

REQUEST FOR *EX PARTE* REEXAMINATION TRANSMITTAL FORM

Address to:
Mail Stop *Ex Parte* Reexam
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Attorney Docket No.: _____

Date: July 23, 2020

1. ☒ This is a request for *ex parte* reexamination pursuant to 37 CFR 1.510 of patent number US 7103640 B1 issued 09/05/2006. The request is made by:
☐ patent owner. ☒ third party requester.
2. ☒ The name and address of the person requesting reexamination is:
Aneeta Mishra
Aneeta X LLC, 50 Steeplechase, Irvine, CA 92602
3. Requester ☒ asserts small entity status (37 CFR 1.27) or ☐ certifies micro entity status (37 CFR 1.29). Only a patent owner requester can certify micro entity status. Form PTO/SB/15A or B must be attached to certify micro entity status.
4. This request is accompanied by payment of the reexamination fee as set forth in:
☐ 37 CFR 1.20(c)(2); or
☒ 37 CFR 1.20(c)(1). In checking this box for payment of the fee set forth in 37 CFR 1.20(c)(1), requester asserts that this request has forty (40) or fewer pages and complies with all other requirements of 37 CFR 1.20(c)(1).
Payment of the reexamination fee is made by the method set forth below.
a. ☐ A check in the amount of \$ _____ is enclosed to cover the reexamination fee;
b. ☐ The Director is hereby authorized to charge the reexamination fee
to Deposit Account No. _____;
c. ☐ Payment by credit card. Form PTO-2038 is attached; or
d. ☒ Payment made via EFS-Web.
☐ In addition, the Director is hereby authorized to charge any fee deficiencies to
Deposit Account No. _____.
5. ☐ Any refund should be made by ☐ check or ☐ credit to Deposit Account No. _____.
37 CFR 1.26(c). If payment is made by credit card, refund must be to credit card account.
6. ☒ A copy of the patent to be reexamined having a double column format on one side of a separate paper is enclosed. 37 CFR 1.510(b)(4).
7. ☐ CD-ROM or CD-R in duplicate, Computer Program (Appendix) or large table
☐ Landscape Table on CD

[Page 1 of 3]

This collection of information is required by 37 CFR 1.510. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) a request for reexamination. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to take 18 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Mail Stop *Ex Parte* Reexam, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

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

8. ☐ Nucleotide and/or Amino Acid Sequence Submission
If applicable, items a. – c. are required.
- a. ☐ Computer Readable Form (CRF)
- b. Specification Sequence Listing on:
- i. ☐ CD-ROM (2 copies) or CD-R (2 copies) **or**
- ii. ☐ paper
- c. ☐ Statements verifying identity of above copies.
9. ☐ A copy of any disclaimer, certificate of correction or reexamination certificate issued in the patent is included.
10. ☒ Reexamination of claim(s) 17-18 and 24 is requested.
11. ☒ A copy of every patent or printed publication relied upon is submitted herewith including a listing thereof on Form PTO/SB/08, PTO-1449, or equivalent.
12. ☐ An English language translation of all necessary and pertinent non-English language patents and/or printed publications is attached.
13. ☒ The attached detailed request includes at least the following items:
- a. A statement identifying each substantial new question of patentability based on prior patents and printed publications. 37 CFR 1.510(b)(1).
- b. An identification of every claim for which reexamination is requested, and a detailed explanation of the pertinency and manner of applying the cited art to every claim for which reexamination is requested. 37 CFR 1.510(b)(2).
14. ☐ A proposed amendment is included (only where the patent owner is the requester). 37 CFR 1.510(e).
15. ☒ It is certified that the statutory estoppel provisions of 35 U.S.C. 315(e)(1) or 35 U.S.C. 325(e)(1) do not prohibit requester from filing this *ex parte* reexamination request. 37 CFR 1.510(b)(6).
16. Service
- a. ☒ It is certified that a copy of this request (if filed by other than the patent owner) has been served in its entirety on the patent owner as provided in 37 CFR 1.33(c).
- The name and address of the party served are:
Brinks Gilson & Lione, P.O. Box 10395, Chicago, IL 60610

- Date of Service: July 23, 2020

- OR
- b. ☐ A duplicate copy is enclosed since service on patent owner was not possible. An explanation of the efforts made to serve patent owner **is attached**. See MPEP 2220.

17. Correspondence Address: Direct all communication about the reexamination to:		
☐ The address associated with Customer Number:		
OR		
☒ Firm or Individual Name <u>Aneeta Mishra</u> (at the address identified below)		
Address <u>50 Steeplechase</u>		
City <u>Irvine</u>	State <u>CA</u>	Zip <u>92602</u>
Country <u>USA</u>		
Telephone <u>(949) 343-9735</u>	Email <u>am@aneetax.com</u>	
18. ☒ The patent is currently the subject of the following concurrent proceeding(s): a. ☐ Copending reissue Application No. _____ b. ☐ Copending reexamination Control No. _____ c. ☐ Copending Interference No. _____ d. ☒ Copending litigation styled: <u>Kove IO, Inc. Amazon Web Services, Inc., No. 18-08175 (N.D. Ill.)</u> _____ _____		
WARNING : Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.		
<u>/Aneeta Mishra/</u> _____ Authorized Signature		<u>07/23/2020</u> _____ Date
<u>Aneeta Mishra</u> _____ Typed/Printed Name		_____ Registration No.
☐ For Patent Owner Requester ☒ For Third Party Requester		

TABLE OF CONTENTS

I.	INTRODUCTION	1
II.	LEVEL OF ORDINARY SKILL IN THE ART	3
III.	OVERVIEW OF THE 640 PATENT	3
A.	Written Description and Enablement	3
B.	Related Patents	3
C.	The Alleged Problem in Prior Art	4
D.	The Alleged Solution in the Patent	4
E.	Prosecution History	5
1.	1998-07-08: The 1998 Priority Application Filed	5
2.	1999-09-13: The 1999 Priority Application Filed	6
3.	2000-09-13: The 640 Application Filed	6
4.	2004-02-25: All Pending Claims Rejected	7
5.	2004-08-25: Claims Modified to Use “Redirect Message”	7
6.	2005-02-09: Yet Prior Art “Clearly Anticipated” Claims	7
7.	2005-08-09: 25 New Claims Added	8
8.	2005-11-14: The 25 New Claims Issued	8
F.	Litigation	8
IV.	OVERVIEW OF THE PRIOR ART	8
A.	Jaeger (1996)	8
B.	Netlib Publications	9
1.	Browne I, <i>Location-Independent Naming for Virtual Distributed Software Repositories</i> (1995)	9
2.	Browne II, <i>The National HPCC Software Exchange</i> (1995)	9
3.	Browne III, <i>Management of the NHSE—a Virtual Distributed Digital Library</i> (1995)	10
4.	Moore, <i>Resource Cataloging and Distribution System</i> (1997)	10
5.	Fagg, <i>Scalable Networked Information Processing Environment (SNIPE)</i> (1997)	11
C.	RFC 1737, <i>Functional Requirements for Uniform Resource Names</i> (1994)	11
D.	Domain Name Service	12
1.	RFC 1034, <i>Domain Names—Concepts and Facilities</i> (1987)	12
2.	RFC 1035, <i>Domain Names—Implementation and Specification</i>	

	(1987).....	12
E.	Other Publications	13
1.	Knuth, <i>The Art of Computer Programming, Volume 3: Sorting and Searching</i> (1973)	13
2.	Cormen, <i>Introduction to Algorithms</i> (1990).....	13
3.	Binstock, <i>Hashing Rehashed</i> (1996)	13
4.	Tanenbaum, <i>Computer Networks</i> (1996).....	14
5.	Kernighan, <i>The C Programming Language</i> (1988).....	14
V.	SUBSTANTIAL NEW QUESTIONS OF PATENTABILITY	14
A.	Claim Construction.....	14
B.	Statement Identifying the Substantial New Questions of Patentability.....	16
1.	SNQ 1: Jaeger.....	16
2.	SNQ 2: Netlib-Related Publications.....	18
3.	SNQ 3: Jaeger and Netlib-Related Publications.....	18
C.	Detailed Explanation of the Pertinency and Manner of Applying the Cited Prior Art to Each Challenged Claim.....	19
1.	Claim 17.....	19
2.	Claim 18.....	28
3.	Claim 24.....	29
VI.	CONCLUSION	30

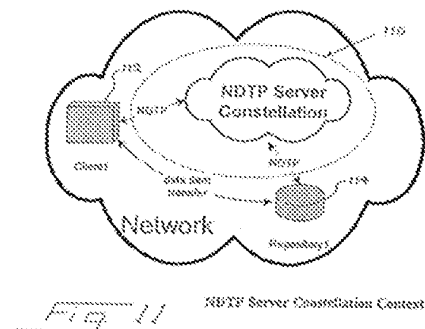
I. INTRODUCTION

The 640 patent discloses storage and retrieval of information—that is, table look-up. Donald E. Knuth, *The Art of Computer Programming, Volume 3: Sorting and Searching* 389, Addison-Wesley (1973) (“This chapter [titled ‘SEARCHING’] might have been given the more pretentious title, ‘Storage and Retrieval of Information’; on the other hand, it might simply have been called ‘Table Look-Up.’”) (AX1465). The patent discloses that each network resource, stored at a network location, is assigned an identifier. That identifier-location association is then published to lookup servers. A network user trying to locate that resource uses that identifier. The user queries its lookup server using the identifier. The server returns a location associated with the identifier or, if it does not store that association, just a table of other location servers. AX0511 (“Hash tables are just like most tables . . . and the old-fashioned word for ‘arrays.’”); AX0530 (“A hash table is a generalization of the simpler notion of an ordinary array.”). **Claims 17-18 and 24**, to which this reexamination request is directed, teach that lookup aspect.

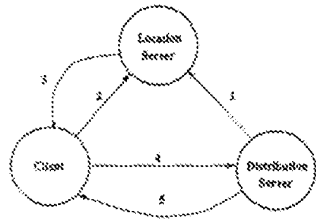
Jaeger however previously disclosed an architecture for distributed networks that fairly suggested the same. So did the publications related to the **Netlib** project, a repository of software, documents, and databases for the scientific community maintained by AT&T Bell Laboratories, the University of Tennessee, and Oak Ridge National Laboratory. The purported solutions recited in claims 17-18 and 24 would have been obvious under Jaeger or Netlib-related references or their combinations, the knowledge of the relevant artisan, and indeed the artisan’s common sense.

The patent claims that in prior art systems, one had to know which data repository to query for a particular information. Instead of polling data repositories, which was inefficient and impractical, the claimed invention separated the “where” from the “what” using “location” servers,

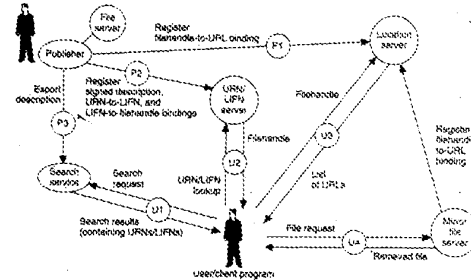
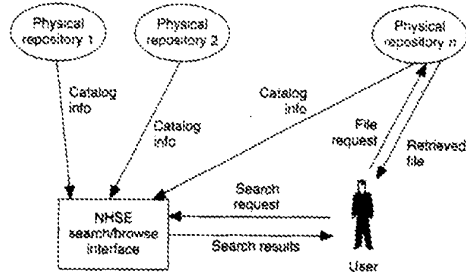
wherein the location servers acted as intermediaries between clients requesting data and the data repositories. A client would first lookup the location associated with an identifier by contacting its location server. The client would receive the location and then request data from the data repository at that location. Alternatively, the client would receive a table of other location servers, into which the client would look up to identify the correct location server that stored the identifier-location mapping, contact that server to get the location, and then request data from the correct data repository.



Jaeger previously disclosed confronting that issue: “[a] . . . problem with file retrieval from the Internet is that it is difficult to locate a specific file without knowledge of its specific location or significant user involvement.” AX0321-0322. One did not have to know which data repository to query for particular information because Jaeger too disclosed using location servers that served as intermediaries between clients requesting data and the corresponding data servers.



The publications related to the Netlib project also contemplated LIFN “location server” intermediaries, where such servers intermediated between the clients/users and the data repositories—separating the “where” from the “what.”



The “redirect message”-related claim limitations is simply a post-solution predictable variation of that which Jaeger and Netlib-related publications disclosed. Tellingly, patentee’s first attempt at claiming such limitations, where the client calculated the location associated with the identifier based on the redirect message without making an additional trip to another location server, was rejected under multiple anticipatory references. Patentee then simply added another trip to a second location server. That approach cautions against patentability: *first*, where the identifier-location associations are maintained in hash tables, that second design is an obvious variation of the first design that was rejected as unpatentable; *second*, the relevant artisan would have discerned the two choices as among the limited set of options obvious to try to solve the problem of locating resources on a distributed network. Given the design need to solve the well-recognized problem of locating resources on the Internet in 1990s and the availability of “a finite number of identified, predictable solutions,” the relevant artisan would have had “good reason to pursue both the known options within his or her technical grasp.” *KSR Int’l Co. v. Teleflex Inc.*, 550 U.S. 398, 421 (2007). If that “led to the anticipated success, it is likely the product not of innovation but of ordinary skill and common sense.” *Id.*

II. LEVEL OF ORDINARY SKILL IN THE ART

The relevant artisan would have had a bachelor's degree in computer science (or in a related field) and two years of experience working with distributed software or networking systems, or at least seven years of experience working with such systems (or in a related field), or equivalent. Additional education would have made up for less work experience and vice versa. That artisan would also have had knowledge of prior art discussed in § IV. *Okajima v. Bourdeau*, 261 F.3d 1350, 1355 (Fed. Cir. 2001) (prior art itself may reflect an appropriate level of skill); *Custom Accessories, Inc. v. Jeffrey-Allan Indus., Inc.*, 807 F.2d 955, 962 (Fed. Cir. 1986) (“The person of ordinary skill is a hypothetical person who is presumed to be aware of all the pertinent prior art.”). At a minimum, the relevant artisan would have readily discerned that associative key-value mappings would have been stored as hash tables, distributed across location servers, and looked up using hash functions for simple, near-random access in $O(1)$ search time. AX1495 (teaching table lookup as “avoid[s] all this rummaging around by doing some arithmetical calculation on K , computing a function $f(K)$ which is the location of K and the associated data in the table”); AX0511 (table lookups provide “simple, near-random access”); AX0530 (“[T]he expected time to search for an element in a hash table is $O(1)$ Instead of using the key as an array index directly, the array index is *computed* from the key.”).

III. OVERVIEW OF THE 640 PATENT

US 7,103,640 B1 (the 640 patent) is currently assigned to Kove IO, Inc.

A. Written Description and Enablement

The patent traces its priority to application no. 09/111,896 (the 1998 priority application) under 35 U.S.C. § 120 and to application no. 60/153,709 (the 1999 priority application) under § 119(e). AX0015 at 1:6-12. Those code sections require the priority applications to disclose the inventions, specifically providing written description and enablement support for the claims that issued. *Research Corp. Techs., Inc. v. Microsoft Corp.*, 627 F.3d 859, 870 (Fed. Cir. 2010) (“Entitlement to a filing date extends only to subject matter that is disclosed.”).

B. Related Patents

The 640 patent has two other related patents. The 978 patent issued in 2007 from a continuation-in-part application of the 640 application. 978 patent at 1:9-26. Amazon et al. petitioned for an IPR of the 978 patent, but the PTAB found that the petitioners “failed to show a reasonable likelihood of success” in establishing “anticipation” in view of RFC 1034, “an

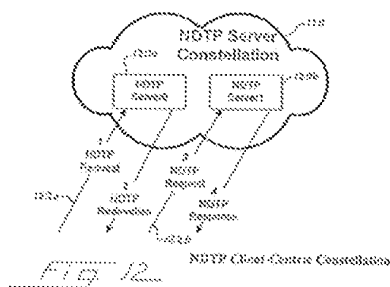
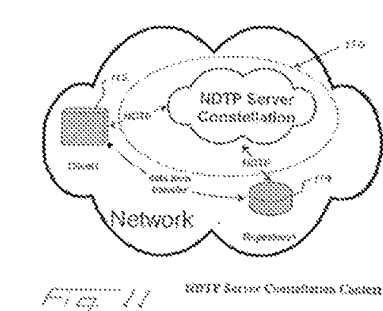
introduction to concepts and facilities, but not implementation and specification,” of Domain Name Service (DNS). *Amazon Web Services, Inc. v. Kove IO, Inc.*, Case IPR2020-00276, *15, *19-22 (P.T.A.B. June 16, 2020) (Paper 23) (denying petition requesting IPR of the 978 patent claims) (“*IPR Decision*”). The 170 patent issued in 2010 from a continuation application of the 640 application. 170 patent at 1:6-15. The written descriptions of the three patents are materially similar for present purposes: The definitions of “identifier,” “location,” “client,” and “server” appear to be the same, given a comparison of the pertinent disclosures. Figures 11-12 of the 640 patent, AX0013, too carried over to Figures 18-19 of the 978 patent.

C. The Alleged Problem in Prior Art

Prior art retrieval systems based on client-server architecture were inflexible: “Typically, only data that are statically associated with that server are returned. Disadvantageously, the search is also usually restricted to previously known systems. The search is thus conducted only where the server knows in advance to look. Another disadvantage of known retrieval systems is the difficulty in accessing data in different forms.” AX0015 at 1:37-43. Retrieving plain-text from repositories was purportedly different from finding software libraries (and their dependencies).

D. The Alleged Solution in the Patent

The patent teaches a mechanism to solve that problem, looking up information across



information sources—indeed, even reaching data repositories across distributed networks like the Internet. AX0015 at 2:12-14, 1:26-30. The protocol involves (1) storing a key-value mapping on location servers, where the key is referenced as “identifiers” and the associated value as “location,” AX0016 at 3:10-11, 3:18-20; and (2) a client searching the value using the key as a search criterion, AX0016 at 3:11-13. Claims 17-18 and 24 are merely focused on the latter. AX0025-0026.

Creating the key-value mapping: The key-value mapping is stored on location servers. AX0017 at 6:47. When new information is freshly captured by any device and stored on a data repository, the location server connected to that repository first creates a key-value mapping. AX0016 at 3:18-20. This key-value mapping—not the actual data—is then propagated to other location servers.

Claims 17-18 and 24 assume key-value mappings having been created and propagated on the servers, and indeed rely on the pre-existence of that association.

Looking up the key-value association: A client uses the key as its lookup criterion when it connects to its location server. If that location server happens to be the one that previously stored a key-value mapping, it returns to the client the value corresponding to the key (the location of the data repository). Otherwise, the server returns a table of other location servers. AX0012-0013; AX0023 at 17:8-20, 17:61-18:8; AX0023 at 18:9-13 (“Behind the scenes, the server constellation 110 could also be using a hierarchical organization or a cluster organization for managing indices. The important point of this topology is *pushing processing emphasis toward clients (not shown) rather than toward servers 120a,b.*” (emphasis added)).

Under the client-centric approach, “[i]f the contacted server 120a does not have all the information, as for example in a redirect, then it passes the request to another server 120b. If the second server 120b is appropriate it responds appropriately, or it passes the request on to another server (not shown), and so on.” AX0023 at 17-67:18-5 (emphasis added); see also AX0023 at 17:49-57 (“The NDTP_RDR_RSP message . . . information may be an array of NDTP server hosts from which one is selected by applying a well-known function to the identifier string supplied in the NDTP request 122b[.]” (emphases added)). The client uses its key as the lookup criterion into that table/array to find a new location server that maintains the relevant key-value association. See, e.g., AX0025 at claim 17. The client then connects to the new location server, which returns to the client the value corresponding to the key—the location of the data repository to which the client must then connect to retrieve data. *Id.* The client thus remains firmly in control.

E. Prosecution History

1. 1998-07-08: The 1998 Priority Application Filed

The 1998 priority application noted that the invention is “similar in function to the domain name service which supports all Internet addresses.” AX0034. It characterized DDNS, the “software implementation” of the invention, as “generalized” compared to the “similar” domain name server implementation. AX0034; see also AX0036 (“The DDNS implementation is similar to DNS (Domain Name Service), which supports all Internet name lookup service. . . . However DDNS is generalized and optimized for resolving database locations and database service locations.”). The application showed a location server recursively asking another location server whether an identifier mapped to any location. AX0064-0065 ¶ 21; AX0065 ¶ 22 (“DDNS-2:ISP,”

which does not find a location corresponding to client's identifier, then asks "DDNS-1 Servers" where to find that location).

2. 1999-09-13: The 1999 Priority Application Filed

The 1999 priority application described an iterative lookup alternative to the recursive one discussed in the 1998 priority application. It specified a first-level location server sending back to client forwarding addresses to other location servers, and the client then iteratively following through on each such referrals to successively contact other location servers till it finds an association. AX0115 (Figure 4); AX0110 (explaining the iterative lookup: "Data retrieval can be [as follows:] Machine C (54) wishes to retrieve data keyed to identifier 123.abc. Machine C (54) queries its upstream server, machine A (50). Machine A (50) returns to machine C (54) forwarding instructions about how to reach machine B (52). Machine C (54) queries machine B (52) directly, which returns forwarding instructions to machines D (56) and E (58). Machine C (54) may query machines D (56) or E (58) in series or in parallel to retrieve the actual data.").

3. 2000-09-13: The 640 Application Filed

Application no. 09/661,222 (the 640 application) carried over the recursive lookup of the 1998 priority application. *Compare* AX0077-0078 (Figs. 2 and 3 of the 1998 priority application) *and* AX0064-0065 ¶ 21 *and* AX0065 ¶ 22, *with* AX0012 *and* AX0014 (Figs. 10 and 13 in the 640 application) *and* AX0023-0024 (describing the server-centric recursive lookup approach). It also carried over the iterative lookup from the 1999 priority application. *Compare* AX0115 (Fig. 4 in the 1999 priority application) *and* AX0110 (describing Fig. 4), *with* AX0013 (Fig. 12 in the 640 application) *and* AX0023 (describing the client-centric iterative approach). The application however appeared to soften the comparison to DNS:

One way to understand [NDTP] is as a highly dynamic DNS for data. DNS maintains a mapping between names and machines. NDTP and its associated servers maintain a mapping between entity identifiers and data locations. The identifier/location mapping maintained by NDTP servers is much more dynamic (more frequent updates), than the domain name/IP address mapping maintained by DNS.

AX0158. The application acknowledged the invented "networking protocol" referenced as NDTP "maintain[ed] a mapping between entity identifiers and data locations" just like "DNS maintains a mapping between [domain] names and machines [IP address]." AX0158. The only distinguishing feature was purportedly that "[t]he identifier/location mapping maintained by NDTP servers is

much more dynamic (more frequent updates), than the domain name/IP address mapping maintained by DNS.” AX0158. But no specific enablement beyond the recursive and iterative lookups (like those in DNS) was provided to support that assertion.

4. 2004-02-25: All Pending Claims Rejected

None of the claims survived prosecution. AX0165. Examiner found that prior art anticipated the claims. AX0168 (rejecting claim 9) (“Edelstein teaches wherein the first server entity transmits the client request to the second server entity if the first server entity cannot provide the at least one location string to the client entity”); AX0170 (rejecting claim 13) (same). With respect to claim 8, a claim that recited that key-value mappings at the location servers is “performed using a hash function,” the examiner took “Official Notice . . . that the use of a [] hashing operation in a computer networking environment was well known in the art,” AX0172.

5. 2004-08-25: Claims Modified to Use “Redirect Message”

Rejected claims were then modified to recite the iterative approach packaged as “a redirect message,” AX0178 (claim 12) and AX0180 (e.g., claim 21). *See also* AX0177 (claim 8 reciting “wherein the shared function comprises a hash function and the redirect message comprises a data table configured for operation with the hash function”). Notably, no distinction was made between the *server-centric* recursive lookup and *client-centric* iterative lookup, both cited as support for the “redirect message”-related claims. AX0184 (“Support for the amendments to claims 13-16 and added claims 21-23 may be found throughout the specification as at, for example, page 24, line 13 – page 25, line 9; page 29, lines 20-31 and FIGS.11-13.”).

Claim 21 recited “redirect message comprises . . . data configured in a table” where the client “determining the location of at least one location string by applying a shared function to the table.” AX0180. That language unsurprisingly made clear that the client need not connect to another location server and—merely based on the “redirect message”—could calculate the location string associated with the identifier. That result flowed from the properties of hash tables and functions. *E.g.*, AX1495-1496; *see also* AX0217 (patentee conceding that in the recited “‘well-known function’ redirection[,] both servers and clients compute the same function on the same data to generate the same location results.” (emphasis added)).

6. 2005-02-09: Yet Prior Art “Clearly Anticipated” Claims

A new examiner, AX0188, still rejected the claims “as being clearly anticipated,” AX0193 ¶ 26; AX0215 (listing four § 102 references against all pending claims). The examiner also noted

that “each and every one of the[] independent claims related to an identical inventive concept,” discerning that a redirection step added nothing more. AX0196-AX0197 ¶ 39; AX0197 ¶ 40.

7. 2005-08-09: 25 New Claims Added

All claims were then cancelled and 25 new claims added, focused on an additional trip from the client to a second location server. AX0210-0214. Rationalizing that choice, it was noted that “the claimed redirection feature” is intended to use “a theoretically one-point-failure-retry redirection method that configurably distributes computation among clients and servers.” AX0220.

8. 2005-11-14: The 25 New Claims Issued

Renumbered as claims 1-25, pending new claims 24-37 and 39-49 then issued.

F. Litigation

Claims 17-18 and 24 have been asserted in *Kove IO, Inc. v. Amazon Web Services, Inc.*, No. 18-08175 (N.D. Ill.) (filed Dec. 12, 2018); *id.*, Docket No. 129, Counterclaims ¶ 4 (May 1, 2020). There, the court recently issued a scheduling order. *Id.*, Docket No. 167 (July 6, 2020).

IV. OVERVIEW OF THE PRIOR ART

A. Jaeger (1996)

Jaeger defines the following problem: “Distribution of files in an untrusted network is complicated by . . . location of files” and relatedly that “it is difficult to locate a specific file without

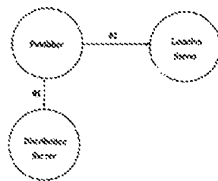


Figure 2. The file publication protocol: (1) Publisher uploads a file identifier-to-pathname mapping to distribution server; (2) Publisher uploads a file identifier-to-distribution server mapping to location server

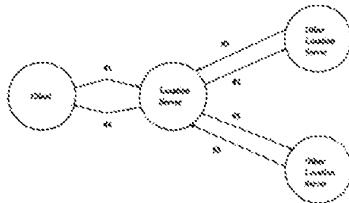


Figure 3. The file location protocol: (1) Client sends a file identifier to the client's location server to find the distribution servers that possess a file; (2) Client's location server requests that other location servers find the distribution servers that possess a file (optional); (3) Other location servers return to the client's location server a set of distribution servers that possess a file; (4) Client's location server returns a set of distribution servers that possess the file.

knowledge of its specific location or significant user involvement.” AX0321-0322. Jaeger teaches a file transfer protocol to “provide[] the capability to automatically locate, retrieve, and verify files specified by a client.” AX0321; AX0322.

Jaeger teaches that when a “client wants to retrieve a file, it specifies a file identifier.” AX0322. It explains that “[c]lients can locate a file on a particular distribution server because it is ‘published’ to the network.” AX0326. Its “implementation has a publisher process that uploads file location information to both location servers and distribution servers.” AX0326. Location servers thus “store a map of file identifiers to the set of servers that provide them.” AX0326. “The results of the file publication protocol are: (1) the location server can list the set of distribution servers that claim to possess a file that matches a file identifier

and (2) the distribution server can locate that file and the file's certificate given a file identifier." AX0327. The client's location server can provide the location for a file identifier or check with other location servers whether they store the identifier-location mapping and respond to client with a location structure message containing "[a]n [a]rray of [d]istribution [s]ervers." AX0327-0328.

B. Netlib Publications

Netlib repository, a collection of high-quality publicly available mathematical software, has been in operation at least since September 1985. AX0507.

1. Browne I, *Location-Independent Naming for Virtual Distributed Software Repositories* (1995)

Browne I describes retrieval of software through a virtual distributed repository. Instead of a user or client program directly accessing a software repository, it does so through an "NHSE"

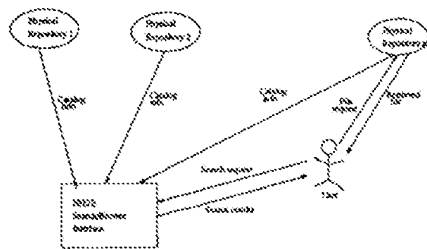


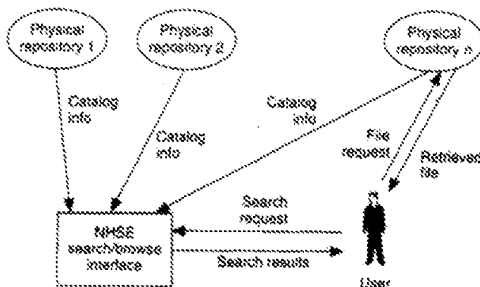
Figure 1: Virtual Repository Architecture

intermediary as part of "[a] location-independent naming system for network resources." AX0332. That was necessitated because of the proliferation of "a large number of less familiar [software] repositories," which required "interoperable trust mechanisms and to reduce the number of parties with whom the user must establish

trust." AX0333. "[I]mplementing a location-independent naming system," including "mechanisms for authenticity and integrity checking," ensured trust. AX0333. The implementation involved "a naming system that provides for two levels of naming. The binding between a lower-level name (called a LIFN) and file contents is unchangeable and verifiable." AX0333. "A higher-level name (called a URN) is associated with a cataloging record that includes the lower-level name as well as other descriptive information." AX0333.

2. Browne II, *The National HPCC Software Exchange* (1995)

The paper discusses the NHSE, referenced in Browne I. "The NHSE is a NASA-funded



project being developed by members of the Center for Research on Parallel Computation." AX0341. The paper further explains that "[t]he NHSE is an Internet-accessible resource that promotes the exchange of software and information among those involved with high-performance computing and communications."

Rice, CalTech, and Argonne).” AX0370.

RCDS discloses use of URNs, LIFNs, and URLs to identify resources. AX0360-0361; AX0357. “Among the services provided by RCDS are: URN-to-catalog record mapping and LIFN-to-location mapping.” AX0362. The paper teaches the following components: (1) clients: consume the network resources; (2) file servers: provide access to the files themselves and can be ordinary general-purpose file servers or some other file access protocol or servers; (3) resource catalog servers: maintain information about the characteristics of network-accessible resources and accept queries about the characters of such resources from clients; (4) location servers: maintain information about the locations of network resources and accept queries for location data from clients. AX0360; AX0365 (Fig. 2).

5. Fagg, *Scalable Networked Information Processing Environment (SNIPE)* (1997)

“[RCDS] is designed to facilitate very scalable and fault-tolerant access to network-accessible resources and metadata, and to provide end-to-end authenticity and integrity guarantees for those resources and metadata.” AX0374. “RCDS accomplishes this by replicating the resources and metadata at a potentially large number of (perhaps geographically dispersed) locations. The set of locations for a resource are maintained in a highly distributed and replicated location registry. Similarly, the metadata for a resource (a list of attribute ‘name=value’ pairs called assertions) are maintained in a separate distributed and replicated registry, which is indexed by the resource’s Uniform Resource Locator (URL) or Uniform Resource Name (URN).” AX0374.

C. RFC 1737, *Functional Requirements for Uniform Resource Names* (1994)

It “specifies a minimum set of requirements for a kind of Internet resource identifier known as Uniform Resource Names (URNs).” AX0386. It explains that to “build applications in the most general case, the user must be able to discover and identify the information, objects, or . . . resources” and “[a]s the network and interconnectivity grow, the ability to make use of remote, perhaps independently managed, resources will become more and more important.” AX0386; AX0387. “A URN identifies a resource or unit of information. It may identify, for example, intellectual content, a particular presentation of intellectual content, or whatever a name assignment authority determines is a distinctly namable entity.” AX0387.

D. Domain Name Service

1. RFC 1034, *Domain Names—Concepts and Facilities* (1987)

RFC 1034 introduces concepts related to the Domain Name System (DNS). AX0393. It describes distributed database problems, where “a particular name server may be presented with a query that can only be answered by some other server.” AX0396. It prefers an approach it labels “‘iterative,’ in which the server refers the client to another server and lets the client pursue the query,” but is open to the approach it labels “‘recursive,’ in which the first server pursues the query for the client at another server.” AX0396. The “response by the name server either answers the question posed in the query, refers the requester to another set of name servers, or signals some error condition.” AX0408.

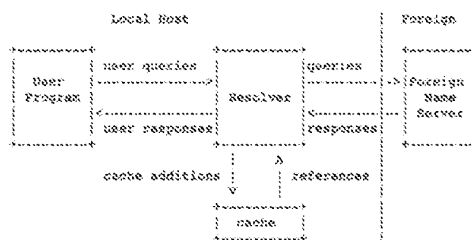
2. RFC 1035, *Domain Names—Implementation and Specification* (1987)

RFC 1035 “details . . . the domain system and protocol, and assumes that the reader is familiar with the concepts discussed in [RFC-1034].” AX0448.

“From the user’s point of view, domain names are useful as arguments to a local agent, called a resolver, which retrieves information associated with the domain name. . . . To enable the user to request a particular type of information, an appropriate query type is passed to the resolver with the domain name.” AX0450.

“From the resolver’s point of view, the database that makes up the domain space is distributed among various name servers. Different parts of the domain space are stored in different name servers The resolver starts with knowledge of at least one name server. When the resolver processes a user query it asks a known name server for the information; in return, the resolver either receives the desired information or a referral to another name server. Using these referrals, resolvers learn the identities and contents of other name servers.” AX0450.

“The simplest, and perhaps most typical, configuration is as shown here. AX0451. “Note that the resolver may have to make several queries to several different foreign name servers to answer a particular user query, and hence the resolution of a user query may involve several network accesses and an arbitrary amount of time.” AX0451. Finally, “[i]f RD [‘Recursion Desired’ flag] is set, it directs the name server to pursue the query recursively. Recursive query support is optional.” AX0474.



E. Other Publications

1. Knuth, *The Art of Computer Programming, Volume 3: Sorting and Searching* (1973)

This well-known volume on sorting-and-searching teaches that its chapter on “Searching” relevant here “might have been given the more pretentious title, ‘Storage and Retrieval of Information’; on the other hand, it might simply have been called ‘Table Look-Up.’” AX1465. The volume explains that a “possibility [for table lookup] is to avoid all this rummaging around by doing some arithmetical calculation on K , computing a function $f(K)$ which is the location of K and the associated data in the table.” AX1495. “The verb ‘to hash’ means to chop something up or to make a mess out of it; the idea in hashing is to chop off some aspects of the key and to use this partial information as the basis for searching. We compute a *hash function* $h(K)$ and use this value as the address where the search begins.” AX1496. The volume further explains that “[h]ash functions” were “known.” AX1497.

2. Cormen, *Introduction to Algorithms* (1990)

“A hash table is a generalization of the simpler notion of an ordinary array.” AX0530. “Under reasonable assumptions, the expected time to search for an element in a hash table is $O(1)$.” AX0530. “Directly addressing into an ordinary array makes effective use of our ability to examine an arbitrary position in an array in $O(1)$ time.” AX0530. “[H]ash tables [are] an effective alternative to directly addressing an array, since a hash table typically uses an array of size proportional to the number of keys actually stored. Instead of using the key as an array index directly, the array index is *computed* from the key.” AX0530.

3. Binstock, *Hashing Rehashed* (1996)

Hashing algorithms “occupy a unique place in the hearts of programmers,” “[d]iscovered early on in computer science,” and are “among the most approachable algorithms.” AX0511. “Hashing” is defined as “[a] method for converting representations of value within fields, usually keys, to a more compact form” and alternatively as “[a]n addressing technique that uses keys to store and retrieve data in a file.” AX0517.

“Hash tables are just like most tables” and “the old-fashioned word for ‘arrays.’” AX0511. “A table becomes a hash table when nonsequential access to a given element is achieved through the use of a hash function.” AX0511. “A hash function is generally handed a data element, which it converts to an integer that is used as an index into the table.” AX0511.

“In most programs, the hash function occupies an insignificant amount of the execution time.” AX0511. The article presents “two excellent, general-purpose hash functions. HashPJW() (see Example 1) is based on work by Peter J. Weinberger of AT&T Bell Labs, and is better known. ElfHash() (Example 2) is a variant of HashPJW().” AX0511; AX0512 (detailing the examples).

4. Tanenbaum, *Computer Networks* (1996)

Under the heading “Locating Information on the Web,” Tanenbaum discloses that “one of the data structures needed is a hash table,” where the “main use of the hash table is to start with a URL,”—that is, an identifier, and “be able to quickly determine whether it is already present in *url_table*,”—that is, a hash table. AX1349. It further explains that “[t]he heart of the search engine is a recursive procedure *process_url*, which takes a URL string as input. It operates as follows. First, it hashes the URL to see if it is already present in *url_table*. If so, it is done and returns immediately. Each URL is processed only once.” AX1349-1350.

5. Kernighan, *The C Programming Language* (1988)

This foundational volume on the “C” programming language teaches in a section directed to “a table lookup” that “a table-lookup package . . . code is typical of what might be found in the symbol table management routines of a macro processor or a compiler.” AX1686. “There are two routines that manipulate the names and replacement texts. *install(s, t)* records the name *s* and the replacement text *t* in a table; *s* and *t* are just character strings. *lookup(s)* searches for *s* in the table, and returns a pointer to the place where it was found, or NULL if it wasn’t there.” AX1686. “The algorithm is a hash search—the incoming name is converted into a small non-negative integer, which is then used to index into an array of pointers. An array element points to the beginning of a linked list of blocks describing names that have that hash value.” AX1686-1687. It explains that “[t]he *for* loop in lookup is the standard idiom for walking along a linked list.” AX1688.

V. SUBSTANTIAL NEW QUESTIONS OF PATENTABILITY

A. Claim Construction

Claims are given “their broadest reasonable interpretation consistent with the specification (see *In re Yamamoto*, 740 F.2d 1569 (Fed. Cir. 1984)).” MPEP § 2240. Here, the written description defines certain terms. “An ‘identifier string’ or an ‘identifier’” is defined as “a unique string with which zero or more location strings are associated.” AX0016 at 4:7-9; AX0016 at 4:37-42 (“The NDTP server treats the entity key as an unstructured stream of octets, which is assumed to be unique to a particular entity. The precise structure of the NDTP entity key and the mechanism

for ensuring its uniqueness are a function of the application in which the NDTP server is used.”). “A ‘data location’ or a ‘location’” is defined as “a string that is a member of a set of strings associated with an identifier string in [a] . . . server.” AX0016 at 4:9-11; AX0016 at 4:45-48 (“NDTP also defines a data location specifier as a string used to specify a data repository [*sic*] in which data associated with a particular entity may be found.”). “An ‘NDTP client’ or a ‘client’” is defined as “a network-attached component that initiates update or lookup of identifier/location mappings from an NDTP server with NDTP request messages.” AX0016 at 4:11-14. “An ‘NDTP server’ or a ‘server’” is defined as “a network-attached component that maintains a set of identifier/location mappings that are modified or returned in response to NDTP request messages from NDTP clients.” AX0016 at 4:14-18.

That notwithstanding, the broadest reasonable interpretation may yield a broader understanding of the terms, especially as the terms may be used in a claims. For example, with respect to “identifier,” though the written description indicates a one-to-one identifier-to-entity mapping, the PTAB disagreed with the patentee (under *Phillips v. AWH Corp.*, 415 F.3d 1303 (Fed. Cir. 2005) (en banc) and progeny) that “a 1-to-1 relationship between an identifier and the entity that it is identifying” is necessarily warranted. *IPR Decision* at *12-13. Though that view pertained to the 978 patent, identifier-related written description overlaps between the 640 and the 978 patents. *See also NTP, Inc. v. Research In Motion, Ltd.*, 418 F.3d 1282, 1293 (Fed. Cir. 2005) (when construing claim terms in patents that derive from the same parent application and share common terms, “we must interpret the claims consistently across all asserted patents”). The PTAB stated that “a given identifier must resolve to a single entity, i.e., at most, a many-to-1 relationship” may be apt in a given context. *IPR Decision* at *13. The relevant artisan would have discerned for example that all files under a directory may uniquely resolve to a single directory name, and hence the directory name could have been used as an identifier; all data in a file may resolve to a single encoding, and hence that encoding could have been used as an identifier. For example, all data for a specific patient may resolve to a single SSN or patient ID belonging to that patient, and hence the SSN or patient ID could have been used as an identifier; all data (files or folder or “a stream of bits”) for a car may resolve to the car’s VIN, and hence the VIN could have been understood as the identifier. Accordingly, an identifier could have been broadly understood to reference whatever stream of bits that has been identified to be published across the distributed network using that identifier, under the “broadest reasonable interpretation” standard.

B. Statement Identifying the Substantial New Questions of Patentability

A prior art patent or printed publication raises a substantial question of patentability where there is a substantial likelihood that a reasonable examiner would consider it important in deciding the patentability of a claim. MPEP § 2242. Here, the prosecution history does not indicate that any substantive consideration was given to the architectures disclosed by Jaeger or Netlib-related publications or their combinations in evaluating the patentability of claims 17-18 and 24.

Even if Office considered these references or their combinations during the prosecution of the 640 patent, this request presents the references in a new light not previously considered by the Office because it focuses on the portions of these references that were neither cited by the applicant nor fully discussed on the record by the examiner. A reasonable examiner would consider Jaeger, Netlib-related publications, or their combinations (in light of the knowledge and common sense of the relevant artisan) important in deciding whether claims 17-18 and 24 are patentable because these references or their combinations would have taught or fairly suggested the claimed inventions. The references thus raise substantial new questions of patentability.

To be sure, each of the below substantial new questions of patentability (SNQs) is raised in light of the background knowledge and common sense of the relevant artisan. *In re Baird*, 16 F.3d 380, 383 (Fed. Cir. 1994) (“[A] reference must be considered not only for what it expressly teaches, but also for what it fairly suggests.” (alteration in original)); *In re Aslanian*, 590 F.2d 911, 914 (CCPA 1979) (explaining that in determining obviousness, all references are assessed “on the basis of what they reasonably disclose and suggest to one skilled in the art” (quoting *In re Baum*, 374 F.2d 1004, 1009 (CCPA 1967))).

1. SNQ 1: Jaeger

Claims 17-18 and 24 would have been obvious under Jaeger. Jaeger teaches that key-value associations are maintained in a distributed fashion across multiple location servers. The relevant artisan would have readily discerned that Jaeger function appears similar to a naming server and so key-value mappings would have been stored as hash tables spread across Jaeger’s location servers—in other words, that each location server maintains a partial set of key-value mappings in its hash table. RFC1034 describes the use of “name servers,” which each have “complete information about *a subset* of the domain space, and pointers to other name servers that can be used to lead to information from any part of the domain tree.” AX0398 (emphasis added); AX0450 (“From the resolver’s point of view, the database that makes up the domain space is distributed

among various name servers. Different parts of the domain space are stored in different name servers, although a particular data item will be stored redundantly in two or more name servers.”); AX0193 (examiner observing that “naming servers inherently implemented tables, with mapping hashing functions to lookup location/address/etc. information, as well as to effect direction of naming requests to other naming servers.”).

Jaeger teaches that when the client’s location server does not store a key-value association corresponding to the identifier, it contacts other location servers and returns an array of distribution servers (which the client may then contact to retrieve the network resource). The relevant artisan would have discerned that this approach would have been better modified to return simply a table of those other location servers so the client could cache the results and then take an iterative approach, on which a hash function could have been used for near-random access in $O(1)$ search time. AX1495; AX0511; AX0530. For example, the relevant artisan would have recognized that name servers required an iterative approach. AX0396; AX0474. A foundational book on programming taught a local iterative “*for* loop in lookup [as] the standard idiom for walking along a linked list.” AX1688. Long before that, a well-known volume on sorting-and-searching explained that a “possibility [for table lookup] is to avoid all this rummaging around by doing some arithmetical calculation on K , computing a function $f(K)$ which is the location of K and the associated data in the table.” AX1495. More broadly, the relevant artisan would have recognized that “[h]ashing is useful anywhere a wide range of possible values needs to be stored in a small amount of memory and be retrieved with simple, near-random access.” AX0511; *see also* AX0511 (“As such, hash tables occur frequently in database engines” where “the table is a principal data structure” and “all accesses to the table must be done through the hash function.”).

Indeed, Jaeger marks its server-centric location-resolution as “[o]ptional.” AX0327. Given that Jaeger is focused on “trust” in looking up network resources, the relevant artisan would also have been motivated to take a more client-centric approach over the expressly taught server-centric one. That is, instead of the client’s location server passing around the identifier key to other location servers across the network, it may simply return a table of other location servers for the client to use. Given the hashed nature of the table, the relevant artisan would have readily discerned that applying a hash function to the identifier would provide an index into that table to ascertain the location server that would contain that location associated with the identifier, arriving at the claimed limitation of claims 17-18 and 24. AX1495; AX0511; AX0530.

2. SNQ 2: Netlib-Related Publications

Claims 17-18 and 24 would also have been obvious under Netlib-related publications, which (like Jaeger) teach decoupling network resources from their locations. The decoupling is achieved by interposing intermediary LIFN or catalog servers that maintain key-value associations, where LIFNs are identifiers for such resources at a lower level and URNs at a higher level. Indeed, these publications reference URNs, defined in RFC 1737, which identifies a network resource or unit of information for “a distinctly namable entity.” AX0387. RFC 1737 also taught “resolvers or location services of some sort assist or provide URN to URL mapping” and that “one or many such services for the resources named by a particular naming authority.” AX0391.

More specifically, Browne I’s figures 1-3 and related description, Browne II’s figures 2-3 and related description, Browne III’s figure 1 and its description, Moore’s figure 2-3 and related description—all of that collectively discloses or fairly suggests decoupling of networking resources from their locations in the same manner as taught in claims 17-18 and 24. The relevant artisan would have readily recognized that the LIFN location server may pass back the resolved location corresponding to a LIFN identifier or alternatively that the client may calculate from the information returned by the first LIFN server returned which next LIFN server to contact that could have provided the location for the network resource.

Indeed, Moore teaches “[t]he client can also avoid fetching the resource if it can tell by the catalog record that it is not needed or that a locally-cached copy exists,” AX0368, fairly suggesting that a client-centric approach is beneficial over a LIFN-server centric approach where that server does all the work. Because the LIFN servers store key-value assertions, *see* AX0374, the relevant artisan would have known to use the information passed back by the first LIFN server to identify the next LIFN server that contains the location corresponding to the key. *See also* AX1839 at 3:21-24 (“An alternative approach is for clients to use a standard hash function to determine from which server to request a document.”). The relevant artisan would thus have arrived at the limitations of claim 17-18 and 24.

3. SNQ 3: Jaeger and Netlib-Related Publications

Claims 17-18 and 24 would also have been obvious under a combination of Jaeger and Netlib-related publications. For example, Jaeger’s file identifiers would have been extended to: (1) Netlib’s LIFNs to satisfy a byte-by-byte uniqueness supporting the one-to-one view of identifier-entity mappings; or (2) Netlib’s URNs to meet a more generalized many-to-one understanding of

those mappings. Similarly, Netlib’s architecture would have been extended so the first LIFN server (or indeed catalog server) returned an array of other similarly situated servers as Jaeger taught so the architecture would have become more client-centric, leading to local caching and so a “simple, near-random access” with $O(1)$ lookup time, AX0511; AX0530; AX0328; AX0368, and more security and trust (less lookups across the network) in the model, AX0323; AX0331, and AX0511.

A combination of Jaeger and Netlib-related publications—indeed, any substitution of element(s) of one for the element(s) of the other—would have been a combination based on similar structure and function for related prior art. Where two known alternatives are interchangeable for a desired function, the relevant artisan would not even have needed an express suggestion to substitute one for the other to render a substitution obvious. *In re Fout*, 675 F.2d 297, 301 (CCPA 1982); *In re Siebentritt*, 372 F.2d 566, 568 (CCPA 1967); *see KSR Int’l Co. v. Teleflex Inc.*, 550 U.S. 398, 417 (2007). That said, both Jaeger and Netlib-related publications use intermediary location servers to decouple a network resource from its location for dynamic lookups for distributed networks such as the Internet. Both also use unique identifiers to identify network resources: file identifiers (where each file is described as “a stream of bits”) in Jaeger; and URN and LIFN in Netlib-related publications. The relevant artisan would thus have combined the two set of teachings to arrive at the claimed limitations of claim 17-18 and 24.

C. Detailed Explanation of the Pertinency and Manner of Applying the Cited Prior Art to Each Challenged Claim

Claims 17-18 and 24 would have been obvious in view of Jaeger, Netlib-related publications, or their combinations, given the discussion in § V.B.1-3 (incorporated herein by reference) and further in view of the discussion below.

1. Claim 17

a. A system for retrieving data location information for data stored in a distributed network, the system comprising:

Jaeger, Netlib-related publication, or their combination discloses or fairly suggests such a system. AX0321-0331 (Jaeger); AX0332-0338 (Browne I); AX0339-0346 (Browne II); AX0347-0356 (Browne III); AX0357-0372 (Moore); AX0373-AX0385 (Fagg). For example, Jaeger suggests a lookup system for “automatically” (AX0321-0322) “retrieving” (AX0321) files—where a file is defined as “a stream of bits referenced by a file identifier,” AX0322—“distributed over the Internet,” AX0321-0322; AX0322 and AX0329-0330 (disclosing a shell interpreter

implementing Jaeger's service). Similarly, Netlib-related publications disclose a location-independent naming system for network resources for the Internet, a distributed network. *See, e.g.*, AX0332-AX0333.

b. a plurality of data repositories configured to store data, wherein the data is associated with a respective identifier string in each data repository;

Jaeger discloses or fairly suggests this limitation because it discloses that the relevant artisan could implement "an unlimited number of distribution servers," AX0325, and each "can process a file identifier and determine if it possesses a file that maps to the identifier," AX0322, where a "file identifier" uniquely defines a file, which in turn is defined as "a stream of bits," AX0322, and is the recited "data"; AX0323 (disclosing a plurality of "distribution servers," each "connected to the Internet"); AX0325 ("Our service is implemented as a set of communicating principals distributed in an untrusted network. There are six types of principals in the implementation," among them, "distribution servers").

To the extent a broader many-to-one understanding of the recited "identifier" is warranted, the relevant artisan would have readily discerned that Jaeger's file identifier could have been mapped not just to a single file on a distribution server, but alternatively to even an entire file directory entity that resolved to that identifier. Jaeger discloses "[w]e expect that people responsible for distribution servers will know what files they want to offer, such as in the WWW where users make file URL's available." AX0322. The relevant artisan would have known, for instance, that the contents of a web-published folder may resolve to a single URL reference. Indeed, the file or folder or contents of such web-published file or folders need not even be stored on a single distribution server because Jaeger's "publisher process" takes responsibility for notifying the distribution server of what may be served as a network resource. *See, e.g.*, AX0326 ("[O]ur implementation has a publisher process that uploads file location information to both location servers and distribution servers."). Prior art in the same field similarly disclosed that retrieving a file by its identifier may allow underlying dependencies to be resolved and retrieved. *See, e.g.*, AX0333 ("[T]he Netlib Software Repository provides dependency checking that allows the user to retrieve a toplevel routine plus all routines in its dependency tree (i.e., those routines that are called directly or indirectly by the top-level routine).").

In the same vein, to the extent Jaeger's file identifier needed to be further extended to help

craft a more extensible “identifier” framework useful for locating distributed network resources, the relevant artisan would have also readily referenced URNs and LIFNs, predictable variations of file-identifiers. The relevant artisan would therefore have necessarily been motivated to enhance Jaeger’s “file identifier” to refer to whatever could have been understood as “a stream of bits,” AX0322. Unsurprisingly, RFC 1737 taught in 1994 URNs, AX0386-0387; and Netlib-related publications then disclosed LIFNs, drawing upon the URN-related disclosures, AX0332-0333. Accordingly, the motivation to make any requisite modifications to Jaeger’s file identifier is flatly disclosed in prior art. Indeed, Browne I explained that it leveraged prior work with URNs to identify network resources and devise the use of LIFN. AX0333. It explained the motivation: “With a large number of less familiar repositories, . . . it becomes necessary to establish interoperable trust mechanisms and to reduce the number of parties with whom the user must establish trust.” AX0333. Those “problems can be alleviated by implementing a location-independent naming system,” AX0333.

Netlib-related publications similarly disclose or fairly suggest multiple data repositories, in each of which a LIFN or a URN identifies data. *See, e.g.*, AX0342 (Fig. 2) (showing physical repositories); AX0342 (“The NHSE’s approach to these issues will be to implement a location-independent naming architecture that unambiguously associates a unique name, called a location-independent filename (LIFN), with the byte contents of a published asset.”); AX0352-0353 (“The processing involves retrieval of the underlying files making up the contribution, fingerprinting these files, assigning the contribution a unique identifiers. After the files making up a contribution have been received, each of the file is fingerprinted using a hash function, which in turn is used to construct a unique identifiers for the submission called LIFN The submission can subsequently be retrieved from the NHSE software catalog by specifying its LIFN.”); AX0333 (discussing “a naming system that provides for two levels of naming. The binding between a lower-level name (called a LIFN) and file contents is unchangeable and verifiable. . . . A higher-level name (called a URN) is associated with a cataloging record that includes the lower-level name as well as other descriptive information.”).

- c. a data location sewer [sic] network having a plurality of data location servers, each of the plurality of data location servers containing location strings associated with respective identifier strings and each of the plurality of data location servers having

computer executable code configured to execute the following steps:

Jaeger discloses or fairly suggests a location server network having more than one location servers wherein the servers store key-value associations like that which is claimed. AX0324

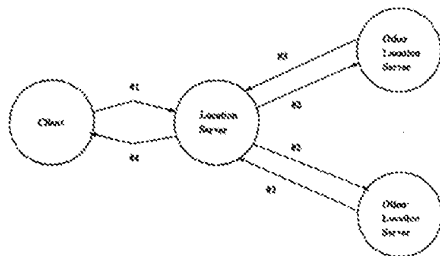


Figure 3: The file location protocol: (1) Client sends a file identifier to the client's location server to find the distribution servers that possess a file; (2) Client's location server requests that other location servers find the distribution servers that possess a file (optional); (3) Other location servers return to the client's location server a set of distribution servers that possess a file; (4) Client's location server returns a set of distribution servers that possess the file.

(“**Location server:** Principals that map file identifiers to distribution servers”); AX0327 (Figs. 2-3). Jaeger relatedly discloses “a publisher process that uploads file location information to both location servers and distribution servers” and that “[l]ocation servers store a map of file identifiers to the set of servers that provide them.” AX0326 (emphases added); AX0327 (annotation to Figure 2: “Publisher uploads a file identifier-to-distribution server mapping to location server.”).

Alternatively, the relevant artisan would have readily discerned that Jaeger function appears similar to a naming system and so key-value mappings would have been stored as hash tables spread across Jaeger’s location servers—in other words, that each location server maintains a partial set of key-value mappings in its hash table. AX0450 (“the database that makes up the domain space is distributed among various name servers”); AX0398 (each “name server” has “information about *a subset* of the domain space, and pointers to other name servers that can be used to lead to information from any part of the domain tree” (emphasis added)); *see also* AX0193 (examiner noting that “naming servers inherently implemented tables, with mapping hashing functions to lookup location/address/etc. information, as well as to effect direction of naming requests to other naming servers.”).

Jaeger is focused on *automating* the search and retrieval of files, so it follows that the location servers have computer executable code implementing its role in the topology. *See, e.g.*, AX0322 (“The goal of our service is to automate the file location, retrieval, and authentication tasks of the client.” (emphasis added)); AX0331 (“Automated location is possible because all remote files are published with location servers available to clients. Clients can query a location server for the identity of the distribution servers that possess a copy of a file. Clients then retrieve a copy of the file from one of the distribution servers. Clients control the location and retrieval protocols, which enables a client to retrieve a file without having to trust either the location servers or the distribution servers.”); AX0321 (“Our service provides the capability to automatically

locate, retrieve, and verify files specified by a client.”).

Netlib-related publications similarly contemplate multiple locations servers that maintain a LIFN-location mapping and have computer executable code to assist clients or users directing their search queries to the location servers. *See, e.g.*, AX0335-0336 (“The LIFN-to-location mapping service is provided by a network of LIFN servers, collectively called the LIFN database. These servers process queries for locations of LIFNs.”). The client queries one of those location servers, which returns the locations of the file identified by the LIFN. AX0363. The publications also treat the LIFN servers (with LIFN-to-location associations) analogous in structure to the catalog servers (with URN-to-LIFN associations). *See e.g.*, AX0369 (“Catalog information stored in an RCDS server is currently assumed to be ‘flat’ lists of (attribute, value) pairs. Clients can retrieve portions of the catalog for a resource by the attribute’s full name or by the prefix of a name.”); AX0374 (“RCDS accomplishes this by replicating the resources and metadata at a potentially large number of (perhaps geographically dispersed) locations. The set of locations for a resource are maintained in a highly distributed and replicated location registry [in the LIFN database]. **Similarly**, the metadata for a resource (a list of attribute ‘name=value’ pairs called assertions) are maintained in a separate distributed and replicated registry, which is indexed by the resource’s Uniform Resource Locator (URL) or Uniform Resource Name (URN). The metadata for a resource is self-defining and can contain elements from arbitrary schema or data models.” (emphasis added)). Browne II explains that “automatic search mechanisms will reduce the time scientists and research engineers spend dealing with information overload,” AX0340, fairly suggesting to the relevant artisan that computer code handles the requests from clients or users.

The relevant artisan would have also readily recognized Jaeger and Netlib-related publications to be directed to a highly dynamic naming service with frequent updates as network resources (files) change locations where they are stored. The artisan would have therefore resorted for alternative design choices to RFC 1034, which describes the use of “name servers,” each of which has “complete information about *a subset* of the domain space, and pointers to other name servers that can be used to lead to information from any part of the domain tree.” AX0398 (emphasis added); AX0450 (“From the resolver’s point of view, the database that makes up the domain space is distributed among various name servers.”). It also explains that the “response by the name server either answers the question posed in the query [or] refers the requester to another set of name servers,” AX0408; AX0193 (examiner observing that “naming servers inherently

implemented tables, with mapping hashing functions to lookup location/address/etc. information, as well as to effect direction of naming requests to other naming servers”).

Using hash tables to store key-value associations in the distributing networking environment was well-known. AX0172 (taking “Official Notice (see MPEP § 2144.03) . . . that the use of a [] hashing operation in a computer networking environment was well known in the art at the time the invention was made.”). The relevant artisan would have recognized that doing so would have simplified a lookup as the distributed system scaled. AX1466 (“Searching is the most time-consuming part of many programs, and the substitution of a good search method for a bad one often leads to a substantial increase in speed.”). It was well-known that “it is often possible to arrange the data or the data structure so that searching is eliminated entirely” and “to avoid all this rummaging around,” “[a] third possibility” is to “comput[e] a function $f(K)$ which is the location of K and the associated data in the table.” AX1466, AX1495-1497; AX1496 (“We compute a *hash function* $h(K)$ and use this value as the address where the search begins.”); AX0511 (presenting “two excellent, general-purpose hash functions,” including “HashPJV(),” which “accept[s] a pointer to a string and return an integer”); AX0530 (“[T]he expected time to search for an element in a hash table is $O(1)$ Instead of using the key as an array index directly, the array index is *computed* from the key.”). The relevant artisan would have thus readily recognized location servers using hash tables as the underlying structure to store their identifier-location mappings.

- d. in response to receiving a data location request from a client to retrieve a location string associated with an identification string provided in the data location request, transmitting a redirect message to the client if the identification string is not associated with a location string at the data location server, wherein the redirect message contains information for use by the client to calculate a location of a different data location server in the plurality of data location servers, wherein the different data location server contains the location string.

See, e.g., § V.C.1.c.

Jaeger discloses or fairly suggests “a data location server receiving a data location request from a client to retrieve a location string associated with an identification string provided in the data location request” because it discloses that a “client sends a file identifier to the client’s location

server.” AX0327. Jaeger discloses that “the location server can list the set of distribution servers that claim to possess a file that matches a file identifier,” AX0327—that is, the location servers maintain a list of key-value pair that represents “file identifier”-“distribution server” mappings, which fairly suggest that Jaeger’s location servers inherently implemented hash tables. AX0193 (examiner observing that “naming servers inherently implemented tables, with mapping hashing functions to lookup location/address/etc. information, as well as to effect direction of naming requests to other naming servers.”); *see also* § V.C.1.c (discussing that the relevant artisan would have readily recognized location servers using hash tables as the underlying structure to store their identifier-location mappings).

Jaeger also discloses or fairly suggests the “redirect message”-related limitation. First, Jaeger’s client’s location server contacts other location servers “only done if the client’s location server has no knowledge of any distribution servers that possess the file,” AX0327—in other words, when the client’s location server does not maintain the mapping between the file identifier and the corresponding location of the distribution server. Second, “[t]he client’s location server sends the file identifier to some number of the other location servers.” AX0327. “The[] other location servers return location structure messages to the client’s location server,” wherein each location structure message has “[a]n [a]rray of [d]istribution [s]ervers,” AX0327; AX0511 (“Hash tables are just like most tables” and “the old-fashioned word for ‘arrays.’”); AX0530 (“A hash table is a generalization of the simpler notion of an ordinary array.”). The client’s location server then “returns a location structure message to the client.” AX0327. Jaeger further discloses that “the client collects a set of distribution servers (possibly empty) that the client’s location server claims possess the requested file.” AX0328. This allows the client to use the information in the location structure message to contact the distribution server that has files corresponding to the file identifier.

The relevant artisan would have recognized that a simple level of indirection would have enhanced Jaeger’s system’s security and performance. *Cf.* AX0368 (discussing how using “one extra long-distance round-trip” to a location leads to “promotes efficient use of the network”). “Searching is the most time-consuming part of many programs, and the substitution of a good search method for a bad one often leads to a substantial increase in speed. In fact it is often possible to arrange the data or the data structure so that searching is eliminated entirely, i.e., so that we always know just where to find the information we need.” AX1466. The relevant artisan would have therefore modified the client location server to return an array of location servers. The

relevant artisan would have recognized that this added level of one-time indirection would have enabled local caching, which would have benefits in the long run as clients would have looked up locations for other identifiers.

- Performance: Prior art disclosed “cach[ing] the information locally.” AX0322; AX0368 (“The client can also avoid fetching the resource if it can tell . . . that a locally-cached copy exists.”); AX0328 (“The client’s location server caches any new information it obtains from these other location servers. Therefore, the next time a client requests the same file, the location server can provide the set of distribution servers found previously.”). The relevant artisan would have recognized that “lookup operations are a significant factor in system performance.” AX1851 (Jaeger cites this paper as citation [5] at AX0331). A lookup operation across the network, even to its location server, for each file identifier would have been understood to be more expensive and, conversely, a local lookup using a hashing function into a local array of “file identifier”-“location server” mappings would have been understood to be a much faster operation. AX1495 (teaching table lookup as “avoid[s] all this rummaging around by doing some arithmetical calculation on K , computing a function $f(K)$ which is the location of K and the associated data in the table”); AX0511 (table lookups provide “simple, near-random access”); AX0530 (“[T]he expected time to search for an element in a hash table is $O(1)$ Instead of using the key as an array index directly, the array index is *computed* from the key.”).

- Security: The relevant artisan would have preferred building a table of other location servers at the client’s location server (recursively, in a hierarchical organization of location servers; or iteratively, in a peer-to-peer organization of location servers; or a combination approach). Not passing the file identifier to other location servers thus would have enhanced the security of the system, particularly because: (1) less number of “system principals are trusted” to use or handle “file identifiers,” especially in an untrusted distributed network, AX0322; and (2) a single principal, the client, would have then been in charge of future lookups, further reducing reliance on across-the-network communications, which is consistent with Jaeger’s teaching that preferably “[c]lients control the location and retrieval protocols, which enables a client to retrieve a file without having to trust either the location servers or the distribution servers,” AX0331. Prior art “require[d] . . . too many system principals,” AX0322, which fairly suggests that a single point of failure-and-retry (at “client”) is preferable to multiple points of failures-and-retries (at the “location servers”); AX0323 (“Trust in the location server and its communications breaks down if

it is subject to compromise or communicates over an untrusted network.”). Indeed, Jaeger “expect[ed] that distribution servers and location servers connected to the Internet will be attacked, so applications, replicated data, or communications delivered from these machines cannot be trusted.” AX0323. The relevant artisan would have thus recognized that local caching would enhance security, a necessary design consideration for retrieving data on “untrusted network” such as the internet. AX0321. The single point of failure-and-retry would also allow repeatedly referencing a local certificate authority, a “trusted principal” in line with Jaeger, AX0324-0325, given that Jaeger presumes that “a client believes any statement made and digitally signed by a trusted authority,” AX0323.

“[A] person of ordinary skill has good reason to pursue the known options within his or her technical grasp.” *KSR International Co. v. Teleflex Inc.*, 550 U.S. 398, 421 (2007). Here, the relevant artisan would have pursued this alternative design choice, which would have been within his or her technical grasp, particularly given that Jaeger relies on well-known data structures such as arrays/tables. Jaeger itself contemplates such enhancements: It expressly categorizes the client’s location server returning an array of distribution servers as “[o]ptional” and strives that the “[c]lients control the location and retrieval protocols.” AX0330-0331.

The relevant artisan would alternatively have modified the architecture disclosed in the Netlib-related publications for the same or similar reasons as set forth above, given that the publications contemplate that “[g]rowth in the popularity of the Internet and the World Wide Web, as well as the wide availability of WWW client and server software, has accelerated the shift from centrally maintained software repositories to virtual, distributed repositories.” AX0332.

The relevant artisan perusing Jaeger or Netlib-related publications would also have relied on common sense to modify their implementations. *Perfect Web Techs, Inc. v. InfoUSA, Inc.*, 587 F.3d 1324, 1328 (Fed. Cir. 2009) (“Common sense has long been recognized to inform the analysis of obviousness if explained with sufficient reasoning.”); *Arendi S.A.R.L. v. Apple Inc.*, 832 F.3d 1355, 1361 (Fed. Cir. 2016) (“[W]e do consider common sense, common wisdom, and common knowledge in analyzing obviousness.”). Both sets of publications put a heavy premium on trust, unsurprisingly considering the distributed nature of the networks across which the resources must be located. AX0323 (“Trust in the location server and its communications breaks down if it is subject to compromise or communicates over an untrusted network.”); AX0333 (“With a large number of less familiar repositories, . . . it becomes necessary to establish interoperable trust

mechanisms and to reduce the number of parties with whom the user must establish trust.”).

It follows therefore that, under either set of prior art references, the relevant artisan would have resorted to a local client-centric approach, in which the client looks up in a table of location servers returned by the client’s location server. With a single application of a well-known hash function, the client would have calculated the index into that table and determined the location server that would have maintained the mapping associated with the file identifier. That would enable the client to then look up into the correct location server to retrieve the location.

2. Claim 18

- a. A system for retrieving data location information for data stored in a distributed network, the system comprising:

See, e.g., § V.C.1.a.

- b. a client responsive to a data query to query a data location server for location information associated with the identifier string;

Jaeger discloses or fairly suggests such “a client” because it discloses “clients, which are principals that request files by sending a file identifier to its location server. *See, e.g.,* AX0327; AX0327 (Jaeger at Figure 3). Netlib-related publications too disclose such a client. AX0342 (Figure 2); AX0342 (Figure 3); AX0360 (explaining clients as one of the components of RCDS, describing them as the consumers of the resources provided by the system); AX0363 (disclosing that the client queries location servers).

- c. a data location server network comprising a plurality of data location servers, at least one of the plurality of data location servers containing location information associated with the identifier string, wherein each of the plurality of data location servers comprises computer executable code configured to execute the following steps in response to receiving a data location request from the client:

See, e.g., § V.C.1.c.

- d. if the data location server contains the location string associated with the identification string provided in the data location request, the data location server transmits location information for use by the client to calculate a location of the data associated with the

identification string:

See, e.g., § V.C.1.c-d.

Jaeger discloses or fairly suggests this limitation in its Figure 1 and related description. AX0324. Netlib-related publications too disclose or fairly suggest this limitation. AX0342 (Figure 3 illustrating the “[l]ocation server” returning a list of URL locations associated with a “[f]ilehandle); AX0362 (“Among the services provided by RCDS [is] . . . LIFN-to-location mapping.”); AX0363 (teaching that the client queries one of the location servers, which returns the locations of the file identified by the LIFN, information that can then be used by client to choose one of the file servers and fetch the file from that file server); AX0360, AX0365 (disclosing RCDS architecture in figure 2, in which location servers maintain information about the locations of network resources and accept queries for location data from clients).

- e. if the data location server does not contain the location string associated with the identification string, the location server transmits a redirect message to the client, wherein the redirect message contains redirect information for use by the client to calculate a location of a different data location server in the plurality of data location servers, wherein the different data location server contains the location string.*

See, e.g., § V.C.1.c-d.

3. Claim 24

- a. The system of claim 18, wherein*

See, e.g., § V.C.2.

- b. the location information comprises a portion of a hash table distributed over the plurality of data location servers.*

See, e.g., § V.C.1.c-d.

Jaeger discloses or fairly suggests this limitation because it discloses multiple location servers each keeping a portion of the universe of “file identifiers”-“file location” mappings. Jaeger further teaches that “[l]ocation servers store a map of file identifiers to the set of servers that provide them,” AX0326 (emphasis added), and that the location server may return a “possibly empty” set of locations, AX0328. All of that fairly suggests that the location information at any particular location server may just be only a portion—indeed, including a null set—of the entire

set of associative values (stored at the distribution servers. *See also* AX0172; AX1686-1688.

Netlib-related publications similarly contemplate multiple locations servers that maintain LIFN-location mappings and catalog servers that maintain URN-LIFN mappings. The publications fairly suggest that both stored key-value pairs “[s]imilarly.” AX0369 (“Catalog information stored in an RCDS server is currently assumed to be *‘flat’ lists of (attribute, value) pairs.*”); AX0374 (“The set of locations for a resource are maintained in a highly distributed and replicated location registry. **Similarly**, the metadata for a resource (a list of attribute *‘name=value’ pairs* called assertions) are maintained in a separate distributed and replicated registry[.]” (emphases added)).

The relevant artisan would have at first look understood disclosures in Jaeger and Netlib-related publications to teach a naming system, and would have therefore referenced RFC1034, which describes the use of “name servers.” The name servers each have “complete information about *a subset* of the domain space[.]” AX0398 (emphasis added); AX0450 (“From the resolver’s point of view, the database that makes up the domain space is distributed among various name servers. Different parts of the domain space are stored in different name servers, although a particular data item will be stored redundantly in two or more name servers.”).

The relevant artisan would have alternatively recognized that “[h]ashing is useful anywhere a wide range of possible values needs to be stored in a small amount of memory and be retrieved with simple, near-random access.” AX0511. Indeed, “hash tables occur frequently in database engines and especially in language processors such as compilers and assemblers, where they elegantly serve as symbol tables.” AX0511; AX0530 (“Many applications require a dynamic set that supports only the dictionary operations INSERT, SEARCH, and DELETE. For example, a compiler for a computer language maintains a symbol table, in which the keys of elements are arbitrary character strings that correspond to identifiers in the language. A hash table is an effective data structure for implementing dictionaries.”). “In such applications, the table is a principal data structure” and “all accesses to the table must be done through the hash function.” AX0511; AX0530 (“[T]he expected time to search for an element in a hash table is $O(1)$ Instead of using the key as an array index directly, the array index is *computed* from the key.”). The relevant artisan would therefore have discerned that structure—that is, hash tables distributed across location servers—to undergird disclosures in Jaeger and Netlib-related publications.

VI. CONCLUSION

The requested reexamination should be ordered.