

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

GOOGLE LLC,
Petitioner

v.

KOVE IO, INC.,
Patent Owner

Patent No. 7,814,170

**DECLARATION OF DR. SANDEEP CHATTERJEE
IN SUPPORT OF PETITION FOR INTER PARTES REVIEW
OF U.S. PATENT NO. 7,814,170**

TABLE OF CONTENTS

I.	INTRODUCTION	1
II.	BACKGROUND AND QUALIFICATIONS	1
III.	MATERIALS REVIEWED	6
IV.	PERSON OF ORDINARY SKILL IN THE ART AND THE TIME OF THE ALLEGED INVENTION	8
V.	TECHNICAL BACKGROUND	9
A.	Distributed Computing Systems	9
B.	Hash Functions and Applications	12
VI.	THE '170 PATENT	15
A.	Description	15
VII.	OVERVIEW OF THE PRIOR ART	17
A.	Kahn	17
B.	Vingralek	23
VIII.	CLAIM CONSTRUCTION	26
A.	“identifier”	27
B.	“location information”	27
C.	“location server”	27
D.	“client”	28
E.	“the portion of the data location information included in a corresponding one of the data location servers is based on a hash function used to organize the data location information across the plurality of data location servers, and each one of the data location servers is configured to determine the at least one of the plurality of data location servers based on the hash function applied to the identifier string”	28
F.	“data pertaining to the entity”	29
G.	“plurality of location servers”	29

IX.	THE PRIOR ART DISCLOSES AND/OR SUGGESTS ALL RECITED FEATURES OF CLAIMS 1-17 OF THE '170 PATENT.....	32
A.	Kahn in View of Vingralek Discloses and/or Suggests Claims 1-17	32
1.	Claim 1	32
2.	Claim 2	67
3.	Claim 3	69
4.	Claim 4	73
5.	Claim 5	76
6.	Claim 6	79
7.	Claim 7	97
8.	Claim 8	99
9.	Claim 9	99
10.	Claim 10	100
11.	Claim 11	102
12.	Claim 12	104
13.	Claim 13	106
14.	Claim 14	108
15.	Claim 15	108
16.	Claim 16	112
17.	Claim 17	113
X.	CONCLUSION.....	115

I, Sandeep Chatterjee, declare as follows:

I. INTRODUCTION

1. I have been retained as an independent expert consultant in this proceeding before the United States Patent and Trademark Office (“PTO”) regarding U.S. Patent No. 7,814,170 (“the ’170 patent”) (Ex. 1001).¹ I have been asked to consider whether prior art references disclose or suggest the features recited in claims 1-17 (“the challenged claims”) of the ’170 patent. My opinions are set forth below.

II. BACKGROUND AND QUALIFICATIONS

2. My qualifications are stated more fully in my curriculum vitae, which is attached as Ex. 1003. Below is a summary of my education, work experience, and other qualifications.

3. I am the Chief Executive Officer of Experantis LLC (“Experantis”), a technology consulting company. Previously, I was the Co-founder, Executive Vice President and Chief Technology Officer of SourceTrace Systems, Inc., a technology and services company enabling the delivery of secure remote electronic

¹ In this Declaration, I refer to exhibits that I understand are to be attached to the petition for *Inter Partes* Review of the ’170 patent.

services over landline and wireless telecommunications networks.

4. I received my Bachelor's degree in Electrical Engineering and Computer Science from the University of California, Berkeley in 1995. I received my Master's degree in Computer Science from the Massachusetts Institute of Technology (MIT) in 1997, and my Doctorate in Computer Science from MIT in 2001. I received a certificate of completion for an executive education program on global leadership from Harvard University in 2011.

5. My doctoral dissertation at MIT, entitled "Composable System Resources for Networked Systems," involving networked and distributed computer systems and architectures, was selected as one of the top inventions in the history of MIT's Laboratory for Computer Science. This invention is showcased in a time capsule at the Museum of Science in Boston, Massachusetts. Other recipients of this honor include Bill Gates, the founder of Microsoft, and Tim Berners-Lee, the inventor of the World Wide Web.

6. As part of applications of my doctoral research, I developed hardware and software systems for intelligent environments within homes and offices. Some of these devices included televisions, digital picture frames, refrigerators and children's toys. The distributed computing system of these intelligent environments included distributed data storage and retrieval.

7. In 2011, I was named a Young Global Leader. This honor, bestowed each year by the World Economic Forum, recognizes and acknowledges the top leaders—all below the age of 40—from around the world for their professional accomplishments, commitment to society, and potential to contribute to shaping the future of the world. In 2016, I was appointed to the World Economic Forum's expert network as an expert in technology and innovation.

8. From 1997, I was the Entrepreneur-in-Residence at FidelityCAPITAL, the venture capital arm of Fidelity Investments. In 1999, I founded and served as President and Chief Technology Officer (CTO) of Satora Networks, which developed tools and technologies for building appliances and services for the Internet using wireless and other technologies to extend it beyond the desktop.

9. In 2001, I joined Bluestone Software's Mobile Middleware Labs as a Senior Engineer developing applications and systems infrastructure for enterprise Java/J2EE, Web services, and enterprise mobile solutions. After the completion of Hewlett-Packard's (HP) acquisition of Bluestone, I became a Senior Member of the Technical Staff at HP's Middleware Division. I was responsible for architecting and developing the company's next-generation Web services platform for enterprise as well as mobile environments, known as the Web Services

Mediator.

10. I was part of the Expert Group that developed the JSR-00172 J2ME (Java 2 Platform, Micro Edition) Web Services Specification, the worldwide industry standard for mobile Web services. I am the co-author, with James Webber, of the book “Developing Enterprise Web Services: An Architect’s Guide” (published by Prentice-Hall in 2004). This book has been adopted by over 100 universities and colleges around the world, and has been translated or reprinted in a number of countries around the world.

11. I have significant experience in developing complex computing systems. For example, through a contract between HP and the United States Agency for International Development (USAID), I architected and led the development of one of the first mobile banking solutions. This system enabled customers to use their mobile phones and other wireless handsets to connect with the core banking systems of banks and other financial institutions, and perform transactions without having to travel to bank branches. This system supported many banking transactions, including deposits, withdrawals, loan applications, loan disbursements and loan repayments.

12. Later, after SourceTrace Systems’ acquisition of this technology, I led the expansion of this solution into multiple countries and into multiple industries.

Banks and other financial services companies utilized this technology to make their tellers more efficient, to provide self-service ATMs within branches, and to provide remote access to banking services. Additionally, through our licensing agreement with Telefonica, one of the largest cellular and telecommunications companies in the world, this solution was deployed in various other industries, including logistics and asset management and customer relationship management. Bloomberg Television selected and featured this technology and the company I co-founded to commercialize this technology on Bloomberg TV's "Bloomberg Innovators" program.

13. I have been a retained expert witness for various disputes that involved significant technology issues, and I have been qualified as a technology expert by U.S. District and State Courts, including in California, Delaware, Florida and Texas in technology areas that are relevant to this case, including but not limited to: computer software systems, distributed computing systems, and mobile and wireless systems. I have previously testified through declaration or expert report, at deposition, and at trial in numerous intellectual property and commercial litigation matters, including for patent litigation, copyright and trade secret misappropriation litigation, and contract dispute cases. I have submitted more than one hundred and fifty expert declarations and expert reports, testified at deposition

more than eighty times, and testified at trial or at hearings at least twelve times. I have been identified as one of the top 1000 patent professionals in the world, and I am listed in the IAM Patent 1000, which identifies the world's leading patent litigation and prosecution attorneys, as well as damages and technology expert witnesses.

14. Experantis is being compensated for my time on this matter at my standard hourly rate and reimbursed for any expenses that I incur related to my work in this matter. Neither Experantis nor I have any financial interest in the outcome of this matter, and Experantis will be paid for my time regardless of the outcome of this matter.

III. MATERIALS REVIEWED

15. The opinions contained in this Declaration are based on the documents I reviewed, my professional judgment, as well as my education, experience, and knowledge regarding systems and processes in the field of distributed computing systems.

16. In forming my opinions expressed in this Declaration, I reviewed the following materials:

Ex. 1001	U.S. Patent No. 7,814,170 to Overton et al. ("the '170 patent")
Ex. 1003	Curriculum Vitae of Sandeep Chatterjee, Ph.D.

Ex. 1004	Prosecution History of U.S. Patent No. 7,814,170
Ex. 1005	WIPO Application No. 1997/043717 to Kahn et al.
Ex. 1006	U.S. Patent No. 6,473,781 to Skagerwall et al.
Ex. 1007	R. Vingralek, <i>Distributed file organization with scalable cost/performance</i> (1993)
Ex. 1011	Japanese Patent Application Publication No. H09-231180 to Sato et al.
Ex. 1014	RFC 768, User Datagram Protocol (Aug. 28, 1980)
Ex. 1016	U.S. Patent No. 5,764,756
Ex. 1056	Joseph B. Greene, <i>Oracle8 Server Unleashed</i> (1998)
Ex. 1057	Joseph B. Greene, <i>Oracle DBA Survival Guide</i> (1995)
Ex. 1058	Betty Salzberg, <i>File Structures</i> (1988)
Ex. 1059	Andrew P. Black & Yeshayahu Artsy, <i>Implementing Location Independent Invocation</i> (1989)

I also considered any other documents and materials I refer to in this Declaration.

17. My opinions contained in this declaration are based on the documents I reviewed and my knowledge and professional judgment. My opinions have also been guided by my appreciation of how a person of ordinary skill in the art would have understood the state of the art, the prior art, and the claims and the specification of the '170 patent at the time of the alleged invention, which I discuss below.

18. I have been asked to consider that the time of the alleged invention of the '170 patent was around 1998 (including and up to July 8, 1998), which corresponds to the filing date of the earliest patent application associated with the '170 patent.

19. Based on my experience and expertise, it is my opinion that certain references disclose and/or suggest all the features recited in challenged claims of the '170 patent, as I discuss in detail below.

IV. PERSON OF ORDINARY SKILL IN THE ART AND THE TIME OF THE ALLEGED INVENTION

20. I am familiar with the level of ordinary skill in the art regarding the '170 patent as of around July 8, 1998. Considering the '170 patent, the technology, the educational level and experience of workers in the field relating to the patent, and problems and solutions in that field, and drawing on my own experience, I believe a person of ordinary skill in the art at the time of the alleged invention (around July 8, 1998) would have had an undergraduate degree in electrical engineering, computer engineering, computer science or a related field along with two years of work experience in the field of distributed computing systems.

21. My opinions in this Declaration regarding the '170 patent and the prior art (including the state of the art) are from the perspective of one of ordinary

skill in the art as I defined above, during the relevant timeframe (e.g., the time of the alleged invention), which I discussed above as being around July 8, 1998.

V. TECHNICAL BACKGROUND

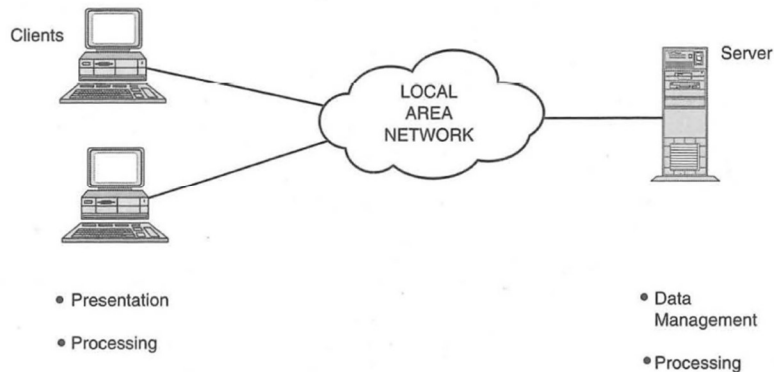
22. Below, I present a brief overview of certain aspects of systems and processes relating to distributed computing systems at the time of the alleged invention for the '170 patent. The functionalities and concepts I describe below in this technical background section reflect the state of the art that a person of ordinary skill in the art would have had knowledge of and understood prior to and at the time of the alleged invention of the '170 patent. I rely on, and incorporate as applicable (even if not expressly mentioned below in Section IX), the following disclosures and opinions to support my opinions in this Declaration, including those opinions relating to how the prior art discloses and/or suggests the challenged claims of the '170 and how and why a person of ordinary skill in the art would have been motivated to consider and combine the disclosures and suggestions from that prior art as I explain below in Section IX. The concepts below would have been within the knowledge and mindset of a person of ordinary skill in the art at the time.

A. Distributed Computing Systems

23. A distributed network is a network where data is distributed instead of

stored centrally. (Ex. 1005, 73:27-31 (“This strategy assures a random distribution of handle server requests among many handle servers distributed on a network without a central nodal point in the system (for reliability).”); *see also* Ex. 1006, 8:15-19 (“It is a key aspect of the invention that the data base of object data is distributed over a plurality of directory servers DS; DS; DS1-DSn, allowing that the number of objects can be scaled almost arbitrarily by adding new directory servers to the existing ones for storing new object data.”).) Within the distributed network are multiple databases which are connected to operate as a singular database. (Ex. 1056, 164 (“The ideal form, where a large network of connected databases act as one database, replicating one another’s data, is a *distributed database*. A distributed database is singular in that although data physically exists at many sites, it is still one database at the highest level that is simply distributed physically....”); Ex. 1001, 15:15-37 (disclosing that hash functions were well known at the time of the alleged invention).) “In distributed databases, you have multiple, completely equal databases,” where the distributed databases may use a client-server-architecture to accomplish tasks. (Ex. 1057, 680.) As shown with reference to Figure 46.1 below, a client-server architecture in its most simple form includes “two intelligent devices (your desktop PC and the server) that share the work load between them.” (Ex. 1057, 674.)

Figure 46.1.
Client-server
architecture.



Alternatively, there may be a network environment that involves a client, a database server, and an application server (which handles most of the application processing). (*Id.*)

24. Distributed network application servers can take applications and provide “reliable, scalable, and robust applications services.” (Ex. 1056, 36 (“Network computing is about universal access to data...universal access to applications...network services...and bringing new applications online faster and at a lower cost.... Powerful applications servers will take simple objects such as Enterprise JavaBeans and turn them into reliable, scalable, and robust applications services.”).) In particular within a distributed network, “load distribution and load balancing are basic means of improving performance.” (Ex. 1011, ¶[0003] (“Furthermore, load distribution and load balancing are basic means of improving performance in distributed systems.”).)

B. Hash Functions and Applications

25. A hash function takes an input, usually in the form of a key, and calculates the location of information (e.g., an address) associated with that key. (Ex. 1058, 202 (“Suppose that we knew that the keys of the records of some file were the numbers from 1 to 1000. Then we could store the record with key 1 in position 1 on the disk, the record with key 2 in position 2, and so on. Then to find a record with key x, we could go directly to position x. The value of the key would yield the address of the key. This would be faster than using a B+-tree for individual record searches. This would be a one-disk-access method. Hashing is a generalization of this process.”).) For example, “[a] *hashing algorithm* or *hash function* is a means of calculating the disk address of a block containing a given record from the value of its key.” (*Id.*)

26. “The point of hash functions is to take a highly nonuniform distribution—probably one with data unevenly clustered—and transform it into one which is nearly uniform.” (*Id.*, 203.) For example, if data is stored non-linearly, a hash function is designed to be “fast for fetching, inserting, and deleting” files within a system. (*Id.*, 215.) Hash functions are also optimal “[w]hen a large number of individual records must be fetched.” (*Id.*, 241 (“Hashing is the best file structure for single-record fetches. **When a large**

number of individual records must be fetched, as with automatic teller machines or airline reservations, **this is the best method.**”); *see also id.*, 215 (“Usually, right after a hash table has been created, fetching takes about one disk access. This is the method of choice for many files and many database management systems. **When most of the operations on the file are individual-record fetches, hashing is the optimal file structure.**”).)

27. Hash functions are used in many applications, but as an example, a hash function can be used to locate a tax record associated with a social security number (SSN) within a distributed system, where the SSNs are not stored numerically. (*Id.*, 209 (“For example, suppose that the California tax records are organized by a hash table based on social security number. This means that there is a randomizing function calculated from the social security number and a remainder taken from this number. The records are *not* stored in order of social security number. Two records whose social security numbers differ only in the last digit may be nowhere near each other.”).) To locate a tax record associated with a SSN, a hash function is applied to the SSN. (*Id.*) The result of the hash function is then used to fetch the tax record. (*Id.*, 209-210 (“If we want to read the records of all the California taxpayers whose social security number is between 000-00-0000 and 000-99-9999 from the file of California tax records, we...have to make 1 million

separate fetches for the 1 million different numbers between 0 and 999,999.....

Making one million hash-table fetches will take about 7 hours.”), 242 (“For creating a hash table from an already existing large file, as happens initially for the linear-hashing table or repeatedly for the traditional hashing method, we need to sort the records by their hash value and create the table in the last pass.”).)

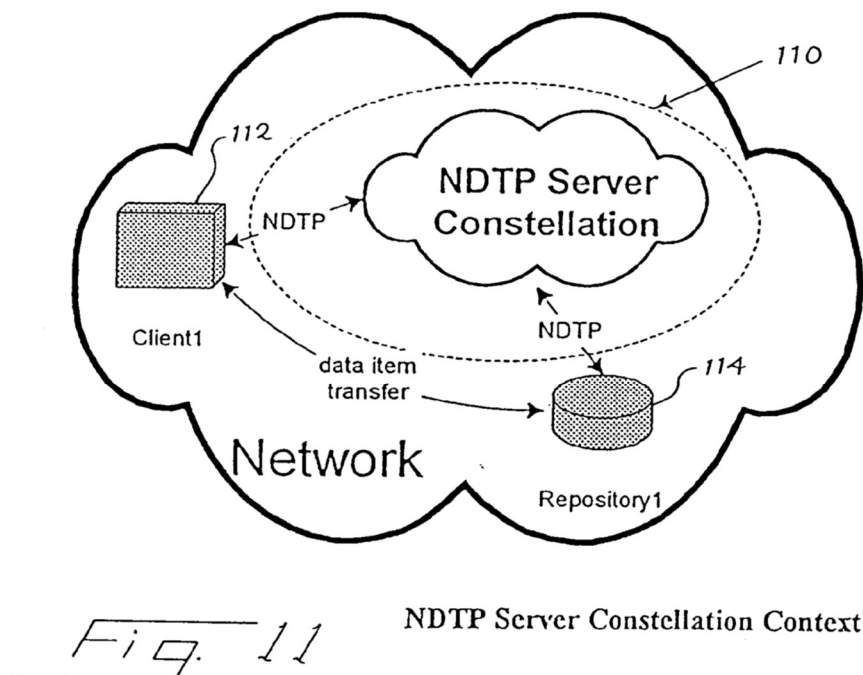
28. When data is not found at the specified location, perhaps due to the data being stored in a new location in the system, a message with a forwarding address may be returned to the requesting system. (Ex. 1059, 553-54 (“If object o is at node n after its a^{th} move, then the pair $\langle n, a \rangle$ is a *temporal address descriptor* (*tad*) for o If o later migrates to m , its descriptor on n is updated to include new *tad* $\langle m, 1 \rangle$. Assuming that objects do not migrate too often, this *tad* represents a forwarding address that is likely to remain correct for some time.... **If o is no longer at m , then m will normally have a newer *tad* for o . What should m do? There are two alternatives: (1) return a failure indication along with its stashed *tad*, or (2) propagate the invocation to the location indicated in its *tad*.** In Hermes, we chose a hybrid of the two approaches. If the invoked node does not have the object, the invocation is propagated to the location indicated in the *tad*; this continues up to some small number of hops.... **Otherwise, a failure with a newer *tad* is returned to the originator of the search, n in this example, which**

starts afresh with the new forwarding pointer.”).) For example, if the tax record is no longer stored at the original location, a message can be returned to the requesting system with the updated location. (*Id.*)

VI. THE '170 PATENT

A. Description

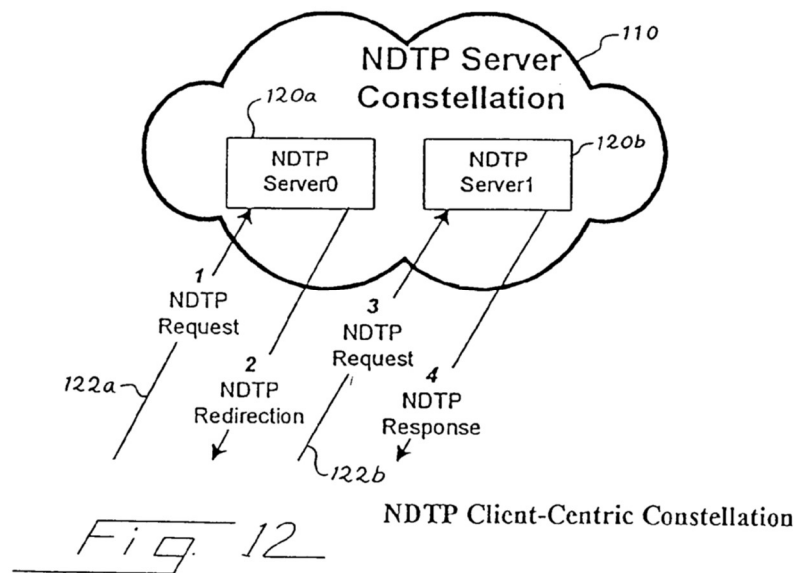
29. The '170 patent describes storing and retrieving information across a network (Ex. 1001, 1:29-32, 1:58-60) and a network distributed tracking protocol (NDTP) (*id.*, 2:14-16, 4:27-38). Figure 11 shows an NDTP server constellation where a client consults servers that return strings in response to a client request. (*Id.*, 17:3-6.)



(*Id.*, FIG. 11.)

30. Server constellations have two organizational approaches: client-centric (shown in Figure 12) and server-centric (shown in Figure 13). In the client-centric approach, processing emphasis is toward clients rather than toward servers.

(*Id.*, 18:25-27.)



(*Id.*, FIG. 12.) In contrast, in the server-centric approach, server processing is emphasized. (Ex. 1001, 18:50-54.)

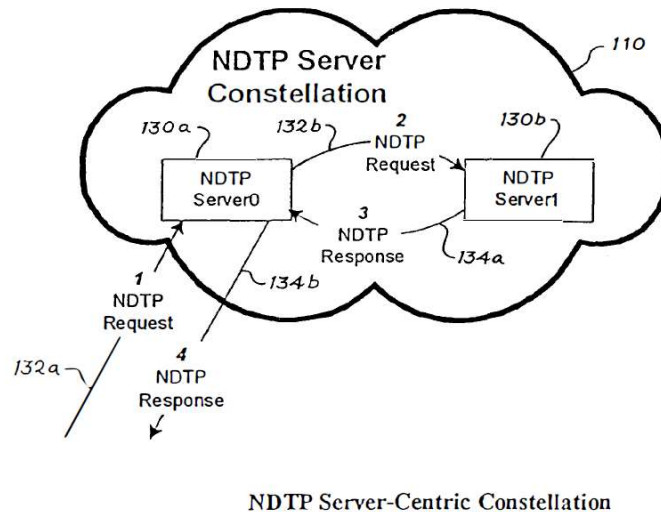


Fig. 13

(*Id.*, FIG. 13.)

31. In both the client-centric and server-centric approaches, a client transmits an identifier specifying an entity to a server in a client request. (Ex. 1001, 2:14-45.) If the server contains location information specifying network locations of data associated with the entity, the server returns the location information. (*Id.*)

VII. OVERVIEW OF THE PRIOR ART

A. Kahn

32. Kahn discloses a method for managing digital objects stored in one or more repositories in a network, where the digital objects are located by one or

more pointers associated with a handle. (Ex. 1005, Abstract (“A method of managing digital objects (1004) in a network is presented. The objects are stored at [repository] 1002.... Other aspects include providing multiple servers (1042) each of which accepts identifiers of a subset of digital objects and returns corresponding pointers to the locations of the digital objects in the network.”), 6:24 (“There may be multiple repositories.”).) Kahn explains that multiple handle servers in the network each may accept handles of the digital objects and return corresponding pointers that indicate the digital objects’ location in the network. (*Id.*, Abstract, 6:28-35 (“A service, accessible on the network, is provided for locating the handle associated with a digital object. The handle is used to obtain a pointer to the network location of...the digital object.”), 7:5-20 (“A handle server, accessible on the network, provides the pointer in response to the presentation of a handle. Multiple servers provide the service, each serving a portion of the handle space. []). Hash values are computed on the handles and the hash values are distributed among multiple handle servers, each handle server having a table which associates handles with pointers.”), 21:6-24 (“[M]ultiple servers...provide...handle-to-pointer translation services (called ‘handle servers’)....[]. It should also be clear that services can be provided on a widely distributed basis by multiple similar servers at different locations on the network, or by single centralized servers.”).)

33. Kahn discloses with reference to Figure 1 a high-level diagram of an exemplary system that includes a repository 1002 (highlighted in blue) for storing digital objects 1004 (highlighted in green) in the network, where the digital objects are identified by their handle 1012 (highlighted in yellow) and associated pointers. (*Id.*, Abstract (“A method of managing digital objects (1004) in a network is presented. The objects are stored at [repository] 1002.... Other aspects include providing multiple servers (1042) each of which accepts [handles (1012)] of a subset of digital objects and returns corresponding pointers to the locations of the digital objects in the network.”).) Handle-pointer mappings are stored in tables within handle servers 1042 (highlighted in red). (*Id.*, Abstract, 7:17-20 (“...each handle server having a table which associates handles with pointers.”), 11:16-18 (“A ‘repository’ 1002 is a digital storage system into which digital objects 1004 may be placed and retained for possible subsequent retrieval.”), 37:31-33 (“Handle servers have the following characteristics: a handle server holds pointers associated with a subset of all handles....”).)

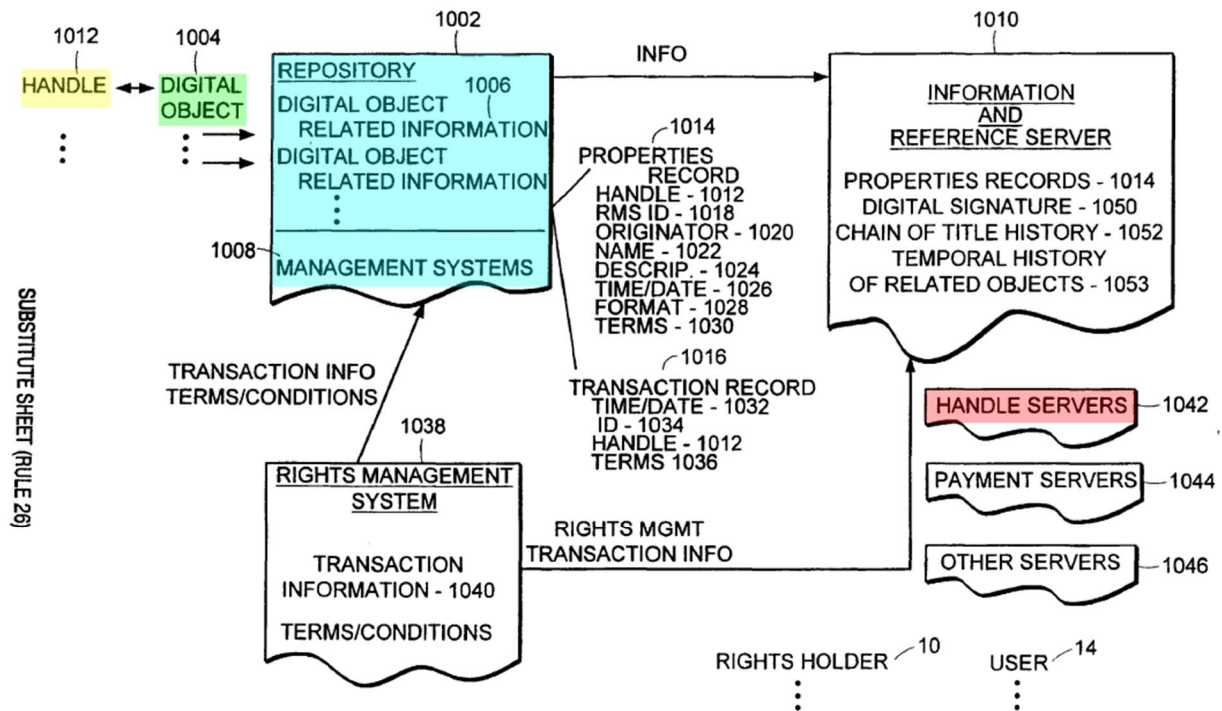


FIG. 1

(*Id.*, FIG. 1 (annotated).)

34. Kahn explains that a digital object may include conventional digital representations of works (e.g., books, papers, images, sounds, software) and each digital object is associated with a unique handle used to identify the digital object.

(*Id.*, 1:16-20 (“Thus digital objects may include conventional representations of works (books, papers, images, sounds, software), and more broadly any digital material which is capable of producing desired manifestations for a computer user.”), 21:3-5 (“Each digital object has a ‘handle’, a concise unique identifier for a digital object used for storage and retrieval operations and other repository

functions.”.) Kahn discloses that a digital object is stored in a repository in a distributed network, as “[digital] objects are accessed at the repository via the network,” where the network does not have “a central nodal point in the system (for reliability).” (*Id.*, Abstract (“The [digital] objects are stored at locations...accessible in the network. []. Other aspects include providing multiple servers [] each of which accepts identifiers of a subset of the digital objects and returns corresponding pointers to the locations of the digital objects in the network....”), 2:25-30 (“Making digital objects available on networks (e.g., Internet), gives rise to at least four specific activities of concern. The first is the ease of movement of digital objects already contained in a computer network environment....”), 3:15-19 (“[T]he [digital] objects are stored at locations accessible in the network”), 6:9-35 (“Copies of the digital objects are stored in a repository.... []. The objects are accessed at the repository via the network.... []. The handle is used to obtain a pointer to the network location of...the digital object.”), 11:16-18 (“A ‘repository’ 1002 is a digital storage system into which digital objects 1004 may be placed and retained for possible subsequent retrieval.”), 21:21-24 (“...services can be provided on a widely distributed basis by multiple similar servers at different locations on the network....”), 73:23-31 (“One or more pointers, once returned, identifies the network location of the one or more

repositories (if one is associated with the object)....This strategy assures a random distribution of handle server requests among many handle servers distributed on a network without a central nodal point in the system (for reliability).”) For example, Kahn discloses that digital objects are stored on the Internet. (*Id.*, 2:25-30 (“Making digital objects available on networks (e.g., Internet), gives rise to...activities of concern. The first is the ease of movement of digital objects already contained in a computer network environment allowing the creation of multiple copies in multiple machines in fractions of second.”).)

35. Kahn discloses further details explaining that the one or more pointers returned by a handle server indicate the location of a stored digital object in the distributed network. (*Id.*, Abstract (“[M]ultiple servers (1042) each of which accepts identifiers of a subset of the digital objects and returns corresponding pointers to the locations of the digital objects in the network....”), 3:19-22 (“[A] pointer indicating the location of the stored digital object is also stored in the network.”), 6:30-35 (“The handle is used to obtain a pointer to the network location of an accessible copy [of the digital object].”), 28:32-34 (“Object pointers contain location information for locating digital objects [].”).) For example, Figure 8 describes a client system retrieving one or more pointers for digital objects. (*Id.*, FIG. 8, 46:25-47:25 (“Given a handle, the following steps, shown in Figure 8, are

followed to obtain a set of pointers associated with the handle.”); *see also id.*,
72:32-74:32.)

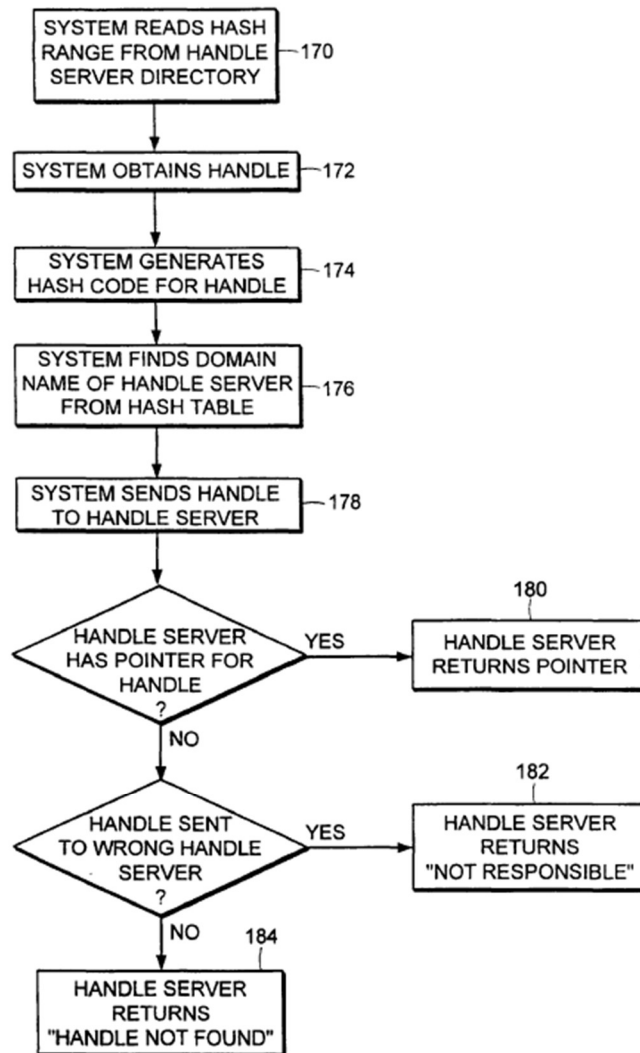


FIG. 8
SUBSTITUTE SHEET (RULE 26)

(*Id.*, FIG. 8.)

B. Vingralek

36. Vingralek discloses a “distributed file organization system” where a

file is “organized into buckets that are spread across multiple servers, where a server may hold multiple buckets.” (Ex. 1007, 4.) Vingralek further explains that clients can modify the file’s distribution. (*Id.*, 9 (“Consider a file consisting of keyed records that is spread across a number of servers connected by a network. []). A server keeps records of the file primarily on disk and we assume that servers are dedicated to managing a distributed file. We assume that there is a distributed group of clients that issue requests to insert, delete, and retrieve records for a given key *K* in the file. Consequently, the file grows or shrinks dynamically.”).) The approach to file distribution in Vingralek is hash-based as hashing is used to distribute keys across buckets, where keys are used to retrieve part of a particular file. (*Id.*, 5 (“The clients locate servers for insertion, update, deletion, and key-based retrieval through a hash function....”), 6 (“This paper presents a new approach for hash-based distributed files....”), 9 (“Thus, a hash file organization is the most appropriate in such an environment.”).)

37. Additionally, Vingralek teaches changes in the data/file distribution may “result [in] changes to the server-locating hash function.” (*Id.*, 5.) In that scenario, the client is notified “in order to update the client’s stale information on the hash function.” (*Id.*; *see also id.*, 5 (“To limit the communication overhead, clients learn about bucket splits and the resulting changes to the server-locating

hash function only when a client request is directed to the ‘wrong’ server.”).) For example, each client and server may have stale information on a parameter called the “file level,” which is the highest server level among all servers. (*Id.*, 10.)

Thus, the hash function, which depends on the “file level” parameter, may also be stale if there has been a change to the servers. (*Id.*)

38. Moreover, instead of updating a client’s hash function immediately after a change to the data/file distribution has occurred, Vingralek discloses such an update is not provided to the client until “a client request is directed to the ‘wrong’ server” in order to limit or reduce the “communication overhead.” (*Id.*, 5 (“To limit the communication overhead, clients learn about bucket splits and the resulting changes to the server-locating hash function only when a client request is directed to the ‘wrong’ server. In such a case, the server forwards the client’s request to the proper server, and notifies the client in order to update the client’s stale information on the hash function. The most salient feature of this approach is that its communication overhead is largely independent of the number of servers and clients in the system; thus, it is considered a scalable approach.”); *see also id.*, 6 (providing “updates...to client...in a deferred manner”).) Indeed, Vingralek explains that this approach would have decoupled the communication overhead from the number of servers/clients in the system and improved scalability. (*Id.*, 5

(“The most salient feature of this approach is that its communication overhead is largely independent of the number of servers and clients in the system; thus, it is considered a scalable approach.”.)

VIII. CLAIM CONSTRUCTION

39. I have been asked to consider and apply in my analysis certain constructions of claim terms, as discussed below. For the remaining claim terms in the challenged claims, I have been asked to give and apply in my analysis their plain and ordinary meaning, as would have been understood by a person of ordinary skill in the art, at the time of the alleged invention (as I mentioned above is the mid-1990s, up to and including July 8, 1998) having taken into consideration the language of the claims, the specification, and the prosecution history of the '170 patent. I have also been asked to consider and apply in my analysis the interpretations of certain terms that I understand are from a prior IPR proceeding and a district court proceeding, neither of which involves Petitioner. I have applied these understandings in my analysis and in forming my opinions where appropriate

in this Declaration.²

A. “identifier”

40. I have been asked to interpret the claimed “identifier” and “identifier string” recited in claims of the ’170 patent to mean:

“a unique encoding that identifies an individual entity, and with which zero or more data location specifiers are associated in a location server.”

B. “location information”

41. I have been asked to interpret the claimed “location information” recited in claims of the ’170 patent to mean:

“information pertaining to one or more locations of data and/or the identities of one or more location servers.”

C. “location server”

42. I have been asked to interpret the claimed “location server” recited in claims of the ’170 patent to mean:

“a network-attached component that maintains a set of

² As I explain below, I have been asked to interpret limitations in claim elements in particular ways, and I applied those interpretations in my analysis. (*See* below in Section IX.)

identifier/location mappings that are modified or returned in response to location request messages from clients.”

D. “client”

43. I have been asked to interpret the claimed “client” recited in claims of the ’170 patent to mean:

“a network-attached component (which may be software or hardware) that initiates update or lookup of identifier/location mappings from a location server with location request messages.”

E. “the portion of the data location information included in a corresponding one of the data location servers is based on a hash function used to organize the data location information across the plurality of data location servers, and each one of the data location servers is configured to determine the at least one of the plurality of data location servers based on the hash function applied to the identifier string”³

44. I have been asked to interpret the claimed “the portion of the data location information included in a corresponding one of the data location servers is *based on a hash function used to organize the data location information across the plurality of data location servers*, and each one of the data location servers is

³ I apply only the bolded and italicized part of this interpretation, as the rest recites the exact claim language.

configured to determine the at least one of the plurality of data location servers based on the hash function applied to the identifier string” recited in a claim of the ’170 patent to mean:

“the portion of the data location information included in a corresponding one of the data location servers is *based on a hash function that maps identifier strings to one or more of the data location servers*, and each one of the data location servers is configured to determine the at least one of the plurality of data location servers based on the hash function applied to the identifier string.”

F. “data pertaining to the entity”

45. I have been asked to interpret the claimed “data [in a network] pertaining to the entity” recited in claims of the ’170 patent to mean:

“data pertaining to a person or thing (real, digital, or abstract) distinct from that data.”

G. “plurality of location servers”

46. I have been asked to interpret the claimed “plurality of location servers” and related term “plurality of data location servers” recited in claims of the ’170 patent to mean:

“location servers in non-hierarchical structures.”

47. As I have indicated, I applied each of the above interpretations for the terms identified above in my analysis where appropriate. These terms and their corresponding interpretations are listed below. I refer to each interpretation as “Interpretation from Other Proceedings.”

'170 Patent Terms	Interpretation from Other Proceedings
“identifier”; “identifier string”	“a unique encoding that identifies an individual entity, and with which zero or more data location specifiers are associated in a location server”
“location information”	“information pertaining to one or more locations of data and/or the identities of one or more location servers”
“location server”	“a network-attached component that maintains a set of identifier/location mappings that are modified or returned in response to location request messages from clients”
“client”	“a network-attached component (which may be software or hardware) that initiates update or

'170 Patent Terms	Interpretation from Other Proceedings
	lookup of identifier/location mappings from a location server with location request messages”
“the portion of the data location information included in a corresponding one of the data location servers is <i>based on a hash function used to organize the data location information across the plurality of data location servers</i>, and each one of the data location servers is configured to determine the at least one of the plurality of data location servers based on the hash function applied to the identifier string”	“the portion of the data location information included in a corresponding one of the data location servers is <i>based on a hash function that maps identifier strings to one or more of the data location servers</i>, and each one of the data location servers is configured to determine the at least one of the plurality of data location servers based on the hash function applied to the identifier string”

'170 Patent Terms	Interpretation from Other Proceedings
“data pertaining to the entity”	“data pertaining to a person or thing (real, digital, or abstract) distinct from that data”
“plurality of location servers”; “plurality of data location servers”	“location servers in non-hierarchical structures”

IX. THE PRIOR ART DISCLOSES AND/OR SUGGESTS ALL RECITED FEATURES OF CLAIMS 1-17 OF THE '170 PATENT

48. In my opinion, the prior art discussed below discloses and/or suggests the limitations of claims 1-17.⁴

A. Kahn in View of Vingralek Discloses and/or Suggests Claims 1-17

1. Claim 1

i) [1.pre] “A system for managing data stored in a distributed network, the system comprising:”

49. I understand that this limitation is a preamble of the claim and I have been asked to assume that the preamble of claim 1 is limiting. Under such an

⁴ In this section (Section IX), I refer to exhibits other than those identified prior art reference(s) that disclose and/or suggest the claimed limitations. Such exhibits reflect the state of the art known to a person of ordinary skill in the art at the time of the alleged invention.

assumption, Kahn discloses the limitations therein. To begin, Kahn discloses a system for managing digital objects stored in one or more repositories in a network, where the digital objects are located by one or more pointers associated with a handle. (Ex. 1005, Abstract (“A method of managing digital objects...in a network is presented. The objects are stored at locations...accessible in the network....Other aspects include providing multiple servers...each of which accepts identifiers of a subset of digital objects and returns corresponding pointers to the locations of the digital objects in the network.”), 1:10-12 (“By a ‘digital object’ we broadly mean any set of sequences of bits or digits and an associated unique identifier which we call a ‘handle.’”), 6:9-10 (“Copies of the digital objects are stored in a repository.”), 6:24 (“There may be multiple repositories.”).) Multiple handle servers in the network each may accept handles of digital objects and return corresponding pointers that indicate the digital objects’ locations in the network. (*Id.*, Abstract, 6:28-35 (“A service, accessible on the network, is provided for locating the handle associated with a digital object. The handle is used to obtain a pointer to the network location of...the digital object.”), 7:2-20 (“A **handle server, accessible on the network**, provides the pointer in response to the presentation of a handle. **Multiple servers provide the service, each serving a portion of the handle space**....Hash values are computed on the handles and the

hash values are distributed among **multiple handle servers**, each handle server having a table which associates handles with pointers.”), 21:6-24 (“**[M]ultiple servers...provide...handle-to-pointer translation services** (called ‘handle servers’)...It should also be clear that services can be provided on a widely distributed basis by **multiple similar servers at different locations on the network**, or by single centralized servers.”).) For example, Kahn discloses with reference to Figure 1 an exemplary system that includes a repository 1002 (blue) for storing digital objects 1004 (green) in the network, where the digital objects are identified by their handle 1012 (yellow) and associated pointers. (*Id.*, Abstract (“A method of managing digital objects (1004) in a network is presented. The objects are stored at [repository] 1002....Other aspects include providing multiple servers (1042) each of which accepts [handles (1012)] of a subset of digital objects and returns corresponding pointers to the locations of the digital objects in the network.”).) Handle-pointer mappings are stored in tables within handle servers 1042 (red). (*Id.*, Abstract, 7:19-20 (“...each handle server having a table which associates handles with pointers.”), 11:16-18 (“A ‘repository’ 1002 is a digital storage system into which digital objects 1004 may be placed and retained for possible subsequent retrieval.”), 37:31-33 (“Handle servers have the following characteristics: a handle server holds pointers associated with a subset of all

handles....”).)

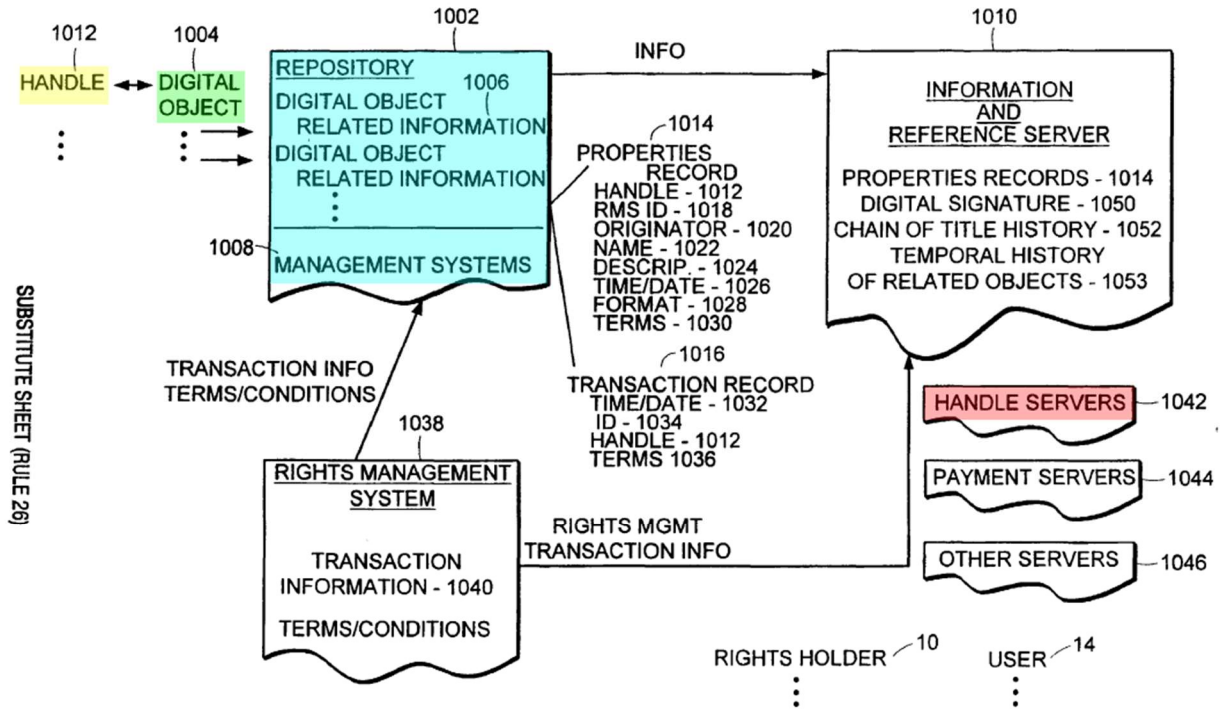


FIG. 1

(*Id.*, FIG. 1 (annotated).)

50. Kahn discloses that digital objects (including “data”) are “stored in a distributed network.” A digital object is represented by a sequence of bits or digits and may include conventional digital representations of real-world objects (e.g., books, images, sounds, software). (*Id.*, 1:10-12 (“By a ‘digital object’ we broadly mean **any set of sequences of bits or digits** and an associated unique identifier which we call a “handle.”), 1:16-20 (“Thus digital objects may include conventional digital representations of works (books, papers, images, sounds,

software), and more broadly any digital material which is capable of producing desired manifestations for a computer user.”.) A person of ordinary skill in the art would have understood that sequences of bits/digits associated with each digital object are “data,” as claimed. Indeed, Kahn discloses that “a digital object could include programs and data” which enable delivery over a network or reproduction on a computer screen. (*Id.*, 1:20-25.) Kahn discloses that a digital object is stored in a repository in a distributed network, such as the Internet, where the network does not have a central nodal point.⁵ (*Id.*, Abstract (“**The [digital] objects are stored at locations...accessible in the network.**”), 2:25-30 (“digital objects available on networks (e.g., Internet)”), 6:9-35 (“Copies of the digital objects are stored in a repository....The objects are accessed at the repository via the network....The handle is used to obtain a pointer to the **network location of...the digital object.**”), 11:16-18 (“A ‘repository’...is a digital storage system into which digital objects...may be placed and retained for possible subsequent retrieval.”), 21:21-24 (“services can be **provided on a widely distributed basis** by multiple

⁵ Kahn’s distributed network (e.g., Internet) is consistent with the ’170 patent’s disclosure associated with a distributed network. (Ex. 1001, 1:58-67, 4:50-57 (describing a network distributed across Internet machines).)

similar servers at different locations on the network”), 73:23-31 (“**One or more pointers, once returned, identifies the network location of the one or more repositories** (if one is associated with the object)....This strategy assures a random distribution of handle server requests among many handle servers distributed on a **network without a central nodal point in the system** (for reliability).”).)

51. Moreover, Kahn discloses that its communication system is for “managing data stored in a distributed network.” For example, Kahn discloses managing digital objects (including “data”) in a distributed network. (*Id.*, Abstract (“A method of managing digital objects (1004) in a network is presented. The objects are stored at [repository] 1002.... Other aspects include providing multiple servers (1042) each of which accepts identifiers of a subset of digital objects and returns corresponding pointers to the locations of the digital objects in the network.”).) For instance, as further described for limitation [1.b], Kahn discloses with reference to Figure 8 (below) a client system retrieving one or more pointers for digital objects. (*Id.*, FIG. 8, 46:26-28 (“Given a handle, **the following steps, shown in Figure 8, are followed to obtain a set of pointers** associated with the handle.”), 46:29-47:25; *see also id.*, 72:32-74:32; Section IX.A.1.iii.)

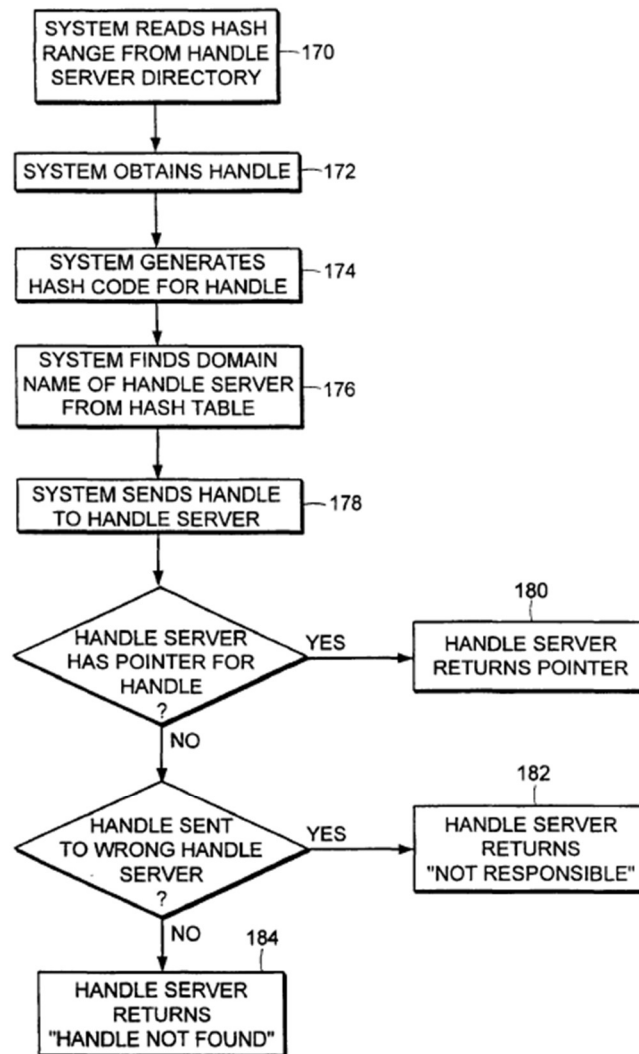


FIG. 8
SUBSTITUTE SHEET (RULE 26)

(Ex. 1005, FIG. 8.) Accordingly, Kahn discloses this limitation.

- ii) [1.a] “a data repository configured to store a data entity, wherein an identifier string identifies the data entity; and”

52. Kahn discloses this limitation. **First**, Kahn discloses “a data repository configured to store a data entity.” As discussed for limitation [1.pre],

Kahn discloses that digital objects (“data entity”) are stored in one or more individual repositories (together a “data repository”) in a network. (Ex. 1005, Abstract (“The [digital] objects are stored at [repositories] accessible in the network...”), 6:9-7:2 (“Copies of the digital objects are stored in a repository....The objects are accessed at the repository via the network....The handle is used to obtain a pointer to the **network location of...the digital object.**”), 11:16-18 (“A ‘repository’ ...is a digital storage system into which digital objects...may be placed and retained for possible subsequent retrieval.”), 21:21-24 (“...services can be **provided on a widely distributed basis** by multiple similar servers at different locations on the network....”), 73:23-31 (“**One or more pointers, once returned, identifies the network location of the one or more repositories** (if one is associated with the object)....This strategy assures a random distribution of handle server requests among many handle servers distributed on a **network without a central nodal point in the system** (for reliability).”), FIG. 1; *see* Section IX.A.1.i.) The sequences of bits/digits (“data”) pertaining to a particular object constitute the claimed “data entity.” (*See* Section IX.A.1.i.)

53. **Second**, Kahn discloses “wherein an identifier string identifies the data entity.” For example, Kahn discloses that each digital object (“data entity”) is associated with a unique handle (“identifier string”) identifying that digital object,

where the handle comprises a unique character string. (Ex. 1005, 7:4-5 (“The handles comprise unique character strings associated with the servers which generated them.”), 21:3-5 (“Each digital object has a **‘handle’**, a **concise unique identifier for a digital object** used for storage and retrieval operations and other repository functions.”); *see also id.*, 6:27-28 (“**The handle for a digital object is unique** both across the network and over time.”), 9:23-26 (“**The handles used to uniquely identify digital objects** are designed to be extensible and expandable to accommodate virtually any number objects over many years.”), 11:8-12 (“Formally, a digital object is an instance of an abstract data type that has two components, data and key-metadata....The key-metadata includes a **handle, i.e., an identifier globally unique to the digital object....**”).)

54. Kahn’s handle is an “identification string” under the Interpretation from Other Proceedings for related term “identifier,” i.e., “a unique encoding that identifies an individual entity, and with which zero or more data location specifiers are associated in a location server.” (*See* above Section VIII.) A person of ordinary skill in the art would have understood that a handle uniquely identifies a real-world object (i.e., an individual entity) because a handle uniquely identifies a digital object (i.e., a data entity), and as discussed for limitation [1.pre], the digital object is a digital representation of a real-world object/work. (Section IX.A.1.i.)

A person of ordinary skill in the art would have further understood that the handle is a unique encoding that identifies the object because the object and its digital representation (i.e., digital object) are uniquely represented and encoded in the system by the characters that comprise the handle. (Ex. 1005, 33:3-14 (“Handles should be globally unique across the network and over time; should be essentially permanent, since rights on an object may last many years; should not have any location information encoded in the identifier’s namespace, since an object may be located at multiple and changing locations over time....”), 33:22-27 (“The authorization and rules for creating a handle are determined on a country-by-country basis. In one scheme, as seen in Figure 5, handles are printable strings...having a country code...appended to a variable length string defined on a per country basis....”); Section IX.A.1.i, *see also* below Section IX.A.1.iii.) As further discussed for limitation [1.b], the handle (“identifier string”) is associated with zero or more pointers (i.e., zero or more data location specifiers, and the claimed “data location information”) in a handle server (i.e., a location server, as interpreted in the Other Proceedings). (*See* below Section IX.A.1.iii (addressing claimed “data location information” and “data location server” and Interpretation from Other Proceedings for a “location server”).)

- iii) **[1.b] “a data location server network comprising a plurality of data location servers, wherein data location information for a plurality of data entities is stored in the data location server network, at least one of the plurality of data location servers includes location information associated with the identifier string, each one of the plurality of data location servers comprises a processor and a portion of the data location information, the portion of the data location information included in a corresponding one of the data location servers is based on a hash function used to organize the data location information across the plurality of data location servers, and each one of the data location servers is configured to determine the at least one of the plurality of data location servers based on the hash function applied to the identifier string.”**

55. Kahn in view of Vingralek discloses and/or suggests this limitation.

First, Kahn discloses “a data location server network comprising a plurality of data location servers.” As discussed for limitation [1.pre], Kahn discloses a network containing multiple handle servers (“a data location server network comprising a plurality of data location servers”). (See above Section IX.A.1.i; Ex. 1005, 7:5-9 (“**A handle server, accessible on the network**, provides the pointer in response to the presentation of a handle. **Multiple servers provide the service, each serving a portion of the handle space**....Hash values are computed on the handles and the hash values are distributed among **multiple handle servers**, each handle server having a table which associates handles with pointers.”), 21:6-24 (“**[M]ultiple**

servers...provide...handle-to-pointer translation services (called ‘handle servers’). It should also be clear that services can be provided on a widely distributed basis by **multiple similar servers at different locations on the network**, or by single centralized servers.”.)

56. Kahn also discloses the claimed “data location server[]” under the Interpretation from Other Proceedings of “location server,” i.e., “a network-attached component that maintains a set of identifier/location mappings that are modified or returned in response to location request messages from clients.” (*See* above Section VIII; *see also* Ex. 1001, 4:17-20.) In particular, Kahn discloses that each handle server (“location server”) is a component attached to a distributed network (i.e., a network-attached component). (Ex. 1005, 7:5-7 (“**A handle server, accessible on the network**, provides the pointer in response to the presentation of a handle.”), 21:21-24 (“**services can be provided on a widely distributed basis by multiple similar servers at different locations on the network**”), 73:27-31 (“**many handle servers [are] distributed on a network** without a central nodal point in the system (for reliability)”).) Each handle server maintains mappings between handles (i.e., identifiers) and pointers (i.e., locations). (*See* Section IX.A.1.i; Ex. 1005, 7:17-20 (“...multiple handle servers, each handle server having a table which associates handles with pointers.”); *see also id.*, 37:31-

33 (“Handle servers have the following characteristics: a handle server holds pointers associated with a subset of all handles...”), 21:6-10 (“[M]ultiple servers...provide...handle-to-pointer translation services (called ‘handle servers’)....”).⁶ Further, Kahn discloses that the handle server returns identifier/location mappings in response to a request from the client for location information associated with a handle (i.e., in response to location request messages from a client). For example, as discussed further below, pointers (i.e., locations) are returned to the client if the handle server determines it contains the requested location information associated with the handle in the request. Given that the returned pointers are associated with the handle in the location information request, a person of ordinary skill in the art would have understood the returned pointers correspond to mappings of the handle to the pointers, consistent with the ’170 patent’s disclosure. (Ex. 1001, 2:38-42, 2:61-65, 6:67-7:4, 17:3-6.) Moreover, Kahn discloses that the full handle record (i.e., including the handle itself and all corresponding data) may be returned in response to a request from the client. (Ex.

⁶ Kahn’s pointer discloses the “location” (or similar terms) as described in the ’170 patent (Ex. 1001, 4:12-14) given that pointers are associated with certain handles and both are stored in a handle server (Ex. 1005, 7:17-20, 21:6-10, 37:31-33.)

1005, 42:1-11 (“We now describe how a client...resolves any handle....[T]he **client request may ask for all the data in a handle record....**”), 44:16-21 (“To resolve a handle, a handle server receives as input a handle and returns some or all of the fields of typed data in the corresponding handle record. **The client can request that all data fields in the handle record be returned....**”).) Thus, a person of ordinary skill in the art would have understood that if pointers are returned to the client system from the handle server, the handle itself may also be returned in response to the location request message from the client. Kahn also discloses that identifier/location mappings may be updated/modified in response to a request from the client to add new location information (i.e., another type of location request message from the client). (*Id.*, 43:31-44:4, 45:22-46:24; Ex. 1001, 6:50-53.) Thus, Kahn discloses the claimed “data location server []” under the Interpretation from Other Proceedings for “location server.”

57. Further, Kahn discloses the claimed “plurality of data location servers” under the Interpretation from Other Proceedings for “plurality of location servers,” i.e., “location servers in non-hierarchical structures.” (*See* above Section VIII.) A person of ordinary skill in the art would have understood that the handle servers (each a “data location server”) are arranged in a flat, cluster topology, rather than a hierarchical or tree-like structure where servers can only

communicate with nodes immediately above or below it in the tree. Kahn discloses that the hash function is used to distribute handles and associated pointers among the handle servers. (Ex. 1005, 4:19-22, 7:8-9, 37:31-38:3, FIG. 7.) A person of ordinary skill in the art would have thus understood that because a hash function is used for distribution across handle servers, Kahn's handle servers are arranged in a non-hierarchical configuration.

58. **Second**, Kahn discloses “wherein data location information for a plurality of data entities is stored in the data location server network.” For example, Kahn discloses that handles and their associated pointers (“data location information”), which indicate the locations of digital objects (each including “data,” and together a “plurality of data entities”), are stored in multiple handle servers in the network (“data location server network”). In particular, Kahn explains that a handle uniquely identifies a digital object, and a pointer indicates the location of the digital object in the network. (Ex. 1005, 28:31-34 (“**A handle for an object may be associated with zero or more object pointers. Object pointers contain location information for locating digital objects.**”).) Kahn further discloses multiple handle servers and each handle server “holds pointers associated with a subset of all handles,” as a handle is used to obtain a pointer to the location of a digital object. (*Id.*, 37:31-33 (“Handle servers have the following

characteristics: a handle server holds pointers associated with a subset of all handles....”); *see also id.*, Abstract (“[M]ultiple servers...each of which accepts [handles] of a subset of the digital objects and returns corresponding pointers to the locations of the digital objects in the network....”), 3:21-22 (“[A] pointer indicating the location of the stored digital object is also stored in the network.”), 6:30-35 (“The handle is used to obtain a pointer to the network location of an accessible copy [of the digital object].”), 28:32-34 (“Object pointers contain location information for locating digital objects....”); *see* Section IX.A.1.i..) For the same reason discussed above, Kahn discloses the claimed “data location information” and “location information” under the Interpretation from Other Proceedings for “location information,” i.e., “information pertaining to one or more locations of data and/or the identities of one or more location servers.” (*See* above Section VIII.)

59. **Third**, Kahn discloses “at least one of the plurality of data location servers includes location information associated with the identifier string.” For example, Kahn discloses each handle server (“at least one of the plurality of data location servers”) includes one or more pointers (“location information”) associated with a handle (“associated with the identifier string”). In particular, Kahn discloses “...multiple handle servers, each handle server having a table

which associates handles with pointers.” (Ex. 1005, 7:17-20; *see also id.*, 21:8-10 (“[M]ultiple servers...provide...handle-to-pointer translation services (called ‘handle servers’)...”), 37:31-33 (“Handle servers have the following characteristics: a handle server holds pointers associated with a subset of all handles....”); *see also* Section IX.A.1.i.)

60. **Fourth**, Kahn discloses “each one of the plurality of data location servers comprises a processor and a portion of the data location information.” For example, Kahn discloses each handle server (“each one of the plurality of data location servers”) provides handle-to-pointer translation services, i.e., a service returning a pointer in response to an associated handle, where each handle server can be on one or more computers operating using software. (Ex. 1005, 21:6-15 (“**The overall system...comprises multiple servers which provide...handle-to-pointer translation services (called ‘handle servers’).**”), 26:28-31 (“[A]n example system...includes a digital object management system...which includes hardware and **software to create and store digital objects and manage rights to the objects.**”), 29:5-12 (“Handle servers...process handle query requests. If the handle which is the subject of the query is found by a handle server, the object pointers associated with the handle are returned to the requesting client. A handle server accepts a handle as input and returns a list of pointers associated with the

handle, where each pointer = {domain name of storage system (repository), domain name of RMS}.”), 42:2-11 (“**[E]ach handle server can be implemented as one or more server computers...**”), 44:6-7 (“A handle is sent to a handle server to locate network addresses of repositories containing that object.”), 65:14-69:14 (defining various software packages run on the client system).) As such, a person of ordinary skill in the art would have understood that each handle server (“each one of the plurality of data location servers”), which may be a computer, includes a “processor.” Further, Kahn discloses each handle server stores a portion of all pointers (“a portion of the data location information”) for locating digital objects. Each handle server only holds “pointers associated with a subset of all handles.” (*Id.*, 7:5-9 (“A handle server, accessible on the network, provides the pointer in response to presentation of a handle. **Multiple servers provide the service, each serving a portion of the handle space.**”), 37:31-38:3 (“Handle servers have the following characteristics: **a handle server holds pointers associated with a subset of all handles**; handles are assigned to handle servers based upon hash values computed on the handles; handle servers are assigned ranges of hash values; the set of all hash values is partitioned among the set of all handle servers. This leads to a highly efficient and reliable mechanism for locating objects and from handles.”), 46:22-24 (“[T]he handle server directory...holds a

table...which associates hash ranges...with domain names of handle servers....”).)

61. **Fifth**, Kahn discloses “the portion of the data location information included in a corresponding one of the data location servers is based on a hash function used to organize the data location information across the plurality of data location servers.” For example, Kahn discloses that the portion of pointers (“the portion of the data location information”) included in a corresponding handle server (“one of the data location servers”) is based on the application of a hash algorithm (“hash function”) used to partition/distribute the handles and associated pointers (“organize the data location information”) across the multiple handle servers (“across the plurality of data location servers”). In particular, Kahn explains that each handle server is assigned a portion of handles and associated pointers and stores the portioned handles as well as the associated pointers based on hash code/value ranges, which are the result of handles being hashed using the hash algorithm, as discussed further below with respect to Figure 7. (*Id.*, 37:31-38:3 (“Handle servers have the following characteristics: a handle server holds pointers associated with a subset of all handles; **handles are assigned to handle servers based upon hash values computed on the handles; handle servers are assigned ranges of hash values; the set of all hash values is partitioned among the set of all handle servers.** This leads to a highly efficient and reliable

mechanism for locating objects and from handles.”), 47:1-11 (“The system then generates the hash code for the handle using a predetermined hashing algorithm....”); *see also id.*, 29:23-26 (“The handle server directory provides a list of domain names of handle servers and **identifies the set of hash values of handles which each of the handle servers can map to pointers.**”).)

62. Kahn discloses that the portion of data location information is included in a corresponding data location server “based on a hash function used to organize the data location information across the plurality of data location servers” under the Interpretation from Other Proceedings, i.e., “based on a hash function that maps identifier strings to one or more of the data location servers.” (*See above Section VIII.*) Kahn discloses that its hash algorithm maps handles (“identifier string[s]”) to one or more handle servers (“data location servers”) because the hash values generated by applying the hash algorithm to the handle are used to partition/distribute the handles amongst multiple handle servers. (Ex. 1005, 29:23-26 (“The handle server directory provides a list of domain names of handle servers and **identifies the set of hash values of handles which each of the handle servers can map to pointers.**”), 37:31-38:3 (“Handle servers have the following characteristics: a handle server holds pointers associated with a subset of all handles; **handles are assigned to handle servers based upon hash values**”)

computed on the handles; handle servers are assigned ranges of hash values; the set of all hash values is partitioned among the set of all handle servers.

This leads to a highly efficient and reliable mechanism for locating objects and from handles.”), 47:1-11 (“The system then generates the hash code for the handle using a predetermined hashing algorithm....”).)

63. **Sixth**, Kahn discloses “determin[ing] the at least one of the plurality of data location servers based on the hash function applied to the identifier string.” For example, as shown in Figure 8 and described below, Kahn describes a process by which a client system retrieves one or more pointers for digital objects, which involves applying a hash function to a handle (“hash function applied to the identifier string”) to determine which handle server to send a request to (“determine the at least one of the plurality of data location servers”). (*Id.*, FIG. 8, 46:26-28 (“Given a handle, **the following steps, shown in Figure 8, are followed to obtain a set of pointers** associated with the handle.”), 46:29-47:25; *see also id.*, 72:32-74:32.)

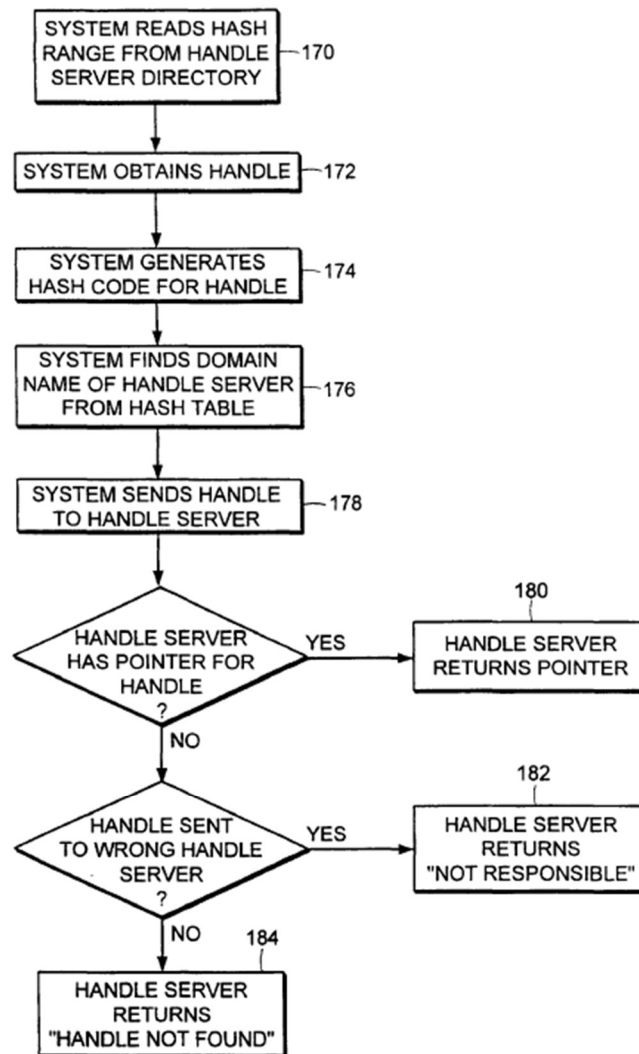


FIG. 8
SUBSTITUTE SHEET (RULE 26)

(*Id.*, FIG. 8.) In particular, as shown in Figure 8 step 170, a client system first, e.g., at initiation, downloads from a handle directory server a hash range/handle-server-domain-name table (also referred to as a hash code/handle server table, *see* Figure 7 below) that associates hash ranges with handle server domain names.

(*Id.*, 46:22-47:5, FIG. 7.) The table allows the client system to determine the

location of a handle server that contains a pointer for retrieving the digital object associated with the handle. (*Id.*, 29:18-26 (“The handle server directory...**allows users to find the correct handle server for processing a given handle....**The handle server directory...distributes handles to handle servers based upon a hash of handles. The handle server directory provides a list of domain names of handle servers and identifies the set of hash values of handles which each of the handle servers can map to pointers.”), 38:9-17 (“[T]he handle server system...includes handle servers...and handle directory servers...located at certain well known locations. **The directory servers will maintain a table of network addresses of handle servers (generally, each handle server will contain such a directory).** **This table will generally be downloaded by each participating site frequently enough to be ‘acceptably’ up-to-date at all times.**”), 46:22-47:5 (“In one example, the handle server directory...holds a table...which associates hash ranges...with domain names of handle servers...(Figure 7)....In a first step 170, a client system downloads the table that associates hash ranges with handle server domain names from the handle server directory for future use. The client also can omit this step if it has previously stored the table; frequent changes in the table may necessitate doing this every time....**The system** then generates the hash code for the handle using a predetermined hashing algorithm (step 174) and **consults**

the hash range/handle-server-domain-name table to determine the domain name of the appropriate handle server (step 176).”), 73:4-74:26 (“The system requesting the digital object obtains...the hash code/handle server table from the handle server directory.... This is done during the system’s initialization....Once the handle is obtained, the system that retrieves the object ‘hashes’ the object’s handle and uses this hash value to perform a table lookup in the **hash code/handle server table....The requesting system sends a request-for-pointer-information UDP packet...to the handle server. One or more pointers, one returned, identifies the network location of the one or more repositories (if one is associated with the object)....For each pointer returned by the handle server, the requesting system uses the pointer to attempt to obtain a copy of the digital object....”).)**

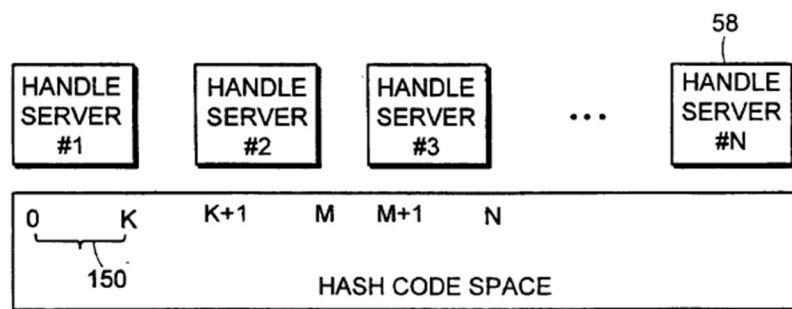


FIG. 7

(*Id.*, FIG. 7.) For instance, as shown in Figure 7 above, each handle server is assigned a range of hash codes/values 150. (*Id.*) The client generates a hash code

for a handle using a predetermined hashing algorithm and then uses the hash code/value as an index into the downloaded table, allowing the client to obtain the domain name or location of the handle server that potentially stores pointers for locating the digital object associated with the handle (“determine the at least one of the plurality of data location servers based on the hash function applied to the identifier string”). (*Id.*, 47:1-11 (“**The system** then generates the hash code for the handle using a predetermined hashing algorithm (step 174) and **consults the hash range/handle-server-domain-name table to determine the domain name of the appropriate handle server (step 176).** The system subsequently sends the handle to the handle server as part of a request pointer information UDP packet (step 178). The handle server then returns its response to the requesting system. **If the handle server found the pointers, a list of pointers is returned (step 180).**”), 73:17-26 (“Once the handle is obtained, the system that retrieves the object ‘hashes’ the object’s handle and **uses this hash value to perform a table lookup in the hash code/handle server table....The requesting system sends a request-for-pointer-information UDP packet...to the handle server. One or more pointers, once returned,** identified the network location of the one or more repositories (if one is associated with the object)...”).)

64. Returning to Figure 8, after downloading the hash range/handle-

server-domain-name table in step 170, the client in step 172 receives a handle from a user's request (by, e.g., a bibliographic search or scanning a barcode) for obtaining a certain digital object. (*Id.*, 46:34-47:1 (“In a later step 172, assume the system obtains a handle for which pointers are desired.”), 73:8-16 (“**A user (or more likely, his software agent) obtains a handle...for an object....**The handle may be obtained as part of a result of a bibliographic search or be provided by some other electronic means such as an electronic reference list in another object, or by scanning a barcoded sequence on paper. The system that is retrieving the digital object is referred to as the requesting [client] system....”).) Subsequently, the client system “generates the hash code for the handle using a predetermined hashing algorithm,” then uses the hash code to index into the hash range/handle-server-domain-name table “to determine the domain name of the appropriate handle server,” and sends the handle to the appropriate handle server in steps 174, 176, and 178 of Figure 8. (*Id.*, 47:1-11 (“**The system** then generates the hash code for the handle using a predetermined hashing algorithm (step 174) and **consults the hash range/handle-server-domain-name table to determine the domain name of the appropriate handle server (step 176).** The system subsequently sends the handle to the handle server as part of a request pointer information UDP packet (step 178). The handle server then returns its response to the requesting

system. **If the handle server found the pointers, a list of pointers is returned (step 180).**”), 73:17-26 (“Once the handle is obtained, the system that retrieves the object ‘hashes’ the object’s handle and **uses this hash value to perform a table lookup in the hash code/handle server table....The requesting system sends a request-for-pointer-information UDP packet...to the handle server. One or more pointers, once returned,** identified the network location of the one or more repositories (if one is associated with the object)....”).)

65. Next, the handle server verifies if it does or, in the event of a dynamic change to the table downloaded to the client, does not contain the handle and its associated pointers. (*Id.*, 47:10-11 (“**If the handle server found the pointers, a list of pointers is returned (step 180).**”), 73:32-74:12 (“**The handle server verifies that the handle falls within the set of handles for whose hash value it is responsible....If it is not in this set, due to some dynamic system change or error condition,** the handle server sends an invalid-handle-server-selected response UDP packet to the requestor....If the requesting system receives an invalid-handle-server-selected response UDP packet, it refreshes its hash code/handle server table [shown in Figure 7] from the handle server directory, and the requesting system repeats prior steps. **This will typically be needed only if the table has changed between the time the table was downloaded and the actual**

request was made.”).) If the handle is indeed located in the queried handle server, one or more pointers for locating the digital object stored in the distributed network are returned to the client system in step 180. (*Id.*, 29:8-10 (“A handle server accepts a handle as input and **returns a list of pointers** associated with the handle....”), 47:10-13 (“If the handle server found the pointers, **a list of pointers is returned (step 180)**. This could include pointers to use one or more repositories....”), 73:23-26 (“**One or more pointers, once returned**, identifies the network location of the one or more repositories (if one is associated with the object)....”), 74:21-26 (“An object may be stored in several repositories. Multiple pointers to these repositories may be returned to the requesting system. For **each pointer returned by the handle server**, the requesting system uses the pointer to attempt to obtain a copy of the digital object....”).) The client system then uses the retrieved pointers to obtain the digital object stored in a repository in the distributed network. (*Id.*, 74:21-29 (“An object may be stored in several repositories. Multiple pointers to these repositories may be returned to the requesting system. For each pointer returned by the handle server, the requesting system uses the pointer to attempt to obtain a copy of the digital object....If a copy is successfully obtained from one repository, the rest of the pointers will generally be ignored.”).)

66. I have been asked to consider a scenario where Kahn does not explicitly disclose “**each one of the data location servers is configured to** determine the at least one of the plurality of data location servers based on the hash function applied to the identifier string.” In that scenario, a person of ordinary skill in the art would have been motivated to include this feature in view of Vingralek.

67. Like Kahn, Vingralek relates to storing data in a distributed network. For example, Vingralek discloses a “distributed file organization system” where a file is “organized into buckets...spread across multiple servers” (Ex. 1007, 4) and clients can modify the file’s distribution by issuing requests to insert/delete/retrieve records for a given key *K* (akin to an identifier of a record) in the file, such that the file grows or shrinks dynamically (*id.*, 9 (“Consider a file consisting of keyed records that is spread across a number of servers connected by a network...A server keeps records of the file primarily on disk and we assume that servers are dedicated to managing a distributed file. We assume that there is a **distributed group of clients that issue requests to insert, delete, and retrieve records for a given key *K* in the file. Consequently, the file grows or shrinks dynamically.**”).). Vingralek offers “a new approach for hash-based distributed files that strives for...‘cost-conscious’ scalability...to minimize the number of servers...used for providing the necessary performance capacity.” (*Id.*, 6.) This

approach is closely-related to a distributed version of linear hashing, such as LH*, which “assigns hash buckets to servers and adds servers dynamically upon bucket splits.” (*Id.*, 5, 8.) In both LH* and Vingralek’s approach, clients locate servers for insertion/update/deletion, and perform “key-based retrieval through a hash function whose target domain is dynamically expanded as buckets are split and servers are added, based on the principle of linear hashing.” (*Id.*, 5.) Further, Vingralek discloses that its hash-based distributed file system is implemented using an address table, which a person of ordinary skill in the art would have understood is akin to a hash table, to look up the server that corresponds to a hashed key. (*Id.*, 4 (“Client requests are serviced by mapping keys onto buckets and looking up the corresponding server in an address table.”), 6 (“[W]e assume an indirection between bucket numbers and server numbers, so that multiple buckets can be assigned to the same server. This indirection is implemented by an *address table*, similar to the directory of the extendible hashing method...but stored in a more compact way.”).) Thus, Vingralek is similar to Kahn, which discloses a hash-based object distribution system and using a handle to locate/retrieve data. (See Section IX.A.1.i.). A person of ordinary skill in the art would have therefore been motivated to consider the teachings of Vingralek when implementing Kahn’s system/method.

68. Additionally, Vingralek teaches a server of its system can calculate the location of a correct server to forward a request to if the request was sent to a wrong server, i.e., a server that does not contain the requested information. As an initial matter, Vingralek discloses that each client and server has its own copy of the address table, but, for example, the client's table may have stale information if changes have been made to the servers since the table was last updated. (Ex. 1007, 10; *see also id.*, 11-13.) When a client seeks to retrieve records associated with a given key, it applies the hash function to the key, identifies the server where that key is located by using the hashed key as an index into the current copy of the address table, and then sends the key in a request to that first server. (*Id.*, 9-10.) The first server that receives the key also applies the hash function to the key and identifies in its own copy of the address table the server where the key resides (at least according to that first server's address table). (*Id.*, 10-11.) If the key was indeed sent to the correct server (i.e., if the identified server is the same as that first server), then that first server performs the requested operation. (*Id.*, 11.) But if the key was sent to the wrong server, the first server forwards the request to another server (i.e., the one identified by the above-described hash function application), and sends a message to the client to update the client's address table. (*Id.*; *see also id.*, 25 ("client requests and server responses" are "messages").)

69. The key is eventually received by the correct server and the request from the client is executed. (*Id.*, 11.) Via the address table update, “clients gradually bring their perception of the file up-to-date, so that the probability of wrongly addressed messages is significantly reduced.” (*Id.*) Vingralek explains that if a server undergoes changes (e.g., if information is moved to another server), the server receives a new address table. (*Id.*, 12-13.) Consistent with the above, a person of ordinary skill in the art would have understood that a server would have up-to-date information with respect to its own keys/information, such that the process of updating the client’s address table each time a request is forwarded to the next server (until the request reaches a correct server) would have gradually brought the client’s perception of the address table up-to-date. Thus, Vingralek discloses that if a server does not have the requested information, each server (“each one of the data location servers”) is configured to apply a hash function to the key (“based on the hash function applied to the identifier string”) to determine another server to forward the request to (“to determine the at least one of the plurality of data location servers”).

70. A person of ordinary skill in the art would have been motivated to include features similar to those disclosed by Vingralek when implementing Kahn. Kahn teaches that every time a handle server does not contain the requested

handle, a message is returned back to the client, triggering the client to download an updated hash table to determine/calculate the correct handle server, and the client subsequently redirects the request to the correct server. (Ex. 1005, 46:25-47:25, 73:1-74:32.) Vingralek's teaching limits communication overhead by having a server (e.g., the "wrong" server) directly forward a request to the correct server, thus limiting both additional communications to the client and additional downloads by the client. (Ex. 1007, 5, 8.) Vingralek's feature would have been beneficial as it would have reduced the frequency of communications to a client and/or the use of the client's computational resources, particularly in situations where the client has limited communication bandwidth and/or limited computational resources (e.g., resources required for calculating the correct handle server). (*Id.*, 5, 7, 8.) Such a modification would have been consistent with Kahn's goals of an "efficient and reliable implementation" in an expandable/scalable system where the server-client communications may increase (Ex. 1005, 9:23-27 ("handles used to uniquely identify digital objects are designed to be extensible and expandable to accommodate virtually any number of objects"))).

71. Moreover, while Kahn discloses that, when receiving a request, "[t]he handle server verifies that the handle falls within the set of handles for whose hash

values it is responsible,” it does not explicitly disclose the specifics of its verification method. (*Id.*, 73:32-34.) Thus, a person of ordinary skill in the art would have been motivated to implement a verification method similar to that of Vingralek, i.e., a location server applying the hash function to determine which location server contains the requested information, as discussed above, especially because Kahn already discloses that the handle server stores a copy of the hash table. (*Id.*, 38:9-17, 46:29-32.) Thus, a person of ordinary skill in the art would have been motivated to include features similar to those disclosed by Vingralek, e.g., a server determining/calculating the location of a correct server by applying the hash function to the handle and use the result as an index into the hash table for forwarding the request to the correct server, when implementing Kahn.

72. A person of ordinary skill in the art would have also had a reasonable expectation of success combining the teachings of Kahn and Vingralek. As discussed above, both references are in the same field of art. (*See above.*) Further, such an implementation would have involved modifying the Kahn system in a manner that Kahn suggests is feasible—having the handle servers apply a hash function—as Kahn, like Vingralek, discloses that both client and handle server store copies of the hash table. (Ex. 1005, 38:9-17 (“[T]he handle server system...includes handle servers...and handle directory servers...located at certain

well known locations. The directory servers will maintain a table of network addresses of handle servers (generally, **each handle server will contain such a directory**). **This table will generally be downloaded by each participating site** frequently enough to be ‘acceptably’ up-to-date at all times.”), 46:29-32 (“In a first step 170, **a client system downloads the table that associates hash ranges with handle server domain names from the handle server directory** for future use.”).) Additionally, the implementation would have been consistent with Kahn’s teaching where a handle server sends a request to another handle server to “enhance the reliability and effectiveness of the system.” (*Id.*, 38:5-8.) Moreover, the combination of Kahn and Vingralek would have involved applications of known technologies (e.g., known distributed data storage systems (Kahn)) according to known methods (e.g., known methods where a server calculates the location of a correct server based on applying a hash function to an identifier for forwarding a request (Vingralek)) to yield the predictable result of a distributed system where a server calculates a correct server’s location for forwarding a request in a manner that improves the efficiency of the system, as discussed above. Accordingly, Kahn in view of Vingralek discloses and/or suggests this limitation.

2. Claim 2: “The system of claim 1, wherein the location information comprises any portion of a hash table generated with the hash function.”

73. Kahn discloses this limitation. As discussed for limitation [1.b], Kahn discloses that a hash algorithm (“hash function”) is applied to a handle, with the output, i.e., a hash code/value, used to index into the hash range/handle-server-domain-name table (“hash table”). (Ex. 1005, 47:1-5 (“The system then generates the hash code for the handle using a predetermined hashing algorithm (step 174) and consults the hash range/handle-server-domain-name table to determine the domain name of the appropriate handle server (step 164).”); *see also* Section IX.A.1.iii.) Kahn further discloses that all handles are partitioned amongst multiple handle servers, where the handles are partitioned/distributed based on the hash algorithm and hash range/handle-server-domain-name table, such that each handle server is assigned a portion of the handle space. (Ex. 1005, 7:5-9 (“A handle server, accessible on the network, provides the pointer in response to presentation of a handle. **Multiple servers provide the service, each serving a portion of the handle space.**”), 37:31-38:3 (“Handle servers have the following characteristics: a handle server holds pointers associated with a subset of all handles; **handles are assigned to handle servers based upon hash values computed on the handles; handle servers are assigned ranges of hash values;**

the set of all hash values is partitioned among the set of all handle servers.

This leads to a highly efficient and reliable mechanism for locating objects and from handles.”), 46:22-24 (“[T]he handle server directory 59 holds a table 149 which associates hash ranges 150 with domain names of handle servers 58 (Figure 7).”); *see also* Section IX.A.1.iii.) Since handles are stored with their associated pointers (“location information”) on handle servers, the associated pointers (“location information”) are likewise partitioned/distributed over multiple handle servers according to the hash range/handle-server-domain-name table, such that the pointers on a given handle server represent only a portion of the full set of pointers. A person of ordinary skill in the art would have therefore understood the pointers (“location information”) on a given handle server comprise a portion of the full hash range/handle-server-domain-name table (“comprises any portion of a hash table”) because the handle server includes pointers associated with a portion of the handles in the hash range/handle-server-domain-name table. A person of ordinary skill in the art would have also understood the pointers on a given handle server represent a portion of the hash range/handle-server-domain-name table generated by the hash algorithm (“generated with the hash function”) because the hash algorithm controls which handles are assigned to each handle server, and thus controls the associated pointers (“location information”) stored on each handle

server.

3. **Claim 3: “The system of claim 1, wherein the data repository comprises a plurality of data repository servers and wherein each of the plurality of data location servers is configured, in response to a request to store data in the data repository, to execute the hash function and store the data on one of the plurality of data repository servers based on a result of the hash function.”**

74. The Kahn-Vingralek combination discloses and/or suggests this limitation. To begin, Kahn discloses “wherein the data repository comprises a plurality of data repository servers.” For example, Kahn discloses a repository system (“data repository”) which may be comprised of multiple individual repositories. (Ex. 1005, 6:22-24 (“Owners of rights in digital objects may deposit copies of the digital objects in the repository, via the network. There may be multiple repositories.”).) Kahn discloses that each of these repositories stores information, performs services, and contains management systems associated with certain tasks, such as those associated with a repository access protocol for adding and accessing digital objects. (*Id.*, 11:16-27 (“The repository has mechanisms for adding new digital objects to its collection (depositing) and for making them available (accessing), using, at a minimum, a so-called repository access protocol. The repository may contain other related **information, services and management systems.**”), 13:2-6 (“**[A] repository may offer a service** in which it deposits a

digital object and registers the handle simultaneously...”), 21:3-5 (“Each digital object has a ‘handle’, a concise unique identifier for a digital object used for storage and retrieval operations and other repository functions.”), 22:11-23 (“A ‘simple repository access protocol’ (RAP) is supported by each repository and discussed below. The RAP may be merely a subset of a larger interface protocol used by repositories, provided that the functions or operation of the RAP not be affected by any implemented supersets of the protocol. In particular,...the RAP...allows for accessing a stored digital object or its metadata by specifying its handle, a service request type, and additional parameters. If this request is complied with, the output of the service request is termed a ‘dissemination’.... A dissemination is the result of an access service request, along with additional data affixed to it.”). Kahn further discloses that each of the repositories may store digital object properties records containing digital object metadata and transaction records containing entries regarding digital object transactions/requests. (*Id.*, 14:28-16:8.) Thus, a person of ordinary skill in the art would have understood that each of the multiple repositories is a server (“a plurality of data repository servers”).

75. The Kahn-Vingralek combination also discloses and/or suggests “wherein each of the plurality of data location servers is configured, in response to

a request to store data in the data repository, to execute the hash function and store the data on one of the plurality of data repository servers based on a result of the hash function.” Kahn discloses commands/requests to modify existing digital objects, such that new information is associated with an already existing digital object. (Ex. 1005, 19:26-31 (“[N]etwork users...may wish to obtain, use, **modify**, transmit, perform, and enhance **selected digital objects**....”), 24:25-27 (“A DEPOSIT_DO command may...be used to directly **modify an existing mutable digital object**.”).) A person of ordinary skill in the art would have understood that such a request is a “request to store data in the data repository” which results in “stor[ing] the data on the plurality of data repository servers” because digital objects are stored in the data repository (and thus on data repository servers) and are comprised of sequences of bits/digits (“data”), and thus associating new information with an already existing digital object would have involved storing more sequences of bits/digits (“data”) with the digital object on the data repository servers.

76. As discussed for limitation [1.b], in the Kahn-Vingralek combination, each handle server (“each of the plurality of data location servers”) is configured to execute a hash function to determine the handle server that stores pointers associated with the handle. (See Section IX.A.1.iii.) A person of ordinary skill in

the art would have been motivated to configure each handle server to also execute the hash function “in response to a request to store data in the data repository.” A person of ordinary skill in the art would have understood that when modifying an existing digital object, it would have been advantageous to store the “data” corresponding to the digital object (and thus to the same handle) in similar locations (e.g., on the same data repository servers), so as to reduce processing requirements associated with the retrieval of a digital object. This is particularly true because, while Kahn discloses associating new information with an already existing digital object, Kahn does not expressly disclose the specifics regarding such a process. Thus, a person of ordinary skill in the art would have been motivated to implement the Kahn-Vingralek system such that, in response to a request to store additional “data” associated with a handle on the data repository servers (“in response to a request to store data in the data repository”), the system retrieves the pointers associated with the digital object’s handle to identify which repository servers should store the new data. Consistent with the discussion for limitation [1.b], each handle server (“each of the plurality of data location servers”) is configured to execute the hash function (“to execute the hash function”) to determine the handle server that stores pointers associated with the handle. (*See* Section IX.A.1.iii.) The system would then store the new “data” associated with

an already existing digital object on one of the data repository servers (“one of the plurality of data repository servers”) based on the pointers associated with the handle at the particular handle server identified by the result of the hash function (“based on a result of the hash function”). (*Id.*)

77. A person of ordinary skill in the art would have also had a reasonable expectation of success in implementing this modification because it would have involved modifying the Kahn-Vingralek system in a manner that Kahn suggests is feasible—implementing a method of modifying an existing digital object. (Ex. 1005, 19:26-31, 24:25-27.) Such a modification would have involved applications of known technologies (e.g., known systems for storing and modifying digital objects (Kahn)) according to known methods (e.g., a server executing a hash function to retrieve location information (Vingralek)) to yield the predictable result of a system that has a handle server execute a hash function as part of the process to store data on a data repository server, as discussed above. Accordingly, the Kahn-Vingralek combination discloses and/or suggests this limitation.

4. **Claim 4: “The system of claim 1, wherein each one of the data location servers is configured to return all location information associated with a respective one of the data entities in response to a request for any location information associated with the respective one of the data entities.”**

78. Kahn discloses this limitation. Kahn discloses “a request for any

location information associated with the respective one of the data entities.” For example, Kahn discloses the client transmits a handle query in the form of a request pointer information UDP packet (“request”) to a handle server to retrieve one or more pointers (“any location information”) associated with the digital object (“associated with the respective one of the data entities”) identified by the handle in the request. (*See* Section IX.A.1.iii.) In particular, Kahn discloses the client system sends the disclosed “request” containing the handle in a handle query/request pointer information UDP packet (“request”) to the handle server to determine if the handle server has one or more pointers for retrieval to locate the requested digital object. (*Id.*; Ex. 1005, 29:5-12 (**“Handle servers...process handle query requests. If the handle which is the subject of the query is found by a handle server, the object pointers associated with the handle are returned to the requesting client.** A handle server accepts a handle as input and returns a list of pointers associated with the handle...”), 47:5-11 (**“The [client] system subsequently sends the handle to the handle server as part of a request pointer information UDP packet (step 178).** The handle server then returns its response to the requesting system. If the handle server found the pointers, a list of pointers is returned.”), 73:32-34 (“The handle server verifies that the handle falls within the set of handles for whose hash values it is responsible...”); *see also id.*, 7:5-7 (“A

handle server, accessible on the network, provides the pointer in response to presentation of a handle.”), 43:20-23 (“[A] user can present a handle to a handle server and be advised of some repository which...contains the digital object designated by the handle.”), 44:6-7 (“A handle is sent to a handle server to locate network addresses of repositories containing that object.”).)

79. Additionally, Kahn discloses that “in response to” such a “request,” “each one of the data location servers is configured to return all location information associated with a respective one of the data entities.” For example, as discussed for limitation [1.b], Kahn discloses each handle server (“data location server”) returns all pointers (“all location information”) associated with a handle identifying a digital object (“data entity”) to the client system. (*See* Section IX.A.1.iii.) In particular, Kahn discloses “[a] handle server accepts a handle as input and returns a list of pointers associated with the handle,” where the handle identifies a digital object. (Ex. 1005, 29:8-12; *see* Section IX.A.1.iii.) Kahn further discloses that the full handle record (i.e., including the handle itself and all corresponding data) may be returned in response to a request from the client, which includes returning all pointers (“location information”) associated with a respective digital object (“respective one of the data entities”). (Ex. 1005, 42:1-11 (“We now describe how a client...resolves any handle....[T]he client request may

ask for all the data in a handle record....”) 44:16-21 (“To resolve a handle, a handle server receives as input a handle and returns some or all of the fields of typed data in the corresponding handle record. **The client can request that all data fields in the handle record be returned....”**.) Accordingly, Kahn discloses this limitation.

5. **Claim 5: “The system of claim 1, wherein the location information includes a hash table and the hash table includes an association between a hash of the string identifier and at least one identifier of the at least one of the plurality of data location servers.”**

80. Kahn discloses this limitation. **First**, Kahn discloses “the hash table includes an association between a hash of the string identifier and at least one identifier of the at least one of the plurality of data location servers.” For example, as discussed for limitations [1.pre] and [1.b], Kahn discloses the hash range/handle-server-domain-name table (“hash table”) includes an association of a range of hash codes/values (“hash of the string identifier”)⁷ and the domain name of a handle server (“at least one identifier of the at least one of the plurality of data location servers”). (See Sections IX.A.1.i, IX.A.1.iii.) Indeed, Kahn discloses

⁷ I assume the claimed “string identifier” is intended to refer to the “identifier string” in claim 1. (See Section IX.A.1.ii.)

calculating a hash code/value by applying a hash algorithm to the handle (“hash of the string identifier”) and using that value to search the hash table, which associates the hash of the handle (“a hash of the string identifier”) to the domain name (“at least one identifier”) of the handle server (“at least one of the plurality of data location servers”) containing the handle and its associated pointers. (Ex. 1005, 46:22-47:5 (“In one example, the handle server directory...holds a table...which associates hash ranges...with domain names of handle servers 58 (Figure 7)....In a first step 170, a client system downloads the table that associates hash ranges with handle server domain names from the handle server directory for future use. The client also can omit this step if it has previously stored the table; frequent changes in the table may necessitate doing this every time....**The system then generates the hash code for the handle using a predetermined hashing algorithm (step 174) and consults the hash range/handle-server-domain-name table to determine the domain name of the appropriate handle server (step 176).**”), 73:4-74:26 (“The system requesting the digital object obtains...the hash code/handle server table from the handle server directory.... This is done during the system’s initialization....Once the handle is obtained, the system that retrieves the object ‘hashes’ the object’s handle and uses this hash value to perform a table lookup in the **hash code/handle server table**.... The requesting system sends a

request-for-pointer information UDP packet...to the handle server. One or more pointers, one returned, identifies the network location of the one or more repositories (if one is associated with the object).....For each pointer returned by the handle server, the requesting system uses the pointer to attempt to obtain a copy of the digital object....”).)

81. **Second**, Kahn discloses “wherein the location information includes a hash table.” As discussed for claim 2, the “location information” on a given handle server “comprises any portion of a hash table,” and thus a person of ordinary skill in the art would have understood that the “location information” collectively comprises/includes a hash table. (*See* Section IX.A.2.) I have been asked to consider a scenario where claim 5 is read to require that the claimed “hash table” is “location information.” Under such a scenario, a person of ordinary skill in the art would have understood that a hash table is “location information” because it provides information regarding the locations of handle servers containing requested pointers, and by extension the locations of digital objects (“data”), since the requested pointers indicate digital object locations. Kahn also discloses that the hash range/handle-server-domain-name table is “location information” under the Interpretation from Other Proceedings, i.e., “information pertaining to one or more locations of data and/or the identities of one or more

location servers.” (See above Section VIII.) For example, as discussed above, the hash table provides information pertaining to the identities of one or more handle servers (i.e., location servers) because it maps hashes of handles to domain names (i.e., identities) of the handle servers. Additionally, the hash table also provides information pertaining to one or more pointers (i.e., locations of data), which provide information about the location of a digital object (including “data”). Accordingly, Kahn discloses this limitation.

6. Claim 6

- i) [6.pre] “A system for managing data location information and providing the data location information in response to location queries, the system comprising:”**

82. I understand that this limitation is a preamble of the claim and I have been asked to assume that the preamble of claim 6 is limiting. Under such an assumption, Kahn discloses the limitations therein. Kahn discloses this limitation for reasons similar to as discussed for limitations [1.pre] and [1.b]. For example, Kahn discloses a system for managing one or more pointers (“data location information”) and providing pointers in response to requests (“location queries”) to retrieve digital objects. (See Sections IX.A.1.i, IX.A.1.iii.)

- ii) **[6.a] “a location server configured to receive a location addition request, the location addition request formatted in conformance with a transfer protocol, the location addition request comprising an identifier and at least one location to associate with the identifier, wherein the identifier identifies an entity and wherein each of the at least one location specifies a location of data in a network pertaining to the entity;”**

83. Kahn discloses and/or suggests this limitation. **First**, Kahn discloses “a location server configured to receive a location addition request...the location addition request comprising an identifier and at least one location to associate with the identifier.” Kahn discloses that when a digital object is deposited in a repository, it is made available in the handle system. (Ex. 1005, 20:4-14 (An “originator”...is an entity that authorizes or validates a set of digital objects; it is responsible for each such digital object including making it available in the handles system...and defining terms and conditions for its use. Every digital object has an originator, which may be an individual or an organization. Originators may deposit and access the digital objects they authorize or validate and may authorize others to do so (this also includes the right to withdraw or modify the objects), subject to the procedures established by individual repositories.”), 37:4-6 (“An originator may control a naming authority, but there may be naming authorities that are not controlled by the originators.”).) In particular, Kahn discloses that when the digital

object is deposited in a repository, a corresponding handle is generated and the handle and associated pointers are inserted into a handle server through an insertion request (“location addition request”). (*Id.*, 20:15-17 (“Naming authorities have the right to insert handle entries for handles they generate into the handle server system and to authorize others to do so.”), 39:34-40:1 (“Naming authorities can choose to store the handles they create on any handle servers for which they have access permissions....”), 43:31-35 (“Handle servers provide a number of services, three of which are RESOLVE, INSERT, and DELETE. A party that is authorized to insert, delete and otherwise change handle entries for a particular naming authority is called a handle administrator.”), 45:10-21 (“However, if the location of the repository were to change, the handle server would have to be notified so it could make the corresponding changes....The handle administrator may determine whether to store IP addresses or domain names or other information [i.e., pointers] in the handle server. The entries are typed and therefore one or more of the above information types may be provided by the administrator for retention in the handle server.”).) Thus, Kahn discloses that a handle server (“a location

server”)⁸ may receive a request to insert a handle and associated one or more pointers (“location addition request”). For reasons discussed above, the “location addition request” includes a handle (“identifier”)⁹ and associated one or more pointers (“at least one location”) indicating the location of the digital object.

84. I have been asked to consider a scenario where Kahn does not explicitly disclose a handle server receiving a location addition request that comprises a handle and associated pointers. Under such a scenario, a person of ordinary skill in the art would have been motivated to include such a feature in Kahn. A person of ordinary skill in the art would have found it beneficial to implement this feature because sending the handle (“identifier”) and associated pointers (“at least one location”) in a single request (“location addition request”) would allow for combining handle insertion and storage processes. Indeed, Kahn discloses processes for inserting handles and associated information (e.g., pointers)

⁸ Kahn discloses a “location server” under the Interpretation from Other Proceedings for the same reasons as discussed for limitation [1.b]. (Sections VIII, IX.A.1.iii.)

⁹ Kahn discloses an “identifier” under the Interpretation from Other Proceedings for the same reasons as discussed for limitation [1.a]. (Sections VIII, IX.A.1.ii.)

into a handle server (Ex. 1005, 20:4-17, 37:4-6, 39:34-40:1, 43:31-35, 45:10-21), and further discloses that the information is stored on the handle server in a table that associates handles with pointers (*id.*, 7:17-20, 37:31-33). A person of ordinary skill in the art would have understood that combining the insertion and storage processes, such that already-associated handles and pointers are sent in a single location addition request, would have reduced processing steps and communication overhead since additional steps (beyond an insertion request) would not be required to associate handles and pointers on the handle server.

85. A person of ordinary skill in the art would have also had a reasonable expectation of success in implementing this modification. Kahn discloses that handles and associated information may be sent to a handle server (Ex. 1005, 20:4-17, 37:4-6, 39:34-40:1, 43:31-35, 45:10-21), and thus the handle server would be readily able to receive a location addition request containing a handle and associated pointers. Moreover, Kahn is already capable of storing handles along with associated pointers (*id.*, 7:17-20, 37:31-33). Such a modification would have involved applications of known technologies (e.g., known handle server storing a table that associates handles with pointers) according to known methods (e.g., known handle server receiving a request to insert handles and associated information) to yield the predictable result of a handle server receiving a handle

and associated pointers as part of a location addition request, as discussed above.

86. **Second**, Kahn discloses “the location addition request formatted in conformance with a transfer protocol.” For example, Kahn discloses the handle system is based on the UDP protocol, which “enables a large number of transactions to be handled efficiently.” (Ex. 1005, 26:8-14, FIG. 33 (showing a diagram of portions of a handle system).) Indeed, Kahn discloses that messages/requests sent to a handle server are sent as UDP packets. (*Id.*, 47:5-8, 54:12-13, 59:23-25, 73:21-22.) Thus, a person of ordinary skill in the art would have understood that transactions such as requests sent to a handle server to insert a handle and associated one or more pointers are formatted in conformance with the UDP protocol (“a transfer protocol”).

87. **Third**, Kahn discloses “wherein the identifier identifies an entity and wherein each of the at least one location specifies a location of data in a network pertaining to the entity.” For reasons discussed for limitation [1.pre], Kahn discloses that a handle identifies an object (“the identifier identifies an entity”) and a pointer indicates the location of the sequence of bits/digits (“data”) in the network pertaining to the object (“each of the at least one location specifies a location of data in a network pertaining to the entity.”) (*See* Section IX.A.1.i.)

88. Kahn discloses “data in a network pertaining to the entity” under the

Interpretation from Other Proceedings for “data pertaining to an entity,” i.e., “data pertaining to a person or thing (real, digital, or abstract) distinct from that data.”

(See above Section VIII.) Kahn discloses that its system is for managing information associated with real-world objects, including conventional works such as books, papers, images, and sounds (i.e., a thing that is real). (Ex. 1005, 1:16-20 (“Thus digital objects may include conventional digital representations of works (books, papers, images, sounds....”).) A person of ordinary skill in the art would have understood that the sequences of bits/digits (“data”) pertaining to objects in the system are associated with real-world objects and form the data representation of those objects (i.e., digital objects), but are merely information stored in repositories and thus are distinct from the real-world objects themselves.

iii) [6.b] “wherein the location server includes a processor; and”

89. Kahn discloses this limitation. As discussed for limitation [1.b], each handle server (“location server”) may be a computer and thus includes a “processor.” (Section IX.A.1.iii.)

- iv) **[6.c] “programming logic stored on the location server, wherein the programming logic is configured to return, in response to a location query related to a desired entity, a location message, the location message in conformance with the transfer protocol and comprising at least one location associated with the desired entity, wherein the programming logic is further configured to return the location message if the location server contains location information for the desired entity, and wherein the programming logic is further configured to return a redirect message if the location server lacks the location information for the desired entity, the redirect message comprising a list of at least one other location server known to have the location information for the desired entity.”**

90. The Kahn-Vingralek combination discloses and/or suggests this limitation. **First**, Kahn discloses “programming logic stored on the location server.” For example, as discussed for limitation [6.b], Kahn discloses that each handle server (“location server”) may be a computer that includes a processor. (See Section IX.A.6.iii.) Kahn further discloses that the handle system interfaces with the handle servers using routines written in the C programming language. (Ex. 1005, 40:31-41:2 (“The handle system provides a client library of routines...currently written in the C programming language....They can be included in client programs that wish to contact handle servers.”).) Thus, a person of ordinary skill in the art would have understood that operations/services provided on the computer, such as a handle-to-pointer translation service (*id.*, 21:6-31,

26:28-31, 29:5-12, 42:2-3, 44:6-7, 65:14-69:14), use associated “programming logic” stored and configured on each handle server in order to perform each of the operations of the handle server.

91. **Second**, Kahn also discloses “wherein the programming logic is configured to return, in response to a location query related to a desired entity, a location message.” For example, Kahn discloses that a handle server (having “programming logic” stored and configured thereon) returns a message containing a list of pointers (“return[s]...a location message”) indicating the location of a digital object in response to a request related to an object (“in response to a location query related to a desired entity”). In particular, the request to retrieve the digital object includes sending the handle in the request to a handle server, whereupon a message containing a list of associated pointers is returned to the client system to locate and retrieve the digital object. (*Id.*, 29:8-12 (“A handle server accepts a handle as input and returns a list of pointers associated with the handle, where each pointer = {domain name of storage system (repository), domain name of RMS}.”), 47:5-11 (“If the handle server found the pointers, a list of pointers is returned.”).)

92. **Third**, Kahn discloses “the location message in conformance with the transfer protocol and comprising at least one location associated with the desired

entity.” As discussed for limitation [6.a], Kahn discloses the handle system and associated transactions are based on the UDP protocol. (Ex. 1005, 26:8-14 (“The handle system is based on the UDP protocol. This enables a large number of transactions to be handled efficiently...”), FIG. 33; Section IX.A.6.ii.) Indeed, Kahn discloses that messages sent from a handle server, including messages containing pointers (e.g., as part of separate registration processes), are sent as UDP packets. (Ex. 1005, 59:28-31, 60:3-7, 60:15-18, 74:1-4, 74:13-17.) Thus, a person of ordinary skill in the art would have understood that Kahn’s “location message[s]” are formatted in conformance with the UDP protocol (“a transfer protocol”). (Ex. 1014, 1 (“[User Datagram Protocol] provides a procedure for application programs **to send messages** to other programs with a minimum of protocol mechanism.”), Ex. 1016, 4:37-5:12 (incorporating by reference RFC 768 [Ex. 1014])). Further, the message containing a list of pointers (“location message”) includes at least one pointer and, as discussed for limitations [1.pre] and [1.b], the at least one pointer is associated with the desired object (“entity”). (*See* Sections IX.A.1.i, IX.A.1.iii; Ex. 1005, 29:8-12, 47:10-11.)

93. **Fourth**, Kahn discloses “wherein the programming logic is further configured to return the location message if the location server contains location information for the desired entity.” For example, as discussed above, Kahn

discloses returning a message containing a list of pointers (“location message”) containing one or more pointers (“location information”) indicating the location of the digital representation (i.e., digital object) of an object (“entity”). This message is sent if the handle server (“location server”) contains the one or more pointers associated with the desired object (“location information”). (Ex. 1005, 47:10-11 (“If the handle server found the pointers, a list of pointers is returned.”).)

94. **Fifth**, Kahn discloses “and wherein the programming logic is further configured to return a redirect message if the location server lacks the location information for the desired entity.” As an initial matter, I have been asked to assume that the claimed “redirect message” is met by a cache miss response due to stale or incorrect location information. Applying this assumption, Kahn discloses this limitation in several ways.

95. For example, Kahn discloses transmitting a not-responsible-for-handle message (“redirect message”) to the client system in the form of an invalid-handle-server-selected response UDP packet if the handle server, i.e., a “wrong handle server,” does not contain the handle and its associated pointers for an object (“if the location server lacks the location information for the desired entity”) indicating the location of the digital object. (Ex. 1005, 47:13-15 (“If the handle was sent to the wrong handle server, it returns a **not-responsible-for-handle message** (step

182).”), FIG. 8 (step 182); *see also id.*, 73:1-74:26 (explaining the same lookup procedure in further detail but referring to the not-responsible-for-handle message by the type of packet, i.e., invalid-handle-server-selected response UDP packet, which is sent if the handle server is not responsible for the handle), 73:32-74:4 (“The handle server verifies that the handle falls within the set of handles for [which] it is responsible 750. **If it is not in this set, ...the handle server sends an invalid-handle-server-selected response UDP packet to the requestor 752.**”), FIG. 25.) Kahn discloses that the transmission of the not-responsible-for-handle-message to the client system causes the client to download an updated table to determine/calculate the location of a correct handle server. (*Id.*, 47:13-21 (“If the handle was sent to the wrong handle server, it returns a not-responsible-for-handle message (step 182). In this case, the client system should download the hash range/handle-server-domain-name table [shown in Fig. 7] from the handle server directory again and attempt the mapping again. The requester will determine how many times to try before giving up....”); *see also id.*, 73:17-20 (“Once the handle is obtained, the system that retrieves the object ‘hashes’ the object’s handle and uses this hashed value to perform a table lookup in the hash code/handle server table....”), 74:5-12 (“If the requesting system receives an invalid-handle-server-selected response UDP packet, it refreshes its hash code/handle server table

[shown in Fig. 7] from the handle server directory, and the requesting system repeats prior steps. This will typically be needed only if the table has changed between the time the table was downloaded and the actual request was made.”.)
Thus, Kahn discloses a handle server (having “programming logic” stored and configured thereon) that returns a not-responsible-for-handle-message in the form of an invalid-handle-server-selected response UDP packet (“redirect message”) if the handle server (“location server”) lacks one or more pointers (“location information”) indicating the location of information for the requested object.

96. I have been asked to consider a scenario where Kahn does not explicitly disclose “the redirect message **comprising a list of at least one other location server known to have the location information for the desired entity.**” In that scenario, a person of ordinary skill in the art would have been motivated to implement this feature in view of Vingralek.

97. As discussed for limitation [1.b], a person of ordinary skill in the art would have been motivated to consider the teachings of Vingralek when implementing Kahn’s system/method. (See Section IX.A.1.iii.)

98. Vingralek additionally discloses propagating updates to clients’ and other servers’ address tables in a deferred manner. (Ex. 1007, 6.) As discussed for limitation [1.b], Vingralek discloses a method of deferred address table updates in

which updates to a client's stale address table are deferred until a wrong first server is reached and the location of a second server needs to be calculated. (Section IX.A.1.iii.) In particular, Vingralek discloses a first server calculating the location of a "correct" server to send a request to, and further discloses that if a first server does not have the requested information, a message ("redirect message") is sent to the client with information to update the client's address table. (*Id.*) Vingralek discloses that the "most salient feature" of its approach, where a client's address table is updated in a deferred manner, "is that its communication overhead is largely independent of the number of servers and clients in the system; thus, it is considered a scalable approach." (Ex. 1007, 5.) Thus, Vingralek discloses implementing a redirection mechanism and additionally includes improved features to minimize the number of servers storing data and reduce the cost of scaling up a distributed system. (Ex. 1007, 6, 8.)

99. A person of ordinary skill in the art would have been motivated to implement Vingralek's deferred address table updates in the Kahn system, for instance, by utilizing Kahn's existing mechanism of notifying the client. For instance, Kahn, similar to Vingralek, also uses a deferred update process, where the transmission of a not-responsible-for-handle message ("redirect message") results in the download of an updated hash range/handle-server-domain-name table

for determining/calculating the location of a correct handle server after the initial attempt reached a wrong handle server. Thus, a person of ordinary skill in the art would have been motivated to transmit an updated hash table to the client in such a deferred manner, i.e., in a not-responsible-for-handle message (“redirect message”) after a wrong server was reached in an initial attempt, particularly given that Vingralek discloses such a method would limit communication overhead as discussed above. Decoupling the communication overhead from the number of servers/clients in the system would have improved scalability, consistent with Kahn’s goal and expectation of scaling. (Ex. 1005, 9:23-27, 33:3-14.)

100. A person of ordinary skill in the art would have further understood that Vingralek’s method of updating the client’s address table via a redirect message would also be beneficial to provide the addresses of “correct” location servers to a client. Kahn discloses an arrangement in which a message is first returned to the client, upon which the client downloads an updated hash table, calculates/determines the location of a correct handle server, and forwards the request to the correct handle server. (*Id.*, 46:25-47:25, 73:1-74:32.) However, as discussed above, Vingralek discloses a method that both calculates the addresses of “correct” location servers, and updates the client’s address table, without requiring additional downloads or calculating a location server’s address by the client. A

person of ordinary skill in the art would have understood that Vingralek's redirect message including an address table may be implemented such that only the portion of the address table corresponding to the location server(s) that contain the requested location information ("a list of at least one other location server known to have the location information for the desired entity") would be sent in the message. This would allow the client to update the relevant portions of its own hash/address table, as well as provide the client with the addresses of the location server(s) that should receive the forwarded request. Such a modification would have been beneficial as it would have reduced the frequency of communications to a client and/or the use of the client's computational resources, particularly in situations where the client has limited communication bandwidth and/or computational resources (e.g., resources for calculating the correct handle server). (Ex. 1007, 5, 7, 8.) Thus, a person of ordinary skill in the art would have been motivated to include features similar to Vingralek's "redirect message" features when implementing Kahn.

101. Consistent with the above and in such a combined system, for example, the client would apply the hash algorithm to the handle and its own copy of the hash range/handle-server-domain-name table and then send a request to a handle server ("location server"), which would apply the hash function to the

handle and its own copy of the hash range/handle-server-domain-name table to determine whether it has the requested location information. If the handle server does not have the requested pointers associated with the object (“if the location server lacks the location information for the desired entity”), the handle server would calculate the location of another handle server that has the requested pointers (“at least one other location server known to have the location information for the desired entity”). The handle server would then send a redirect message containing a portion of the hash table corresponding to the handle servers known to have the pointers (“wherein the programming logic is further configured to return a redirect message...the redirect message comprising a list of at least one other location server known to have the location information for the desired entity”). Such a redirect message (similar to Vingralek’s disclosed message to update a client’s address table) would allow the handle server to update its stale hash table, and additionally provide the client with the addresses of a correct handle server, to which the handle server then forwards the request.

102. A person of ordinary skill in the art would have understood that the above-described implementation, in which the handle server calculates the location of a correct handle server for the client to forward a request to, would have been one of only several options (i.e., calculating and forwarding at either the client or

the handle server that processes the information request). Thus, a person of ordinary skill in the art would have been motivated to implement the system to transmit a redirect message with a list of at least one other handle server known to have the request pointers. A person of ordinary skill in the art would have further understood that implementing such a redirect message would allow hash table updates to propagate from handle server to client, while reducing the number of downloads performed from the handle directory servers (i.e., by cutting down the number of downloads by the client).

103. A person of ordinary skill in the art would have also had a reasonable expectation of success combining the teachings of Kahn and Vingralek. As discussed above, both references are in the same field of art. (Section IX.A.1.iii.) Moreover, Kahn itself suggests that its system/process is flexible and additional processes may be included, particularly with respect to the process of relaying a request between a user and handle server. (Ex. 1005, 21 (explaining that “intermediary services” may be implemented between users and servers).) Additionally, it would have involved modifying the Kahn system in a manner that Kahn suggests is feasible—implementing a means of occasionally updating a hash range/handle-server-domain-name table, and implementing a means of providing domain names of correct handle servers to the client. Moreover, such a

modification would have involved applications of known technologies (e.g., known distributed system (Kahn)) according to known methods (e.g., known methods of providing deferred updated hash tables using a redirect message (Vingralek)) to yield the predictable result of a system that updates hash tables in a manner that limits communication overhead and allows for scalability, as discussed above. Accordingly, Kahn in view of Vingralek discloses and/or suggests this limitation.

7. **Claim 7: “The system of claim 6, wherein the list of at least one other location server comprises a list of a plurality of location servers in the network and a corresponding list of a plurality of entities having location information on the respective one of the plurality of location servers.”**

104. Kahn discloses this limitation. As discussed for limitation [1.b], Kahn discloses that a hash table maps domain names of multiple handle servers in a network (“plurality of location servers in the network”) with corresponding hash codes/values (each a hash of a handle). (*See* Section IX.A.1.iii; Ex. 1005, 38:9-17, 46:22-24.) As further discussed for limitation [1.b] with respect to the location information retrieval process, the hash table indicates which handle servers store location information associated with a given handle. (*See* Section IX.A.1.iii.) As discussed for limitations [1.pre] and [1.a], handles identify real-world objects (“plurality of entities”). (*See* Sections IX.A.1.i-ii.) Thus, a person of ordinary skill

in the art would have understood that a hash table includes a list of handle servers in the network (“a list of a plurality of location servers in the network”), along with a corresponding list of hashed handles (which represent objects (“entities”)) that store location information on those servers (“corresponding list of a plurality of entities having location information on the respective one of the plurality of location servers”). (Ex. 1005, 7:8-9, 37:31-38:3, 46:22-24; *see also* Section IX.A.1.iii.)

105. As discussed for limitation [6.c], a person of ordinary skill in the art would have been motivated to implement a redirect message returned to the client that contains a portion of the hash table corresponding to the “correct” handle server (“the list of at least one other location server”) that contains the requested pointers (“location information”). (*See* Section IX.A.6.iv.) A person of ordinary skill in the art would have understood that there may be multiple such “correct” handle servers, as Kahn discloses that data may be replicated across multiple handle servers. (Ex. 1005, 39:24-29.) A person of ordinary skill in the art would have thus understood that like the full hash table, the redirect message with a portion of a hash table (“the list of at least one other location server”) also comprises a list of the correct handle servers (“a list of a plurality of location servers in the network”) and a corresponding list of hashed handles indicating

which objects have location information on each server (“corresponding list of a plurality of entities having location information on the respective one of the plurality of location servers”).

8. Claim 8: “The system of claim 6, wherein each of a plurality of location servers in the network stores only a portion of the data location information.”

106. Kahn discloses this limitation. As discussed for claim [1.b], Kahn discloses that each of the multiple handle servers in the network (“each of a plurality of location servers in the network”) stores only a portion of the one or more pointers (“stores only a portion of the data location information”) as the one or more pointers are partitioned amongst multiple handle servers. (*See* Section IX.A.1.iii.)

9. Claim 9: “The system of claim 6, wherein the location information in the location server is maintained in an indexed location store.”

107. Kahn discloses this limitation. Kahn discloses that pointers (“location information”) in a handle server (“location server”) are maintained in a table (“indexed location store”) associating handles with pointers. In particular, Kahn explains “each handle server ha[s] a table which associates handles with pointers.” (Ex. 1005, 7:17-20 (“Hash values are computed on the handles and the hash values are distributed among multiple handle servers, each handle server having a table

which associates handles with pointers.”), 37:31-33 (“Handle servers have the following characteristics: a handle server holds pointers associated with a subset of all handles....”).) A person of ordinary skill in the art would have understood that the handle server has a storage medium to store the table (“location store”) of one or more pointers and associated handles. A person of ordinary skill in the art would have further understood that the table in the handle server is an “indexed location store” because each handle may be used as an index to retrieve the associated one or more pointers (“location information”). Accordingly, Kahn discloses this limitation.

10. Claim 10: “The system of claim 9, wherein the indexed location store comprises a string store indexed by a hash table.”

108. Kahn discloses this limitation. Kahn discloses “the indexed location store comprises a string store.” Kahn explains that handles are unique character strings, and pointers consist of a repository domain name which a person of ordinary skill in the art would understand is also a string. (Ex. 1005, 7:4-5 (“The handles comprise unique character strings associated with the servers which generated them.”), 29:8-12 (“A handle server accepts a handle as input and returns a list of pointers associated with the handle, where each pointer = {domain name of storage system (repository), domain name of RMS}.”).) Thus, a person of ordinary

skill in the art would have understood that the handle server's table associating handles with pointers comprises a "string store."

109. Second, Kahn discloses that the handle server's table ("indexed location store comprises a string store") is "indexed by a hash table." As discussed for claim 9, the handle is used as an index to retrieve all associated pointers from the handle server's table. (See Section IX.A.9.) Kahn discloses that the indices (i.e., the handles) of the handle server's table are arranged/distributed to multiple handle servers based on a hash function/hash table ("indexed by a hash table") that maps handles to handle servers. (Ex. 1005, 47:1-11 ("**The system** then generates the hash code for the handle using a predetermined hashing algorithm (step 174) and **consults the hash range/handle-server-domain-name table to determine the domain name of the appropriate handle server (step 176). The system subsequently sends the handle to the handle server** as part of a request pointer information UDP packet (step 178). The handle server then returns its response to the requesting system. **If the handle server found the pointers, a list of pointers is returned (step 180).**"), 73:17-27 ("Once the handle is obtained, the system that retrieves the object 'hashes' the object's handle and **uses this hash value to perform a table lookup in the hash code/handle server table.... The requesting system sends a request-for-pointer-information UDP packet...to the handle**

server. One or more pointers, once returned, identified the network location of the one or more repositories (if one is associated with the object)...”).)

Accordingly, Kahn discloses this limitation.

11. Claim 11: “The system of claim 9, wherein the indexed location store comprises a string store indexed by a hash table distributed across a plurality of clients.”

110. Kahn discloses this limitation. As discussed for claim 10, Kahn discloses “the indexed location store comprises a string store indexed by a hash table.” (*See* Section IX.A.10.)

111. Kahn discloses multiple clients (“a plurality of clients”) (Ex. 1005, 38:34, 40:11), and explains that a hash range/handle-server-domain-name table (“hash table”) is shared/distributed to each client during initialization, and an updated table is shared/distributed to a client if the handle server does not contain the handle and its associated pointers (“a hash table distributed across a [client]”). (*Id.*, 46:29-47:21 (“In a first step 170, a client system downloads the table that associates hash ranges with handle server domain names from the handle server directory for future use....If the handle was sent to the wrong handle server, it returns a not-responsible-for-handle message (step 182). In this case, the client system should download the hash range/handle-server-domain-name table [shown in Fig. 7] from the handle server directory again and attempt the mapping again.

The requester will determine how many times to try before giving up....”), 73:4-74:12 (“The system requesting the digital object obtains...the hash code/handle server table from the handle server directory.... This is done during the system’s initialization.... If the requesting system receives an invalid-handle- server-selected response UDP packet, it refreshes its hash code/handle server table from the handle server directory, and the requesting system repeats prior steps. This will typically be needed only if the table has changed between the time the table was downloaded and the actual request was made.”).) Thus, a person of ordinary skill in the art would have understood that a hash range/handle-server-domain-name table may be shared/distributed across multiple client systems (“a hash table distributed across a plurality of clients”).

112. Kahn discloses the claimed “client” under the Interpretation from Other Proceedings, i.e., “a network-attached component (which may be software or hardware) that initiates update or lookup of identifier/location mappings from a location server with location request messages.” (*See* above Section VIII; *see also* Ex. 1001, 4:14-17.) For example, Kahn discloses a user providing a handle to a client system (“client”) to obtain a pointer indicating the location of the digital object in the distributed network. (Ex. 1005, 46:25-47:25, 73:1-74:32.) Moreover, Kahn discloses that the client system’s user interface may be “in the form of

software running on a workstation.” (*Id.*, 26:28-34 (“As seen in Figure 2, an example system...includes a digital object management system...which includes hardware and software to create and store digital objects....In the system, a user agent (UA)...provides a user interface for interactions with other elements of the system. **The UA is in the form of software running on a workstation.**”), FIG. 2 (displaying the UA on the client system).) Kahn further discloses that the user interface is part of a system that operates on a network. (*Id.*, 4:23-32, 19:26-31, 26:28-28:28, FIG. 2.) Thus, a person of ordinary skill in the art would have recognized that the client system (“client”) is a hardware and/or software component attached to a network. As further discussed for limitation [1.b], the client system also sends a request (i.e., location request message) to a handle server (i.e., location server) in order to initiate lookup of mappings of handles (i.e., identifiers) to one or more pointers (which identify a digital object’s location). (*See* Section IX.A.1.iii.)

12. Claim 12: “The system of claim 9, wherein the indexed location store comprises a string store indexed by a hash table distributed across a plurality of servers.”

113. Kahn discloses this limitation. As discussed for claim 10, Kahn discloses “the indexed location store comprises a string store indexed by a hash table.” (*See* Section IX.A.10.)

114. Kahn discloses “a hash table distributed across a plurality of servers.” For example, Kahn also explains the hash range/handle-server-domain-name table is distributed across handle directory servers (“a hash table distributed across a plurality of servers”). In particular, Kahn discloses a hash range/handle-server-domain-name table shared/distributed across servers called handle directory servers, such that each handle directory server stores a hash range/handle-server-domain-name table. (Ex. 1005, 4:29-32 (“A [handle] directory server accepts identifiers [i.e., handles] of any of the digital objects and maintains and returns a table containing the locations of the [handle] servers which accept those identifiers.”), 29:18-26 (“The handle server directory...allows users to find the correct handle server for processing a given handle. Users which obtain handles from servers associate them with digital objects. The handle server directory...distributes handles to handle servers based upon a hash of handles. The handle server directory provides a list of domain names of handle servers and identifies the set of hash values of handles which each of the handle servers can map to pointers.”), 38:11-14 (“The [handle] directory servers will maintain a table of network addresses of handle servers....”), 44:8-10 (“The handle is first mapped to locate the handle server from the handle directory server table....”).) Accordingly, Kahn discloses this limitation.

- 13. Claim 13: “The system of claim 9, wherein the location query identifying the desired entity comprises a unique identifier for the desired entity, and wherein the programming logic stored on the location server further comprises programming logic configured to apply an index function to the unique identifier to retrieve at least a portion of the locations associated with the unique identifier in the indexed location store.”**

115. The Kahn-Vingralek combination discloses and/or suggests this limitation. **First**, Kahn discloses “wherein the location query identifying the desired entity comprises a unique identifier for the desired entity.” As discussed for limitation [6.pre], Kahn discloses a request (“location query”) to retrieve a requested object (“desired entity”). (See Section IX.A.6.i.) As discussed for limitation [6.c], the request sent from client to handle server includes the handle (“unique identifier”) for the requested object (“location query...comprises a unique identifier for the desired entity”), and thus a person of ordinary skill in the art would have understood that the request identifies the desired object (“location query identifying the desired entity”) since each handle uniquely identifies an object. (Ex. 1005, 47:5-11 (“**The [client] system subsequently sends the handle to the handle server as part of a request pointer information UDP packet (step 178).** The handle server then returns its response to the requesting system. If the handle server found the pointers, a list of pointers is returned.”); Section IX.A.6.iv; *see also* Section IX.A.1.ii.)

116. **Second**, the Kahn-Vingralek combination discloses and/or suggests “wherein the programming logic stored on the location server further comprises programming logic configured to apply an index function to the unique identifier to retrieve at least a portion of the locations associated with the unique identifier in the indexed location store.” As discussed for limitation [6.c], Kahn discloses programming logic on a handle server (“location server”) that is configured to perform a handle-to-pointer translation service. (*See* Section IX.A.6.iv.) As further discussed, in the Kahn-Vingralek combination, the handle server (“location server”) is configured to apply a hash algorithm (“index function”) to the handle (“unique identifier”) as part of the process for retrieving a portion of pointers (“at least a portion of the locations”) indicating the location of a digital object associated with the handle (“unique identifier”) that are stored in the table (“indexed location store”) of the handle server (“configured to apply an index function to the unique identifier to retrieve at least a portion of the locations associated with the unique identifier in the indexed location store”). (*See* Section IX.A.6.iv.) A person of ordinary skill in the art would have recognized that an index function is a function that returns an index to a table, and thus a hash algorithm/function, which returns a value that serves as an index to the hash table, is an “index function.”

14. Claim 14: “The system of claim 13, wherein the indexed location store comprises a string store and the index function comprises a hash function.”

117. The Kahn-Vingralek combination discloses and/or suggests this limitation. As discussed for claim 10, Kahn discloses “the indexed location store comprises a string store indexed by a hash table.” (*See* Section IX.A.10.) As discussed for claim 13, the Kahn-Vingralek combination discloses and/or suggests that the hash algorithm applied to the handle is a hash function (“index function comprises a hash function”). (*See* Section IX.A.13.)

15. Claim 15

i) [15.pre] “A method of handling location queries in a network, the network comprising a plurality of location servers including data location information, the method comprising:”

118. I understand that this limitation is a preamble of the claim and I have been asked to assume that the preamble of claim 15 is limiting. Under such an assumption, Kahn discloses the limitations therein. Kahn discloses this limitation for reasons similar to as discussed for limitation [1.pre]. For example, Kahn discloses a method for handling requests (“location queries”) in a network (“network”) to retrieve pointers identifying the location of a digital object, where the network contains multiple handle servers storing pointers (“a plurality of

location servers including data location information”).¹⁰ (*See* Section IX.A.1.i.)

- ii) **[15.a] “correlating each one of a plurality of identifiers with at least one of a plurality of locations in the network, each one of the plurality of identifiers identifying a respective one of a plurality of data entities, wherein the data entities are stored in corresponding locations in the network;”**

119. Kahn discloses this limitation. To begin, Kahn discloses “correlating each one of a plurality of identifiers with at least one of a plurality of locations in the network.” As discussed for limitations [1.pre] and [1.b], Kahn discloses a network of handle servers (“the network”), and discloses that each handle server correlates/associates (“correlating”) each handle (“each one of a plurality of identifiers”)¹¹ with one or more pointers indicating the location of the digital object in the network (“with at least one of a plurality of locations in the network”). (*See* Sections IX.A.1.i, IX.A.1.iii.)

120. Kahn also discloses “each one of the plurality of identifiers

¹⁰ Kahn discloses a “location server” and “plurality of location servers” under the Interpretation from Other Proceedings for the same reasons as discussed for limitation [1.b]. (Sections VIII, IX.A.1.iii.)

¹¹ Kahn discloses an “identifier” under the Interpretation from Other Proceedings for the same reasons as discussed for limitation [1.a]. (Sections VIII, IX.A.1.ii.)

identifying a respective one of a plurality of data entities, wherein the data entities are stored in corresponding locations in the network.” As discussed for limitation [1.a], Kahn discloses that each handle (“each one of the plurality of identifiers”) uniquely identifies a digital object (“identifying a respective one of a plurality of data entities”), where the digital objects are stored in corresponding locations indicated by pointers, i.e., digital objects are stored in repositories, in the network (“wherein the data entities are stored in corresponding locations in the network”). (See Section IX.A.1.ii.) Accordingly, Kahn discloses this limitation.

iii) [15.b] “receiving a location query from a client at one of the plurality of location servers, the location query requesting location information identifying a location of a data entity included in the data entities;”

121. Kahn discloses this limitation. As discussed for limitation [1.b], the client (“client”) transmits a request (“location query”) to a handle server (“receiving a location query from a client at one of the plurality of location servers”) for pointers (“the location query requesting location information”) associated with the handle in the request, where those pointers identify the location (“location information identifying a location”) of the specific digital object (among the larger group of digital objects stored on the network) (“data entity included in the data entities”) associated with the handle. (Section IX.A.1.iii.)

iv) [15.c] “determining which of the plurality of location servers includes the location information;”

122. The Kahn-Vingralek combination discloses and/or suggests this limitation. As discussed for limitation [1.b], the Kahn-Vingralek combination discloses and/or suggests that each handle server may determine/calculate which other handle server (“determining which of the plurality of location servers”) contains the pointers (“includes the location information”) associated with a handle. (Section IX.A.1.iii.) Accordingly, the Kahn-Vingralek combination discloses this limitation.

v) [15.d] “sending a location response message to the client in response to determining the one of the plurality of location servers includes the location information, the location response message comprising the location information; and”

123. Kahn discloses this limitation. As discussed for limitation [6.c], Kahn discloses that if a handle server (“location server”) has the pointers (“location information”) that were requested (“in response to determining the one of the plurality of location servers includes the location information”), the handle server returns a message to the client system (“sending a location response message to the client”) with the requested pointers (“the location response message comprising the location information”). (See Section IX.A.6.iv.)

- vi) **[15.e] “sending a redirect message to the client in response to determining the one of the plurality of location servers fails to include the location information, the redirect message identifying which of the plurality of location servers includes the location information.”**

124. The Kahn-Vingralek combination discloses and/or suggests this limitation. As discussed for limitation [6.c], the Kahn-Vingralek combination discloses and/or suggests that if a handle server (“location server”) determines it does not have one or more pointers (“location information”) that were requested (“in response to determining the one of the plurality of location servers fails to include the location information”), the handle server returns a message to the client system (“sending a redirect message to the client”) that identifies which of the other handle servers includes the requested one or more pointers (“the redirect message identifying which of the plurality of location servers includes the location information”). (*See* Section IX.A.6.iv.)

- 16. Claim 16: “The method of claim 15, further comprising applying an identifier of the data entity to an indexing function to determine which of the location servers includes the location information.”**

125. Kahn discloses this limitation. To begin, Kahn discloses that a handle identifying a digital object is hashed (“applying an identifier of the data entity”) using a predetermined hash algorithm (“to an indexing function”). (*See* Section

IX.A.1.i; Ex. 1005, 47:1-5, 73:17-20.) The client system then uses the output of the hash algorithm, i.e., a hash code/value, to index into the hash range/handle-server-domain-name table to determine the appropriate handle server (“to determine which of the location servers”) that contains one or more pointers (“includes the location information”) indicating the location of the digital object . (See Section IX.A.1.i; Ex. 1005, 47:1-13, 73:17-27.) If the handle server contains the handle and associated pointers, the pointers are returned to the client for it to locate and retrieve the digital object associated with the handle. (See Section IX.A.1.i; Ex. 1005, 47:1-13, 73:17-27.) Accordingly, Kahn discloses this limitation.

- 17. Claim 17: “The method of claim 16, wherein each one of the plurality of location servers comprises a portion of an indexed table of the identifiers and corresponding associated locations, and wherein determining which of the plurality of location servers includes the location information comprises applying one of the identifiers to a hash function to look up corresponding associated locations in the indexed table, the one of the identifiers identifying the data entity.”**

126. Kahn discloses this limitation. **First**, Kahn discloses “wherein each one of the plurality of location servers comprises a portion of an indexed table of the identifiers and corresponding associated locations.” As discussed for limitation [1.b], Kahn describes a hash function that is applied to a handle (“identifier”) to

generate an index to a hash table, where the hash table maps the hash of the handles (“identifiers”) to corresponding addresses of handle servers (“corresponding associated locations”) that store information pertaining to a certain handle. (*See* Section IX.A.1.iii.) A person of ordinary skill in the art would have understood that the hash table is therefore “an indexed table of the identifiers and corresponding associated locations.” As discussed for claim 2, Kahn discloses that each handle server (“each one of the plurality of location servers”) stores location information (i.e., pointers) that comprise a portion of a hash table. (*See* Section IX.A.2.) Thus, a person of ordinary skill in the art would have understood that each handle server (“each one of the plurality of location servers”) also comprises a portion of the hash table (“comprises a portion of an indexed table of the identifiers and corresponding associated locations”) because each server stores data corresponding to a portion of the hash table.

127. **Second**, Kahn discloses “wherein determining which of the plurality of location servers includes the location information comprises applying one of the identifiers to a hash function to look up corresponding associated locations in the indexed table, the one of the identifiers identifying the data entity.” As discussed for limitation [1.b], Kahn discloses determining the location of a handle server that stores the requested pointers (“determining which of the plurality of location

servers includes the location information”) by using a hash function that takes a handle as an input (“applying one of the identifiers to a hash function”). (*See* Section IX.A.1.iii.) As further discussed above, the result of the hash function serves as an index for looking up associated handle server addresses (“to look up corresponding associated locations”) in the hash table (“in the indexed table”). As discussed for limitation [1.a], Kahn discloses that a handle (“identifier”) identifies the corresponding digital object (“data entity”). (*See* Section IX.A.1.ii.) Thus, a person of ordinary skill in the art would have understood that a handle in the request (“the one of the identifiers”) identifies the digital object that will ultimately be retrieved (“identifying the data entity”).

X. CONCLUSION

128. I declare that all statements made herein of my knowledge are true, and that all statements made on information and belief are believed to be true, and that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code.

Dated: June 28, 2024

By:  _____

Sandeep Chatterjee