Parthasarathy's disclosure of use of HTTP requests and HTTP headers did not already implicitly incorporate cookies as part of their HTTP processing, which were well known in 1996, a POSITA would have been motivated to combine at least the cookie disclosure of Angles into the Parthasarathy-Kimera system to augment the HTTP headers to more accurately and efficiently locate, download, verify, install, and register requested software components, and improve the user experience when browsing the Web. Ex. 1002, ¶¶485-487; *see also* Ex. 1010, 11:7-12. Specifically, if the HTTP headers of the Parthasarathy-Kimera system included headers, the browser could "remember some specific information across several pages." A POSITA would recognize the benefits of this functionality and would have been motivated to include cookies in the HTTP headers of requests for software components and other applications. Ex. 1002, ¶¶485-487.

Finally, in the combination, each of the respective disclosures of Parthasarathy, Kimera, and Angles function in the same manner as in their respective sources and are therefore simply a rearranging of known components in accordance with known methods to provide predictable results. *KSR*, 550 U.S. at 416; *Smith & Nephew*, 721 F.3d at 1375; Ex. 1002, ¶488.

2. Dependent Claim 13

a. Claim 13.

The Parthasarathy-Kimera system teaches the method of claim 11 for the reasons set forth in Section V.E.2. Ex. 1002, ¶¶330, 489. A POSITA would have further incorporated the cookies disclosed in Angles to enable functionality such as filling a virtual shopping cart by adding items from a virtual shopping mall, and pay for all of the items once the consumer had finished collecting their desired items. Ex. 1010, 11:7-12; Ex. 1002, ¶¶490-491 (citing Ex. 1017, 2). In this way, once a user requested an application, such as a webpage or other software component, the form of the application would be identified according to the cookie. Ex. 1010, 11:7-12; Ex. 1002, ¶¶489-492.

For example, when a user first visits a webpage and selects items to include in a virtual shopping cart, a POSITA would have understood a cookie is created and stored on the client machine to remember the contents of the shopping cart. Ex. 1010, 11:7-27; Ex. 1002, ¶¶490-491. “Whenever the browsing software sends a HTTP request for a URL on a server for which it has stored cookies, it includes a line in the form: Cookie: NAME=VALUE; NAME=VALUE;...which lists all cookies that apply to that particular URL.” Ex. 1010, 11:27-31. A POSITA would have understood that as the client moved to another webpage of the virtual shopping mall, the HTTP request sent to the centralized server and ultimately to the network

would be further specified by the cookie stored on the client device. Ex. 1002, ¶¶489, 492-493.

Parthasarathy-Kimera-Angles, therefore, renders obvious this claim. Ex. 1002, ¶494.

G. 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 any rejection in the '740 patent's prosecution. Ex. 1004, 30-32, 45-47; *see also id.*, 163. When a reference is not the basis of rejection, and is 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 '740 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.⁴

Fifth, while Petitioner is a named defendant in the parallel proceeding, there has been very little, if any, overlap of the issues because the parallel proceeding is in its infancy.

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

Balancing these factors holistically, the Board should not exercise its

⁴ 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 (PTAB Nov. 9, 2023).

discretion to deny this Petition.

VI. CONCLUSION

The Petition demonstrates a reasonable likelihood that at least one claim is unpatentable. Ex. 1002, ¶495. Accordingly, all grounds in the Petition should be instituted. *SAS Institute Inc. v. Iancu*, 138 S.Ct. 1348, 1359-60 (2018).

Respectfully submitted by

K&L GATES LLP,

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

VII. CLAIM APPENDIX OF THE CHALLENGED CLAIMS

- [11pre] A method operating on a computer system, having a client computer and a server computer, for managing requests to the server computer, the method comprising:
 - [11a] at the server computer, receiving a request from the client computer, the request identifying an application and identifying a form of the application; and
 - [11b] in response to receiving the request: compiling the application into a compiled form;
 - [11c] transforming the compiled application into a transformed form of the compiled form of the application, wherein transforming comprises execution and compression of the compiled form; and
 - [11d] sending the transformed form of the application to the client computer.
- [12] The method of claim 11 wherein the identified form of the application is specified using a Universal Resource Locator.
- [13] The method of claim 11 wherein the identified form of the application is specified using a cookie.
- [14] The method of claim 11 wherein the identified form of the application is specified in a HTTP header.
- [15] The method of claim 11 wherein the identified form of the application is a web page.

- [16] The method of claim 11 Wherein the identified form of the application is a portion of a web page.
- [17] The method of claim 11 wherein transforming further comprises verification of the compiled form.
- [18] The method of claim 11 wherein transforming further comprises optimization of the compiled form.
- [19] The method of claim 11 wherein the transformed form of the application is sent to the client computer in an HTTP response.
- [20pre] The method of claim 11, further comprising:
 - [20a] storing a plurality of available forms of the application;
 - [20b] the server, in response to receiving the request, determining if the identified form of the application is one of the stored plurality of available forms of the application; and
 - [20c] if the identified form of the application is one of the stored plurality of available forms, sending the identified form of the application to the client computer without compiling and transforming.
- [21pre] The method of claim 11, further comprising:

- [21a] storing a plurality of available forms of the application;
- [21b] the server, in response to receiving the request, determining if the identified form of the application is one of the stored plurality of available forms of the application; and
- [21c] if the identified form of the application is one of the stored plurality of available forms, sending the identified form of the application to the client computer without compiling.

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 2nd day of May, 2024:

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

With an electronic courtesy copy to litigation counsel:

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

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

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

The undersigned hereby certifies that the foregoing Petition for *Inter Partes* Review contains **13,380** words, not including a table of contents, table of authorities, mandatory notices under §42.8, certificate of service, certificate of word count, appendix of exhibits or appendix of claim listing as specified by 37 C.F.R. §42.24, according to the word count feature of the word-processing software used to prepare the Petition.

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