

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	10
	B. Hash Functions and Applications	12
VI.	THE '170 PATENT	15
	A. Description	15
	B. The '896 Application Does Not Support All the Features of Claims 1-17	18
VII.	OVERVIEW OF THE PRIOR ART	20
	A. Skagerwall	20
	B. Vingralek	26
VIII.	CLAIM CONSTRUCTION	29
	A. "identifier"	30
	B. "location information"	30
	C. "location server"	30
	D. "client"	31
	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"	31

F.	“data pertaining to the entity”	32
G.	“plurality of location servers”	32
I.	THE PRIOR ART DISCLOSES AND/OR SUGGESTS ALL RECITED FEATURES OF CLAIMS 1-17 OF THE '170 PATENT	35
A.	Skagerwall Discloses Claims 1, 2, 4, and 5	36
1.	Claim 1	36
2.	Claim 2	61
3.	Claim 4	63
4.	Claim 5	64
B.	Skagerwall Discloses and/or Suggests Claim 3	66
1.	Claim 3	66
C.	Skagerwall in View of Vingralek Discloses and/or Suggests Claims 6-17	71
1.	Claim 6	71
2.	Claim 7	96
3.	Claim 8	97
4.	Claim 9	98
5.	Claim 10	99
6.	Claim 11	100
7.	Claim 12	101
8.	Claim 13	103
9.	Claim 14	104
10.	Claim 15	105
11.	Claim 16	111
12.	Claim 17	112
II.	CONCLUSION	114

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. 1021	U.S. Patent Application No. 09/111,896
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 understand that the earliest application associated with the '170

patent is U.S. Patent Application No. 09/111,896 (“the ’896 application”), which was filed on July 8, 1998. (Ex. 1001, 1:6-15.) As discussed in Section VI.B below, in my opinion, a person of ordinary skill in the art reviewing the ’896 application would *not* have understood that the named inventor was in possession of the subject matter recited in claims 1-17 of the ’170 patent. Thus, I have been asked to consider that the time of the alleged invention of the ’170 patent was around 1999 (including and up to September 14, 1999), which corresponds to the filing date of Provisional Application No. 60/153,709 for the ’170 patent. (*Id.*)

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 September 14, 1999. 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 September 14, 1999) 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 September 14, 1999.

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

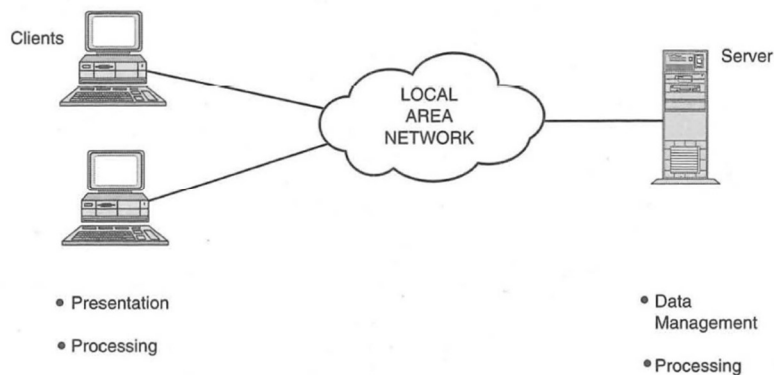
suggestions from that prior art as I explain below in Section IX. The concepts below would have been within the knowledge and mindset of a person of ordinary skill in the art at the time.

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.”).)

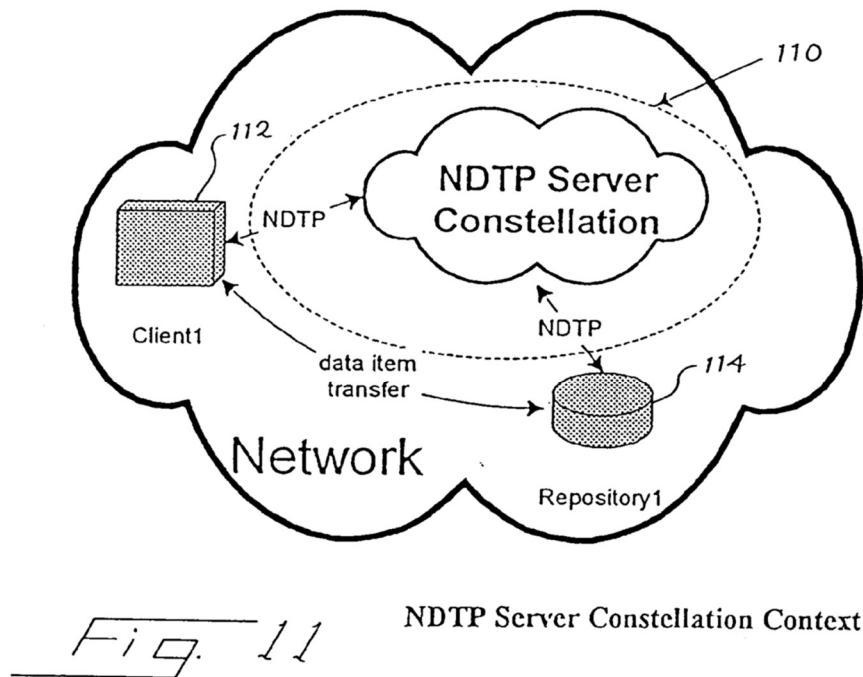
28. When data is not found at the specified location, perhaps due to the data being stored in a new location in the system, a message with a forwarding address may be returned to the requesting system. (Ex. 1059, 553-54 (“If object *o* is at node *n* after its a^{th} move, then the pair $\langle n, a \rangle$ 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

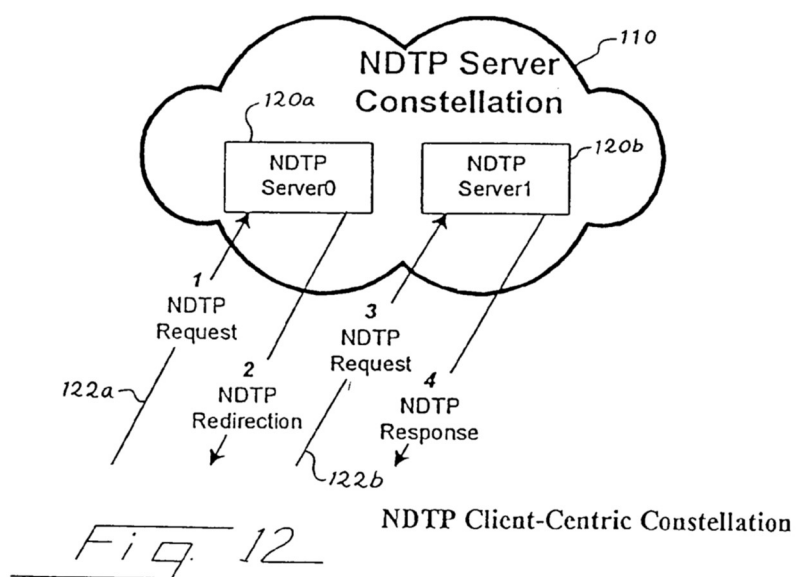
29. The '170 patent describes storing and retrieving information across a network (Ex. 1001, 1:29-32, 1:58-60) and a network distributed tracking protocol (NDTP) (*id.*, 2:14-16, 4:27-38). Figure 11 shows an NDTP server constellation where a client consults servers that return strings in response to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

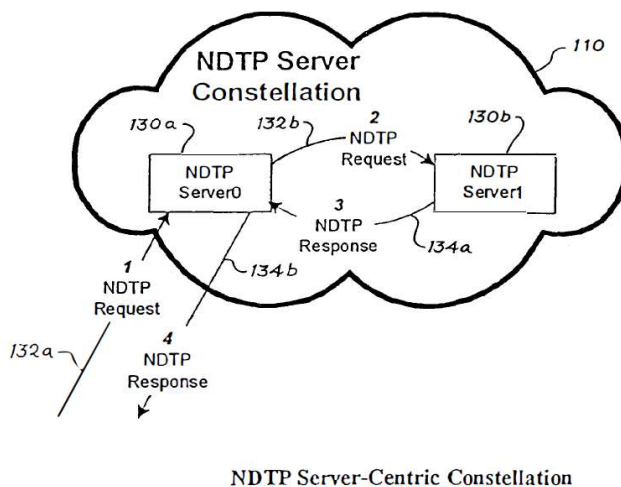


Fig. 13

(*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.*)

B. The '896 Application Does Not Support All the Features of Claims 1-17

32. I understand that the '170 patent issued from U.S. Patent Application No 11/354,224, filed on February 13, 2006, which claims priority to U.S. Patent Application No. 09/661,222 (now the '640 patent) filed on September 13, 2000; Provisional Application No. 60/153,709 filed on September 14, 1999; and the '896 application filed on July 8, 1998. (Ex. 1001, Cover, 1:6-15.)

33. I have been asked to consider whether a person of ordinary skill in the art reviewing the '896 application would have understood that the named inventors were in possession of the subject matter recited in claims 1-17 of the '170 patent at the time of the '896 application, i.e., July 8, 1998. As discussed below, it is my opinion that the '896 application does not support the features of the independent claims of the '170 patent, and thus does not support the features of any of the claims of the '170 patent.

34. In particular, it is my opinion that at least limitations [1.b], [6.c], and [15.e] of the '170 patent are not supported by the '896 application.

35. Limitation [1.b] requires that the data location information stored on data location servers is organized across the servers based on a hash function. The '896 application does not provide support for the use of a hash function or indexing the contents of a data location server using hashing. At most, the '896 application describes an identifier that serves as an index of the data objects themselves (e.g., medical records). (Ex. 1021, 1:4-7, 2:8-3:18, 6:10-17, 7:9-8:14, 9:4-12:5, 13:1-10, 14:17-26:10, 28:6-41:14, 44:8-16, 45:1-48:10, FIG. 1.) However, the '896 application does not provide support for the application of any indexing function to distribute data location information across data location servers, nor does it provide support for the use of a hash function in any context.

36. Limitations [6.c] and [15.e] require that if a data location server does not contain the requested data location information associated with the desired entity or data entity, the server returns a redirect message that identifies other data location server(s) containing the requested location information. The '896 application does not provide support for any such redirect message, nor does it provide support for any of the operations associated with a redirect message. At most, the '896 application describes a “server response message[]” that “answers

client queries, and sometimes also performs actions on behalf of the client called server actions.” (Ex. 1021, 15:23-26.) Although the ’896 application discloses that server response messages may be sent in response to a number of events, it does not disclose the claimed “redirect message” and associated operations recited in claims 6 and 15, as described above. (*Id.*, 15:27-24:24, 29:7-41:10.)

37. Accordingly, it is my opinion that a person of ordinary skill in the art reviewing the ’896 application would *not* have understood that the named inventors were in possession of the subject matter recited in claims 1-17 of the ’170 patent. Based on my opinion on this matter, I have been asked to consider that the time of the alleged invention of the ’170 patent was around 1999 (including and up to September 14, 1999), which corresponds to the filing date of Provisional App. No. 60/153,709 for the ’170 patent.

VII. OVERVIEW OF THE PRIOR ART

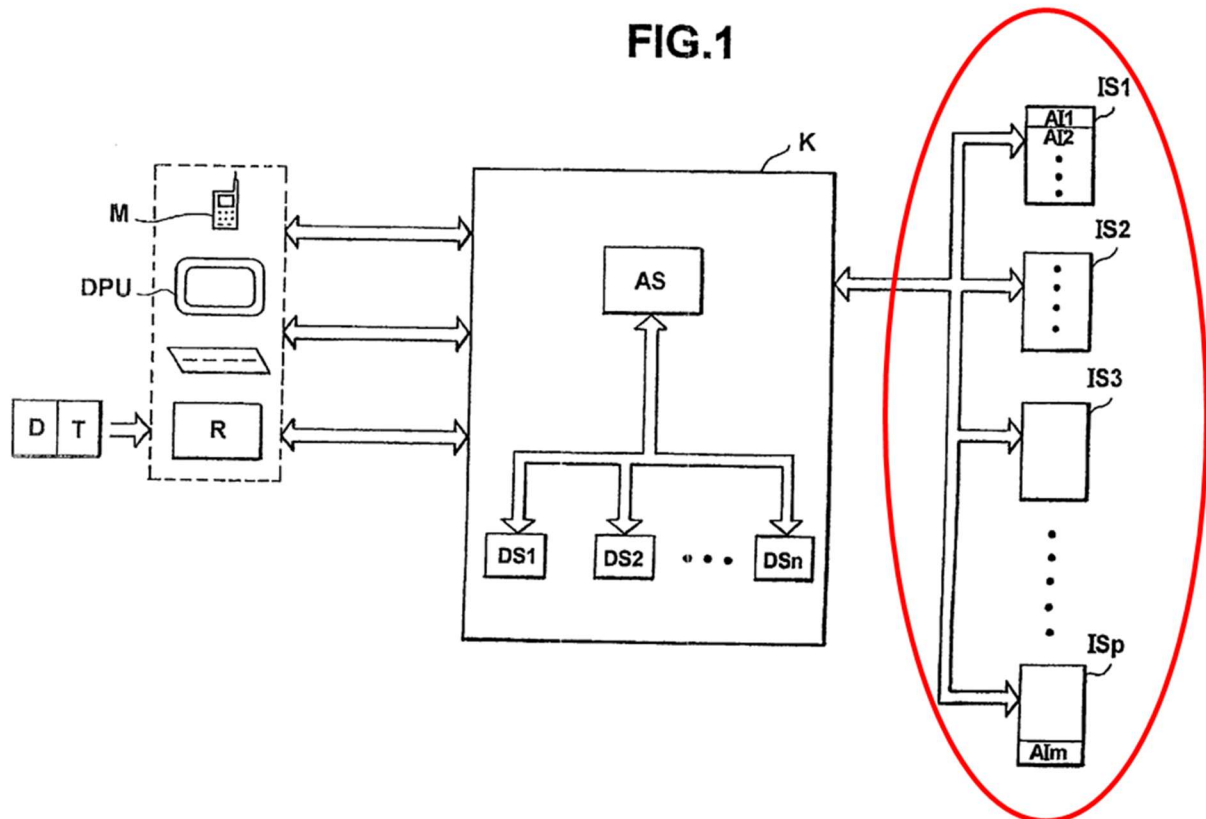
A. Skagerwall

38. Skagerwall discloses a communication system that links objects, such as products or items in the physical world, to different types of information related to the objects. (Ex. 1006, Abstract (“[c]ommunication system and method for linking each of a plurality of tagged object[s] to information services stored on a network”), 6:45-48 (“[t]he system is arranged to link the plurality of objects, which

can, for example, be any products or items of the physical world to various kinds of information related to the objects”).) The system allows for linking an arbitrary number of objects to information or application data packets about those objects, where the application data packets are stored on application servers. (*Id.*, 4:1-5 (“The invention advantageously allows to link an arbitrary number of objects of the real world to information services stored on an arbitrary number of data processing units or application servers, which may be located at arbitrary places.”), 4:6-11 (“The data processing units or application servers preferably store a plurality of application data packets comprising data related to the services. Each of the plurality of application data packets may be associated with one of the plurality of objects of the real world, i.e., it contains information related to the object.”).) The application data packets may include text, images, video, audio, application programs, or other kinds of messages. (*Id.*, 4:5-9 (“The services may be text, images, application programs or any other kind of messages.”), 7:13-17 (“Data stored on the application servers can, for example, be text, such as a user manual of a product, links to other servers storing related data or other kind of written information. Further, the data can comprise images, video, audio signals and the like.”).)

39. Skagerwall discloses that the application data packets are stored in a

distributed network of application servers. For example, Skagerwall discloses with respect to Figure 1 that the application data packets, each identified by an application identifier (shown as AI1-AIm below), are distributed across a plurality of application servers, each identified by an address (shown as IS1-ISp below). (*Id.*, 6:34-37 (“The system kernel K is arranged to communicate data between a plurality of application servers and a telephone M, a data processing unit DPU or the like. The application servers each have an address IS; IS1-ISp and store a plurality of application data packets, each having an application identifier AI1-Aim.”).)

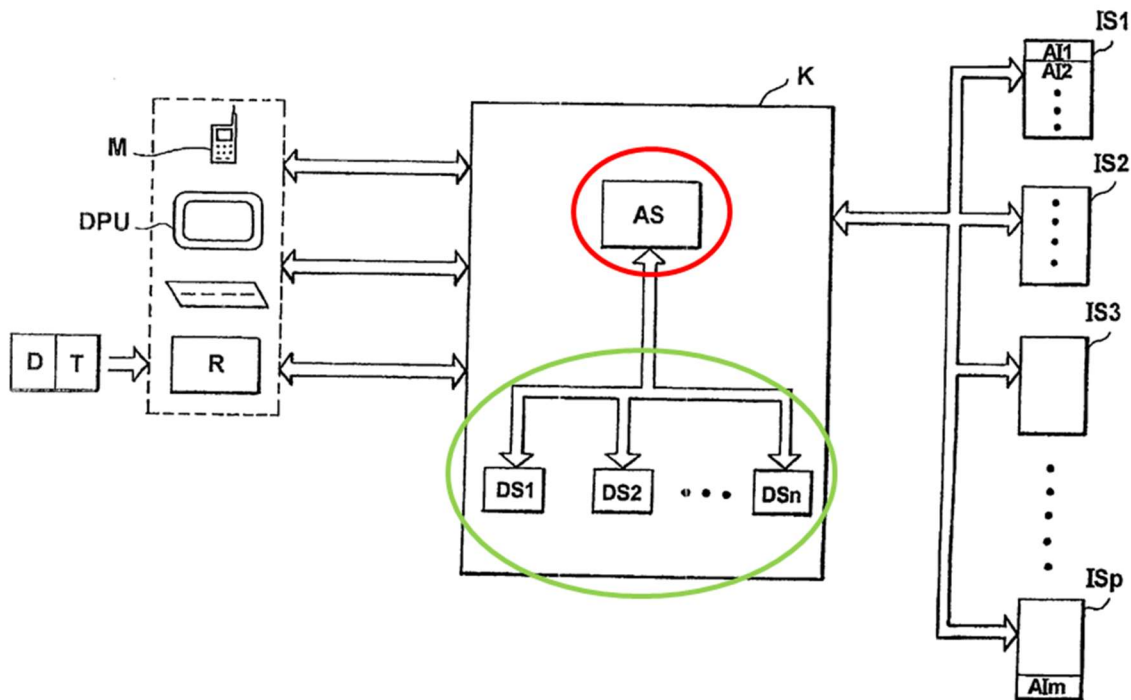


(*Id.*, FIG. 1 (annotated to show application servers, identified by addresses IS1-ISp, and the application data packets, identified by identifiers AI1-Aim, in red).) The application servers are arranged in a distributed network because computing resources and data are spread across multiple servers, with servers able to communicate with one another without going through a centralized point. Indeed, Skagerwall discloses that the application servers may be placed at arbitrary locations, similar to computer networks such as intranets, company networks, or the Internet. (*Id.*, 7:4-8 (“The plurality of application servers with the addresses IS; IS1-ISp and storing data related to the objects D may be placed at arbitrary locations in the world, similar to today’s computer networks, such as intranets, company networks, or the Internet.”).)

40. Skagerwall discloses with respect to Figure 1 that the system includes an access server and a network of directory servers. (*Id.*, 6:32-34 (“The embodiment of FIG. 1 comprises a system kernel K with at least one [access] server AS; AS1-ASr and a plurality of directory servers DS; DS1-DSn.”).)²

² A person of ordinary skill in the art would have understood that the reference to “server AS; AS1-ASr” as “application server[s]” is a typographical error, as these are referred to as “access server[s]” elsewhere in the specification, while the

FIG.1



(*Id.*, FIG. 1 (annotated to show access server AS in red and directory servers DS1-DSn in green).) Each directory server stores object data, which associates objects with application data packets, and serves as a key resource in the retrieval of application data packets from application servers. (*Id.*, 4:13-16 (“[E]ach of a plurality of directory servers stores part of an arbitrarily scaleable data base containing object data associating each of the objects with at least one of the

servers IS; IS1-ISp are referred to as the addresses of “application servers.” (*See, e.g.*, Ex. 1006, 9:1-9.)

application data packets. The plurality of directory servers serve as [a] key resource in retrieving an application data packet related to an object.”), 8:19-22 (“Each of the directory servers DS; DS1-DSn preferably stores object data related to a small number of the range of available object tags T; T1-Tw.”), FIG. 1; *see also id.*, Abstract (“The system reads an object tag of one of the objects and retrieves information about the object stored on storage located at arbitrary places.”).) Skagerwall explains that the object data is associated with an object tag and stored on a directory server, and the object data includes application identifiers of application data packets and addresses of the application servers storing those packets, which together identify where an application data packet is stored. (*Id.*, 8:22-27 (“[O]bject data related to an object tag preferably includes application identifiers AI1-AIm of application data packets related to the object as well as information about storage locations of the application data packets, e.g., the addresses of application servers IS; IS1-ISp storing the application data packets.”), 15:15-52 (explaining how the location of a given application data packet on a network of application servers is characterized by the application server address as well as the application identifier).)

41. Skagerwall discloses that these application identifiers and addresses of application servers identifying application data packets stored in application

servers may be retrieved as one part of a two-step process for the retrieval of application data packets. (*Id.*, 8:28-36.) The first step comprises identifying a directory server storing object data (which includes application identifiers and addresses) relating to an object, and retrieving that object data. (*Id.*, 8:22-32; *see also id.*, 9:5-9 (“[T]he at least one access server AS; AS1-ASr is responsible for retrieving addresses IS; IS1-ISp of application servers obtained from at least one of the directory servers DS; DS1-DSn and/or retrieving corresponding application data packets.”), 9:42-48 (“The obtained addresses IS; IS1-ISp and optionally other information can be returned to the access server AS; AS1-ASr, which...retrieves the specified application data packets from the identified application servers ...”), 14:7-11 (“[T]he directory server storing the object data determines from the object data application server addresses IS; IS1-ISp, storing application data packets associated with the object and returns retrieved address and application identifiers to the access server AS; AS1-ASr.”).) The second step involves using the object data to retrieve application data packets from the application servers, or alternatively, transmitting the object data directly to the user (e.g., a mobile telephone or data processing unit). (*Id.*, 8:32-36.)

B. Vingralek

42. 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.”).)

43. 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.*)

44. 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

45. 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 September 14, 1999) 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.³

A. “identifier”

46. 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”

47. 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”

48. 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

³ 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”

49. 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”⁴

50. 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

⁴ 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”

51. 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”

52. 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.”

53. 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”

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

54. In my opinion, the prior art discussed below discloses and/or suggests the limitations of claims 1-17.⁵

⁵ 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.

A. Skagerwall Discloses Claims 1, 2, 4, and 5

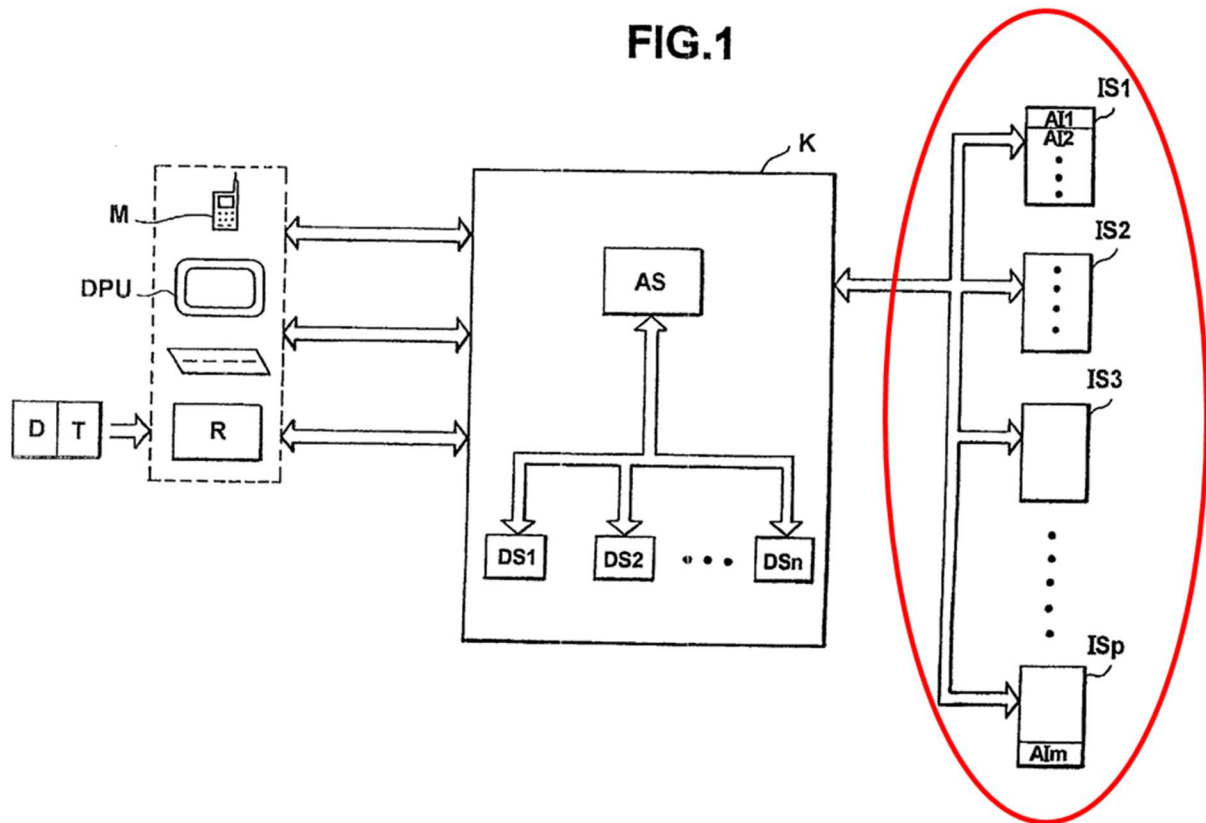
1. Claim 1

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

55. 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 assumption, Skagerwall discloses the limitations therein. To begin, Skagerwall discloses a system that links objects, e.g., products or items in the physical world, to different types of information related to the objects. (Ex. 1006, Abstract (“[c]ommunication system and method for linking each of a plurality of tagged object[s] to information services stored on a network”), 6:45-48 (“[t]he system is arranged to link the plurality of objects, which can, for example, be any products or items of the physical world to various kinds of information related to the objects”).) The system allows for linking objects to information or application data packets (the claimed “data”) about those objects, where the application data packets are stored on application servers. (*Id.*, 4:1-5 (“The invention advantageously allows to link an arbitrary number of objects of the real world to information services stored on an arbitrary number of data processing units or application servers, which may be located at arbitrary places.”), 4:6-11 (“The data processing units or application servers preferably store a plurality of application

data packets comprising data related to the services. Each of the plurality of application data packets may be associated with one of the plurality of objects of the real world, i.e., it contains information related to the object.”.) The application data packets (“data”) may include text, video, or programs. (*Id.*, 4:5-9 (“The services may be text, images, application programs or any other kind of messages.”), 7:13-17 (“Data stored on the application servers can, for example, be text, such as a user manual of a product, links to other servers storing related data or other kind of written information. Further, the data can comprise images, video, audio signals and the like.”).)

56. Skagerwall discloses that the application data packets (“data”) are stored in a distributed network of application servers (“stored in a distributed network”). For example, as shown in Figure 1, the application data packets, each identified by an application identifier (AI1-AIm), are distributed across a plurality of application servers (red), each identified by an address (IS1-ISp). (*Id.*, 6:34-37 (“The system kernel K is arranged to communicate data between a plurality of application servers and a telephone M, a data processing unit DPU or the like. The application servers each have an address IS; IS1-ISp and store a plurality of application data packets, each having an application identifier AI1-Aim.”).)



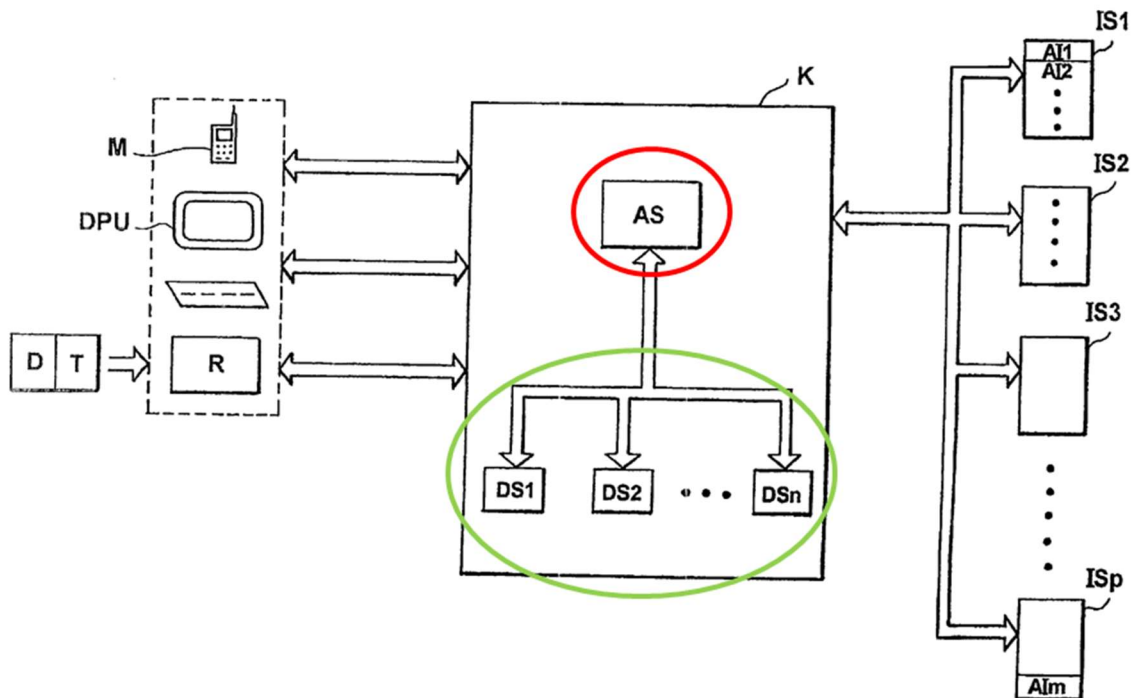
(*Id.*, FIG. 1 (annotated).) Consistent with Figure 1, a person of ordinary skill in the art would have understood that the application servers are arranged in a distributed network because computing resources and data are spread across servers that communicate with one another without going through a centralized point. Indeed, the application servers may be placed at arbitrary locations, similar to intranets, company networks, or the Internet, which a person of ordinary skill in the art would have recognized are distributed networks. (*Id.*, 7:4-8 (“The plurality of

application servers with the addresses IS; IS1-ISp and storing data related to the objects D may be placed at arbitrary locations in the world, similar to today's computer networks, such as intranets, company networks, or the Internet.”.)

57. Moreover, Skagerwall discloses that its communication system is for “managing data stored in a distributed network.” For example, Skagerwall discloses a process for the retrieval of application data packets (“data stored in a distributed network”), which a person of ordinary skill in the art would have understood is a way of managing the application data packets. Skagerwall discloses with respect to Figure 1 that the system includes an access server (red) and a network of directory servers (green). (*Id.*, 6:32-34 (“The embodiment of FIG. 1 comprises a system kernel K with at least one [access] server AS; AS1-ASr and a plurality of directory servers DS; DS1-DSn.”).)⁶

⁶ A person of ordinary skill in the art would have understood that the reference to “server AS; AS1-ASr” as “application server[s]” is a typographical error, as these are referred to as “access server[s]” elsewhere in the specification, while the servers IS; IS1-ISp are referred to as the addresses of “application servers.” (*See, e.g., Ex. 1006, 9:1-9.*)

FIG.1



(*Id.*, FIG. 1 (annotated).) Each directory server stores object data, which associates objects with application data packets (“data”), and is involved in the retrieval of application data packets from application servers (“data stored in a distributed network”). (*Id.*, 4:13-16 (“[E]ach of a plurality of directory servers stores part of an arbitrarily scaleable data base containing object data associating each of the objects with at least one of the application data packets. The plurality of directory servers serve as [a] key resource in retrieving an application data packet related to an object.”), 8:19-22 (“Each of the directory servers DS; DS1-DSn preferably stores object data related to a small number of the range of available object tags T;

T1-Tw.”), FIG. 1; *see also id.*, Abstract (“The system reads an object tag of one of the objects and retrieves information about the object stored on storage located at arbitrary places.”).) The object data is associated with an object tag and includes application server addresses indicating the storage location of the application data packets, as well as application identifiers of the packets. (*Id.*, 8:22-27 (“[O]bject data related to an object tag preferably includes application identifiers AI1-AIm of application data packets related to the object as well as information about storage locations of the application data packets, e.g., the addresses of application servers IS; IS1-ISp storing the application data packets.”).)

58. The application server addresses, which indicate the locations of application data packets (“data”) stored on the network of application servers (“stored in a distributed network”), may be retrieved as one part of a two-step process for retrieving application data packets associated with an object. (*Id.*, 8:28-36.) The first step comprises identifying a directory server storing object data (including application server addresses) relating to an object and retrieving those addresses. (*Id.*, 8:22-32, 9:5-9 (“[T]he at least one access server AS; AS1-ASr is responsible for **retrieving addresses IS; IS1-ISp of application servers** obtained from at least one of the directory servers DS; DS1-DSn and/or retrieving corresponding application data packets.”), 9:42-48 (“**The obtained addresses IS;**

IS1-ISp and optionally other information can be returned to the access server AS; AS1-ASr, which...retrieves the specified application data packets from the identified application servers”), 14:7-11 (“[T]he directory server storing the object data determines from the object data application server addresses IS; IS1-ISp, storing application data packets associated with the object and **returns retrieved address and application identifiers to the access server AS; AS1-ASr.**”).) The second step involves using the object data to retrieve application data packets (“data”) from the application servers, or alternatively, transmitting the object data directly to the user (e.g., a mobile telephone or data processing unit). (*Id.*, 8:32-36.)

59. In addition to the above process for the retrieval of application data packets associated with a specific object, Skagerwall also discloses storing application data packets. (*Id.*, 14:54-56 (“[T]he above operation may also include storing respective application data packets on application servers.”).) A person of ordinary skill in the art would have similarly understood that storing application data packets (“data stored in a distributed network”) is also a way of managing the application data packets.

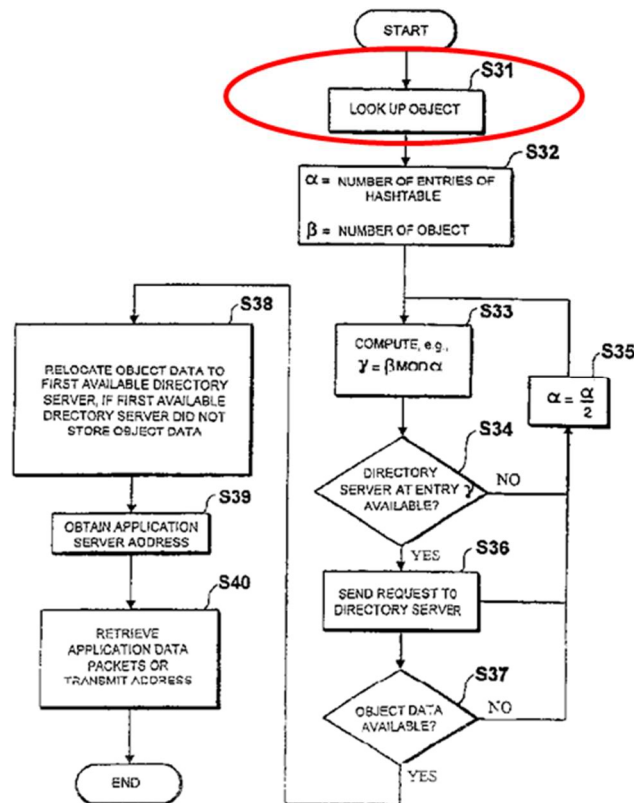
ii) [1.a] “a data repository configured to store a data entity, wherein an identifier string identifies the data entity; and”

60. Skagerwall discloses this limitation. **First**, Skagerwall discloses “a data repository configured to store a data entity.” As discussed for limitation [1.pre], Skagerwall discloses that the application data packets (“data”) associated with an object are stored in a distributed network of application servers. (*See* Section IX.A.1.ii.) A person of ordinary skill in the art would have further understood that the application servers together form a “data repository” because they store the application data packets (“data”) associated with tagged objects. (Ex. 1006, 4:1-12.) A person of ordinary skill in the art would have also understood that all of the application data packets associated with a tagged object are together a “data entity” because they form the data representation of a real-world entity. Accordingly, Skagerwall discloses “a data repository configured to store a data entity” because the application servers store the application data packets.

61. **Second**, Skagerwall discloses “wherein an identifier string identifies the data entity.” With respect to the process for retrieval of application data packets, Skagerwall discloses that the system kernel includes an access server that receives an information request about an object, as shown in step S31 of Figure 3

below, where the request at least includes an object tag (“identifier string”). (Ex. 1006, 9:1-5 (“[A]t least one access server AS; AS1-ASr...is responsible for receiving a request for information about an object including an object tag T; T1-Tw read by the reader R.”), 13:23-26 (“In a step S31 the system kernel K receives a request from a user, requesting information related to an object. A request at least includes an object tag T; T1-Tw of an object D...”), 13:29-30 (“Preferably, the request is received by one of the access servers AS; AS1-ASr.”), FIG. 3; *see also id.*, FIG. 2a (displaying in step S24 a lookup request for object data related to an object tag), 12:27-32 (describing the lookup request in step S24 of Figure 2a).)

FIG.3



(*Id.*, FIG. 3 (annotated).) Skagerwall discloses that the access server receives the request via a reader that reads an object tag (e.g., a barcode) (“identifier string”) and transmits it to the system kernel. (*Id.*, 6:41-43 (“A plurality of objects D is marked with a plurality of tags T... A reader R is provided for reading data of an object tag and transmitting same [to] the system kernel K.”), 7:22-23 (“The reader may be a bar code reader or any other reading device.”).) Upon receipt of this request for application data packets associated with an object, which includes the object tag (“identifier string”), the access server performs operations (e.g., those

involving directory servers) to identify application servers storing application data packets (“data”) related to the tagged object, retrieve the application data packets (together a “data entity”) associated with the object tag from the application servers, and supply the packets to the user. (*Id.*, 7:55-61, 9:5-13, 9:22-27.)

62. A person of ordinary skill in the art would have understood that an object tag such as a barcode is a “string” because a barcode includes a string of characters, e.g., a human readable part or Arabic numerals below the bars and/or the bars themselves readable using a barcode reader. A person of ordinary skill in the art would have further understood that an object tag is an “identifier string” because the tag identifies a given object. (*Id.*, 2:53-54 (“a plurality of objects (D) each identified by an object tag (T)”)). A person of ordinary skill in the art would have similarly understood that the object tag (“identifier string”) not only identifies the object, but also identifies the object’s application data packets (together a “data entity”), because as discussed for limitation [1.pre], the object tag is associated with the object’s application server addresses, which provide the location of the application data packets (together a “data entity”) associated with the tagged object. (*See* Section IX.A.1.i.)

63. Additionally, Skagerwall discloses that the object tag is an “identifier 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; *see also* 1001, 4:10-12.) A person of ordinary skill in the art would have understood that the object tag is an encoding that uniquely identifies both an object (i.e., an individual entity) and its data representation (“data entity”) because, as discussed above, the tag is included in a query in order to identify the specific object for which information is sought. As further discussed for limitation [1.b], the object tag (“identifier string”) is associated with zero or more application server addresses (data location specifiers, and the claimed “data location information”), which specify the location of application data packets (“data”), in a directory 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 the 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.”**

64. Skagerwall discloses this limitation. **First**, Skagerwall discloses “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.” As discussed for limitation [1.pre], Skagerwall discloses a network of multiple directory servers (“a data location server network comprising a plurality of data location servers”). (See Section IX.A.1.i; Ex. 1006, 8:15-19.) The application server addresses (“data location information”) are stored on directory servers (“stored in the data location server network”) and identify the location of application data packets, which form

the data representations of various tagged objects (“for a plurality of data entities”). (See Section IX.A.1.i; Ex. 1006, 8:19-27.) And as further discussed for limitation [1.pre], Skagerwall discloses that if information about a tagged object has been stored, then application server addresses (“location information”) associated with the object’s tag (“identifier string”) are stored on some directory server (“at least one of the plurality of data location servers includes location information associated with the identifier string”). (See Section IX.A.1.i; Ex. 1006, 8:22-27.)

65. Skagerwall discloses the claimed “data location server[]” under the Interpretation from Other Proceedings for “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* 1001, 4:17-20.) Skagerwall discloses each directory server is a component attached to a distributed network. (Ex. 1006, 8:15-19 (“the data base of object data is distributed over a plurality of directory servers”), 16:34-36 (all components in the system may be “connected via a network N for communication with each other”), FIG. 1 (depicting a network of directory servers DS; DS1-DSn); Section IX.A.1.i.) A person of ordinary skill in the art would have also understood that each directory server maintains mappings from object tags (i.e., identifiers) to application server addresses (i.e., locations of application

servers storing application data packets) because Skagerwall discloses that object tags and associated object data are distributed across directory servers based on a hash function, and that each directory server has a memory means associating object tags with corresponding object data (including application server addresses). (Ex. 1006, 8:43-63 (“[T]he hashfunction is used to assign object data related to a group of objects to each of the directory servers. Thus, each of the object tags is assigned to a target directory server using the hashfunction....”), 15:15-20 (“[D]irectory server DS1 further comprises address memory means AM for storing a plurality of address data blocks AB1-ABk, each comprising one of the application identifiers AI1-Aim, the corresponding application server address IS; IS1-ISp and, optionally, corresponding object tags T; T1-Tw.”), 15:49-52 (“[T]he directory server may comprise single memory means, in which case all object data related to an object are stored in the provided memory means.”).)⁷ Further, Skagerwall discloses that the directory server returns identifier/location mappings in response to a request from the access server for location information associated

⁷ Skagerwall’s application server address likewise discloses the “location” (or similar terms) as described in the ’170 patent (Ex. 1001, 4:12-14) for reasons discussed above.

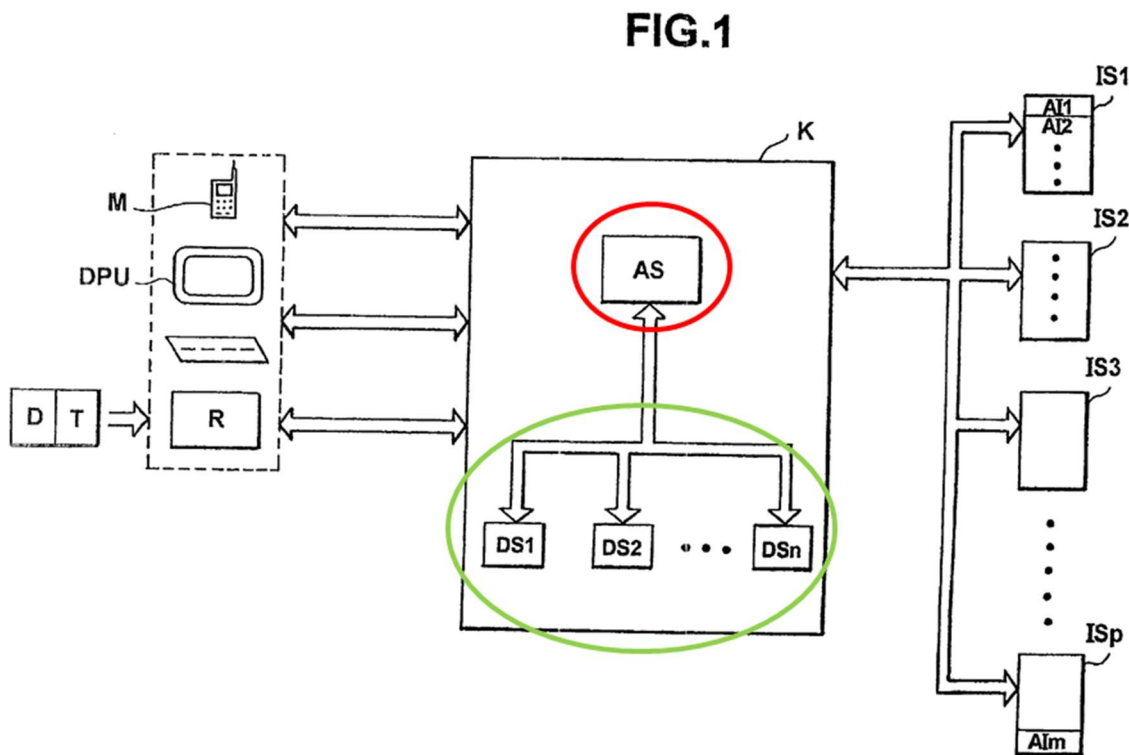
with an object tag (i.e., in response to location request messages from a client).

For example, as discussed below, the application server addresses (i.e., locations) are returned to the access server if the directory server determines it contains the requested location information. (Ex. 1001, 2:36-40, 2:60-65, 6:63-67, 16:53-56.)

Skagerwall also discloses that the corresponding object tags (i.e., identifiers) are returned with the application server addresses. (Ex. 1006, 19:42-46 (“means (DS; DS1-DSn; AS; AS1-ASr) for...retrieving the object data corresponding to the object tag”), 19:51-55 (“object data corresponding to an object comprise an object tag, at least one application identifier and the address of the at least one application server storing the at least one application data packet identified by the at least one application identifier”).) Skagerwall further discloses that identifier/location mappings may be updated/modified in response to a request from the access server to add new location information (i.e., another type of location request message from the client). (*Id.*, 12:3-26, 14:29-56; Ex. 1001, 6:50-53.) Thus, Skagerwall discloses the claimed “data location server[]” under the Interpretation from Other Proceedings for “location server.”

66. Further, Skagerwall 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 directory 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, because—as shown below in Figure 1—there is no hierarchy of intermediary servers between the access server (in red) and the directory servers (in green), and the directory servers themselves are not arranged in a hierarchy.



(Ex. 1006, FIG. 1 (annotated).) Further, Skagerwall discloses that the hash function is used to distribute object tags and associated object data (e.g.,

application server addresses) among the directory servers. (*Id.*, 8:15-27, 8:38-54, 8:55-59.) A person of ordinary skill in the art would have thus understood that because a hash function is used for distribution across directory servers, Skagerwall's directory servers are arranged in a non-hierarchical configuration. Accordingly, Skagerwall discloses this limitation.

67. Skagerwall also 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.) For example, each application server address identifies the location of application data packets ("data"). (Ex. 1006, 8:22-27, 15:15-52.)

68. **Second**, Skagerwall discloses "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." Skagerwall discloses that each directory server ("each one of the plurality of data location servers") has a central processing unit ("comprises a processor"). (*Id.*, 15:1-7 ("The directory server DS1 comprises a central processing unit CPU for

controlling operations of the directory server.”.) Skagerwall also discloses that the database of object data, which includes application server addresses (“data location information”), is distributed across a plurality of directory servers, such that each directory server (“each one of the plurality of data location servers”) stores object data related to certain object tags. (*Id.*, 8:15-27.) Further, Skagerwall discloses that the hash function/hash table distributes the object tags over the available directory servers. (*Id.*, 8:38-54.) “[T]he distribution of object data over the plurality of directory servers thus follows a defined scheme corresponding to the hashfunction,” such that “it is possible to use the same scheme for identifying the target directory server storing object data related to a particular object tag.” (*Id.*, 8:55-59.) In other words, Skagerwall discloses that the object data on a given directory server is only a portion of the full database of object data (“comprises...a portion of the data location information”), and that the full database of object data is distributed across the plurality of directory servers according to a hash function/hash table that assigns object data to corresponding directory servers (“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”).

69. Skagerwall 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.) Skagerwall discloses that its hash function maps object tags (“identifier string[s]”) to one or more directory servers (“data location servers”) by assigning certain object tags to certain directory servers based on the function. (Ex. 1006, 8:43-54 (“[T]he hashfunction is used to assign object data related to a group of objects to each of the directory servers. Thus, each of the object tags is assigned to a target directory server using the hashfunction. Preferably, the hashfunc[tion] distributes the object tags evenly over the available directory servers.”).)

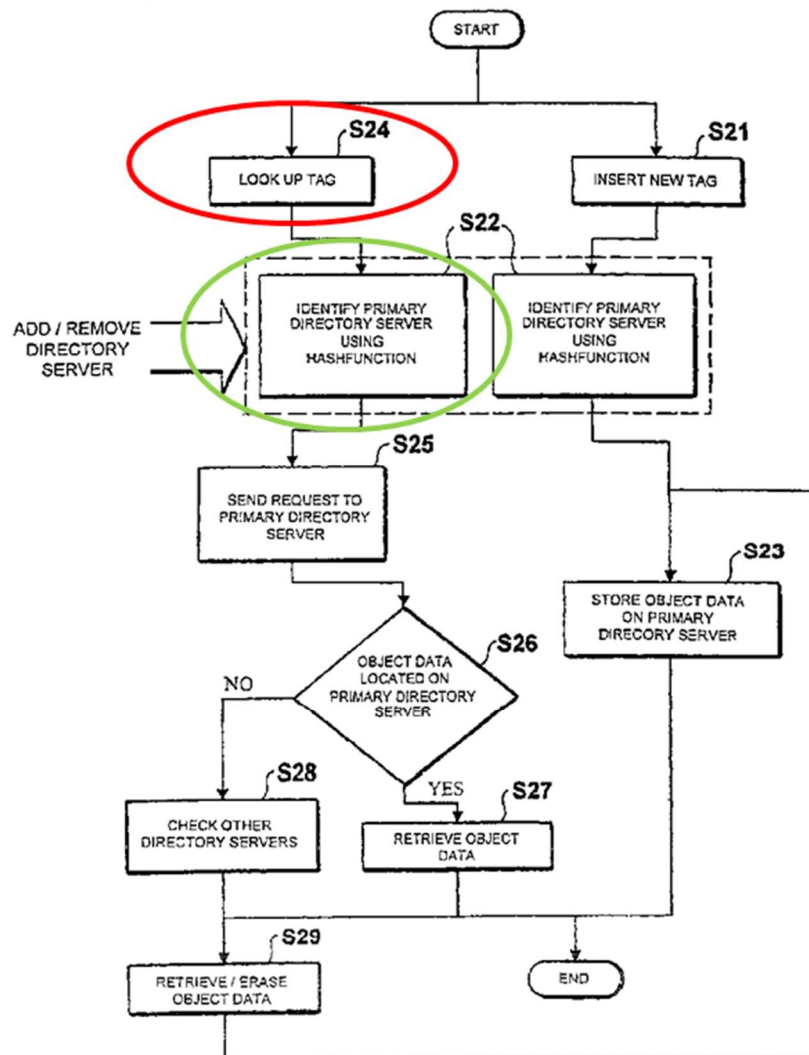
70. **Third**, Skagerwall discloses “each one of the data location servers is configured to determine the at least one of the plurality of data location servers based on the hash function applied to the identifier string.” In particular, as discussed further below, Skagerwall discloses that as part of the process for retrieval of application data packets, a directory server (“data location server”) applies a hash function to an object tag (“identifier string”) in order to determine

the directory server that stores information specifying the location of the application data packets.

71. Skagerwall discloses that upon receipt of the information request, the access server performs operations to identify the directory server (“data location server”) that stores application server addresses identifying the location of application data packets. (*Id.*, 9:18-22 (“After receiving a request for information including the object tag T, the access server AS; AS1-Asr performs operations to identify the directory server DS; DS1-Dsn storing information concerning application data packets related to an object.”).) For example, Skagerwall discloses with respect to Figure 2a that after an information request including an object tag is received in step S24 (red), a primary directory server is determined in step S22 (green) using a hash function. (*Id.*, 12:30-37.) The hash function is applied to the object tag (“identifier string”) from the information request, with the result serving as an index to search a table listing the available directory servers. (*Id.*, 8:38-42 (“The first step of identifying the correct directory server may be performed via a look up in a hashtable H containing addresses or identifiers of all available directory servers DS1-DSn. To facilitate this look up procedure, a hashfunction is used.”), 12:8-12 (“In a step S22 the primary directory server...is computed based on, e.g., a number assigned to the new object tag as well as based

on the number of available directory servers”); *see also id.*, 10:28-11:16, 13:15-57
(describing the application of a hash function to an object tag, with the result
serving as an index into a table of directory servers).)

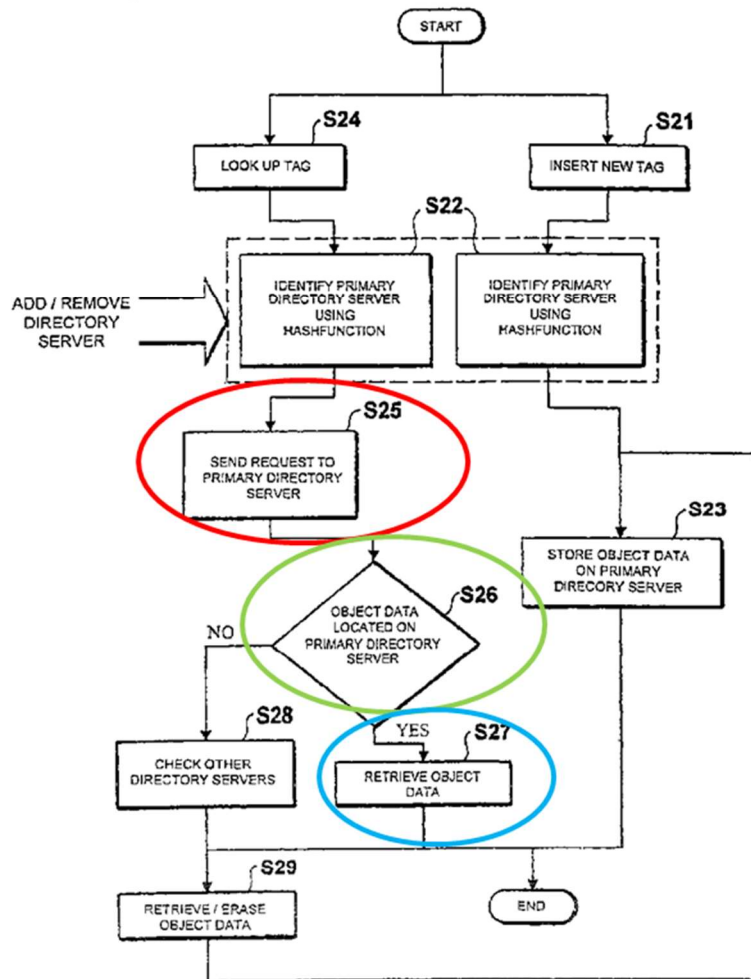
FIG.2a



(*Id.*, FIG. 2a (annotated).)

72. Skagerwall explains that the access server then sends a request to the first directory server in step S25 (red in Figure 2a below) to retrieve application server addresses, where the request from access server to directory server includes the object tag. (*Id.*, 9:9-13 (“[T]he at least one access server AS; AS1-Asr executes the chain of requests connected to retrieving information from the system with respect to the object tag T; T1-Tw...”), 9:23-25 (“[The access server] send[s] the request including the object tag T; T1-Tw to one of the directory servers DS1; DS2; ...; DSn.”), 12:37-40 (access server transmitting a request for object data to the primary directory server in step S25), FIG. 2a (step S25); *see also id.*, 8:22-27 (object data including application server addresses).) In step S26 (green), the primary directory server determines whether it contains the application server addresses associated with the object tag. (*Id.*, 12:41-43; *see also id.*, 8:22-27.) If the directory server contains the application server addresses, then in step S27 (blue), the addresses are retrieved and subsequent steps for providing the user with the desired information are performed (e.g., retrieving application data packets from application servers based on the application server addresses). (*Id.*, 12:54-62.)

FIG.2a

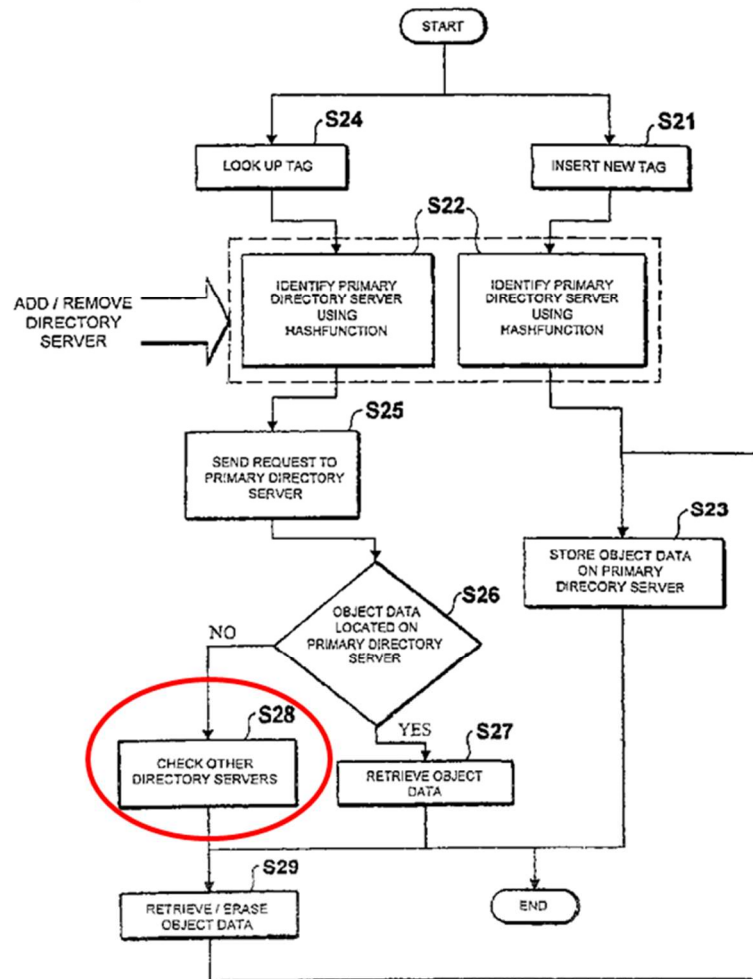


(*Id.*, FIG. 2a (annotated).)

73. However, if the primary directory server determines in step S26 that it does not contain the application server addresses related to the object tag, then the information request is forwarded to another directory server in step S28, where the identity of that server is determined based on the application of the hash function.

(*Id.*, 12:63-13:5.)

FIG.2a



(*Id.*, FIG. 2a (annotated).) Skagerwall discloses a flexible system where one of several different components (e.g., the access server or a directory server) may be the component that identifies which directory server should be sent the information request by executing the hash function. (*Id.*, 9:28-31, 10:25-27.) That is, one of several system components may be the component that identifies a “correct” directory server that a request should be forwarded to, when the request was

initially sent to the wrong directory server, where the correct directory server (the “at least one of the plurality of data location servers”) has the requested application server addresses associated with the object tag (“includes location information associated with the identifier string”). For example, Skagerwall discloses that each of the directory servers (“each one of the data location servers”) may identify the location of a correct directory server (“configured to determine the at least one of the plurality of data location servers”) by applying the hash function to the object tag (“based on the hash function applied to the identifier string”). (Ex. 1006, 9:18-31 (disclosing that after one of the directory servers receives a request including the object tag from an access server, the directory server “in turn, using the hashfunction and hashtable H, determines the directory server for storing the object tag T; T1-Tw and forwards the request to that particular directory server”), 10:22-11:16 (“[I]t is preferred, that one of the directory servers DS1; DS2; ...; DSn identifies the directory servers storing information related to a requested object tag.”).)

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

74. Skagerwall discloses this limitation. As discussed for limitation [1.b], Skagerwall discloses a hash function that is applied to an object tag, with the result

serving as an index to a table of directory servers (i.e., a hash table). (*See* Section IX.A.1.iii; Ex. 1006, 10:28-11:16, 13:15-57.) As further discussed, Skagerwall discloses that the object data (including application server addresses (“location information”)) on a given directory server represents only a portion of the full database of object data, and that the full database of object data is distributed across the plurality of directory servers according to a hash function/hash table that assigns object data to directory servers based on object tag. (*See* Section IX.A.1.iii; Ex. 1006, 8:15-27, 8:38-54, 8:55-59.) A person of ordinary skill in the art would have therefore understood that the object data on a given directory server comprises a portion of the full hash table (“comprises any portion of a hash table”) distributed over the plurality of directory servers because each directory server includes object data associated with a portion of the identifiers in the hash table. A person of ordinary skill in the art would have also understood that the object data on a given directory server represents a portion of the hash table generated by the hash function (“comprises any portion of a hash table generated with the hash function”) because the hash function controls the object tags assigned to each directory server, and thus controls the associated object data (e.g., location information) stored on each directory server.

3. **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.”**

75. Skagerwall discloses this limitation. As discussed for limitation [1.b], a directory server (“data location server”) receives a request for the application server addresses (“location information”) associated with a certain object tag and thus also associated with the application data packets that form the data representation (“data entit[y]”) of an object (“associated with a respective one of the data entities”). (See Section IX.A.1.iii.) As further discussed, in response to that request (“in response to a request for any location information associated with the respective one of the data entities”), a given directory server (“each one of the data location servers”) is configured to return all the associated application server addresses (“configured to return all location information associated with a respective one of the data entities”). (See Section IX.A.1.iii; *see also* Ex. 1006, 16:1-4 (“[L]ookup procedures regarding a plurality of application data packets can be performed by the directory server and ***all results*** will be returned to the unit, which issued the search request.”); *see also id.*, 9:34-38 (“The directory server...performs operations to determine ***all addresses*** of application servers IS;

IS1-ISp storing application data packets associated with the read object tag T; T1-Tw and thus object D.”.)

4. **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.”**

76. Skagerwall discloses this limitation. **First**, Skagerwall 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.” As discussed for limitation [1.b], Skagerwall discloses a hash function applied to an object tag (“string identifier”)⁸ that generates an index to the hash table, where the hash table associates the index (i.e., the hash of that object tag (“a hash of the string identifier”)) with a directory server (“at least one of the plurality of data location servers”). (See Section IX.A.1.iii.) Skagerwall discloses that the hash table specifically maps (“includes an association between”) the index (“a hash of the string identifier”) to a corresponding directory server’s address (“at least one identifier of the at least one of the plurality of data location servers”) of the

⁸ I assume the claimed “string identifier” is intended to refer to the “identifier string” in claim 1. (Section IX.A.1.ii.)

corresponding directory server. (Ex. 1006, 10:31-38 (discussing an example hash table where “addresses of available servers are stored as records in respective sequential entries of the hashtable,” such as records “indicated by A, B, C and D [which] stand for addresses of four directory servers DS; DS1-DSn”).)

77. **Second**, Skagerwall discloses “wherein the location information includes a hash table.” As discussed for claim 2, the “location information” on a given directory server “comprises any portion of a hash table,” and thus a person of ordinary skill in the art would have understood that all the “location information” collectively comprises 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.” In 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 about the locations of directory servers containing requested application server addresses, and by extension the locations of application data packets (“data”) since the application server addresses indicate locations of application data packets. Skagerwall also discloses that the hash 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 directory servers (i.e., location servers) because it maps a hash of an object tag to addresses (i.e., identities) of the directory servers. As further discussed above, the hash table also provides information pertaining to one or more application server addresses (locations of data) because it provides information about which directory servers store the application server addresses associated with a particular object tag.

B. Skagerwall Discloses and/or Suggests Claim 3

- 1. 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.”**

78. Skagerwall discloses and/or suggests this limitation. **First**, Skagerwall discloses “wherein the data repository comprises a plurality of data repository servers.” As discussed for limitation [1.a], the collection of application servers (together a “data repository”) that store application data packets (“data”) comprises multiple application servers (“wherein the data repository comprises a plurality of data repository servers”). (See Section IX.A.1.ii.)

79. **Second**, Skagerwall 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.” Skagerwall discloses a process for inserting new object data (e.g., application server addresses) relating to a new object into the database distributed across a plurality of directory servers. (Ex. 1006, 12:3-26, 14:29-53.) Skagerwall explains that the process involves the execution of a hash function to identify a target directory server for storing the new object data. (*Id.*, 12:16-26, 14-49-53.) Immediately following the description of inserting object data (e.g., application server addresses) pertaining to an object in response to a request from a user (*id.*, 14:29-53), Skagerwall explains that the above-described operation of executing a hash function to store object data may also be used when storing application data packets (“data”) on application servers (“data repository servers”). (*Id.*, 14:54-56 (“Although not described, the above operation may also include storing respective application data packets on application servers.”).)

80. I have been asked to consider a scenario in which Skagerwall does not disclose the specifics of the execution of a hash function in response to a request to store application data packets on application servers. In that scenario, a person of

ordinary skill in the art would have been motivated to implement this feature when, for example, the data corresponding to an object changes and thus additional data packets (“data”) associated with an object tag are stored on application servers.

Skagerwall discloses a system in which the data/information services (e.g., application data packets) for an object and associated object data (e.g., location information) are subject to change. (Ex. 1006, 2:30-40, 4:1-12, 8:15-19, 8:59-67, 11:16-26, 12:47-51.) A person of ordinary skill in the art would have understood that, when storing additional application data packets for an object, it would have been advantageous to store application data packets corresponding to the same object tag in similar locations (e.g., on the same application servers) wherever possible, to reduce communication overhead and processing requirements associated with retrieving application data packets. For example, if packets corresponding to the same object tag were stored on many different application servers, then retrieving packets associated with an object tag would require a greater number of communications between the access server and application servers. Reducing communication overhead and processing requirements is consistent with Skagerwall’s goals of conserving resources and reducing processing time, processing load on the access server, and application complexity. (Ex. 1006, 5:28-37, 10:15-20, 17:1-5, 18:54-58.)

81. Consistent with Skagerwall's process for identifying application servers for retrieving application data packets, as discussed in limitation [1.b], a person of ordinary skill in the art would have understood that identifying the application servers for storing application data packets would have likewise involved identifying/retrieving the application server addresses associated with an object tag. (See Section IX.A.1.iii; Ex. 1006, 9:18-48.) As discussed for limitation [1.b], that process for retrieving application server addresses involves a directory server executing the hash function to calculate the location of a correct directory server (e.g., if a request is initially sent to a wrong directory server). (See Section IX.A.1.iii.) Thus, a person of ordinary skill in the art would have been motivated to implement the Skagerwall system such that, in response to a request to store additional application data packets ("data") associated with an object tag on the application servers ("in response to a request to store data in the data repository"), each directory server would be configured ("each of the plurality of data location servers is configured") to execute the hash function ("to execute the hash function") to determine the directory servers that store application server addresses associated with that object tag. This would have allowed the system to store the application data packets on one of the application servers ("store the data on one of the plurality of data repository servers") based on the application server addresses

associated with the object tag at the particular directory server identified by the result of the hash function (“based on a result of the hash function”). Moreover, such an implementation would have been consistent with Skagerwall’s disclosure that storing application data packets on application servers may involve operations such as executing a hash function. (Ex. 1006, 14:54-56; *see also id.*, 2:30-40, 4:1-12, 8:15-19, 8:59-67, 11:16-26, 12:47-51.)

82. A person of ordinary skill in the art would have also had a reasonable expectation of success in implementing this feature because it would have involved modifying the Skagerwall system in a manner that Skagerwall suggests is feasible—implementing a method of storing application data packets on application servers in a manner similar to the described operation of a directory server executing a hash function to store object data. Moreover, such a modification would have involved applications of known technologies (e.g., known system for managing data stored in a distributed network (Skagerwall)) according to known methods (e.g., executing a hash function to store information (Skagerwall)) to yield the predictable result of a system including a directory server execute a hash function as part of the process to store application data packets, as discussed above. Accordingly, Skagerwall discloses and/or suggests this limitation.

C. Skagerwall in View of Vingralek Discloses and/or Suggests Claims 6-17

1. 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:”**

83. 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, as discussed for limitation [1.pre], Skagerwall discloses a communication system (“system”) that links real-world objects to information related to the objects. (Section IX.A.1.i.) Furthermore, as discussed for limitation [1.b], Skagerwall’s system performs processes for retrieving application data packets (“data”) associated with an object. (Section IX.A.1.iii.) These processes include intermediate steps in which a directory server receives a request (“location quer[y]”) from the access server for information about an object tag and, in response, returns application server addresses (“data location information”) that provide the location of the application data packets (“providing the data location information in response to location queries”). (Section IX.A.1.iii.) A person of ordinary skill in the art would have understood that retrieving information based on a request is a way of managing information, and thus a system that performs

processes that retrieve application server addresses (“data location information”) is a “system for managing data location information.”

- 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;”**

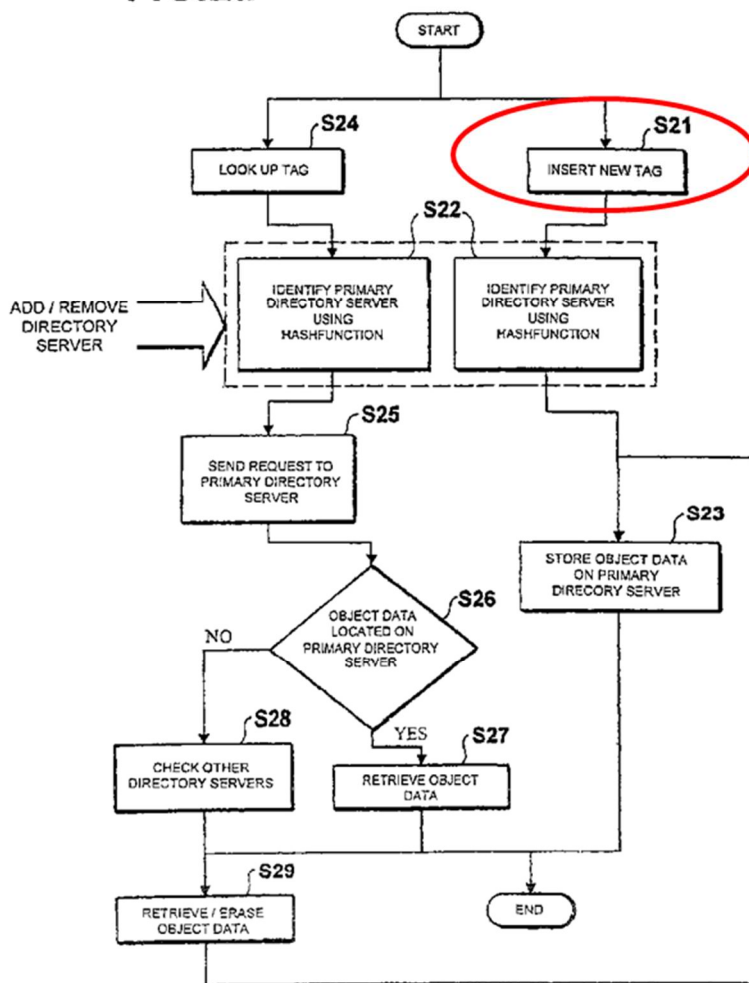
84. Skagerwall discloses and/or suggests this limitation. **First**, Skagerwall discloses “a location server configured to receive a location addition request.” As discussed for limitation [1.b], Skagerwall discloses a network of directory servers (each a “location server”)⁹ that each store application server addresses (“location[s]”) identifying the location of application data packets. With respect to Figure 2a, Skagerwall describes a process for inserting new object data into the database of object data distributed across the directory servers. (Ex. 1006, 12:3-4.) Skagerwall discloses that a request (“location addition request”) for

⁹ Skagerwall 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.)

insertion of new object data (e.g., application server addresses (“location[s]”)) corresponding to an object tag (“identifier”)¹⁰ is received by the system kernel in step S21. (*Id.*, 12:5-7; *see also id.*, 14:37-40 (describing a similar “request...to insert object data” with respect to Figure 3).)

¹⁰ Skagerwall 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.)

FIG.2a



(*Id.*, FIG. 2a (annotated).) In step S22, a target directory server (“location server”) for storing the object data is identified by applying a hash function to the object tag (“identifier”) and using the result as an index for searching the hash table. (*Id.*, 12:8-21; *see also id.*, 14:37-53 (describing similar identification of target directory server with respect to Figure 3).) Finally, in step S23, the request to store the object data is sent to the target directory server (“a location server configured to

receive a location addition request”), which stores the object data and fulfills the sequence of steps for inserting new object data. (*Id.*, 12:22-26.)

85. **Second**, Skagerwall discloses and/or suggests “the location addition request formatted in conformance with a transfer protocol.” A person of ordinary skill in the art would have found it beneficial to format communications sent between an access server and directory server according to a transfer protocol, e.g., hypertext transfer protocol (HTTP). Skagerwall discloses that its system may “use standard and/or amended protocols of a known datacom- or telecom network for retrieving information associated with an object.” (Ex. 1006, 9:59-62.)

Skagerwall also discloses that the system may be “used in combination with a known datacom- or telecom network, such as the Internet,” and that at least lookup procedures may be performed “via a HTTP request.” (*Id.*, 18:2-7, 18:24-26.) A person of ordinary skill in the art would have understood HTTP is a transfer protocol broadly governing communications between web clients and web servers, and its application is not limited to communications occurring during lookup procedures. Moreover, a person of ordinary skill in the art would have found it beneficial to use a transfer protocol compatible with Skagerwall’s system, e.g., HTTP, for not only the lookup processes, but also the processes involving object data insertion requests to enable the system to operate using a widely-used,

Internet-compatible, and well-known protocol to improve reliability. Thus, a person of ordinary skill in the art would have been motivated to format communications (e.g., requests) between an access server and directory server, including Skagerwall's disclosed object data insertion request ("location addition request"), in accordance with HTTP ("formatted in accordance with a transfer protocol").

86. A person of ordinary skill in the art would have also had a reasonable expectation of success in implementing this feature because it would have involved modifying the Skagerwall system in a manner that Skagerwall suggests is feasible—implementing the system in accordance with a standard protocol such as HTTP. Moreover, such a modification would have involved applications of known technologies (e.g., known communication system of Skagerwall) according to known methods (e.g., known transfer protocol used in Skagerwall or HTTP) to yield the predictable result of a system that sends requests between access server and directory server, including requests to add location information ("location addition request"), where the requests are formatted according to HTTP ("in conformance with a transfer protocol"), as discussed above.

87. **Third**, Skagerwall discloses and/or suggests "the location addition request comprising an identifier and at least one location to associate with the

identifier.” As discussed for limitation [1.b], the directory servers each store application server addresses in the form of a set of mappings, which map object tags (“identifier”) to all associated object data (e.g., all application server addresses associated with a specific object tag). (See Section IX.A.1.iii.) Thus, a person of ordinary skill in the art would have been motivated to modify Skagerwall’s system to implement the request (“location addition request”) for adding new object data (e.g., application server addresses (“location[s]”)) such that the request includes the object tag (“identifier”) itself and any associated object data to be added (“the location addition request comprising an identifier and at least one location to associate with the identifier”). A person of ordinary skill in the art would have found it beneficial to implement this feature because aligning the insertion and storage of object data would have allowed for simplified storage of the object tag/object data mappings, without requiring additional steps (beyond the insertion request) to associate the object tag and object data on the directory server.

88. A person of ordinary skill in the art would have had a reasonable expectation of success in implementing this feature in Skagerwall’s system, as it would have involved modifying the Skagerwall system in a manner that Skagerwall suggests is feasible—aligning the processes for insertion and storage of new object data. Indeed, Skagerwall discloses storing the object data on the

directory servers as part of the insertion process (Ex. 1006, 12:1-26), and also discloses storing the object data alongside the associated object tag (*id.*, 8:43-63, 15:15-20, 15:49-52). Moreover, such a combination would have involved applications of known technologies (e.g., known systems for managing data and associated location information (Skagerwall)) according to known methods (e.g., known methods of inserting and storing location information (Skagerwall)) to yield the predictable result of a system in which object data insertion requests include both the object tag and associated object data.

89. **Fourth**, Skagerwall 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.” As discussed for limitation [1.a], Skagerwall discloses that an object tag (“identifier”) identifies a real-world object (“entity”). (*See* Section IX.A.1.ii.) And as discussed for limitation [1.pre], Skagerwall discloses that each of the application server addresses (“each of the at least one location”) specifies the location of application data packets (“data”) stored on a network of application servers (“in a network”) that are associated with the object (“pertaining to the entity”) identified by the object tag. (Section IX.A.1.i.)

90. Skagerwall 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.) Skagerwall discloses that its system is for managing information associated with real-world objects, which may be “any products or items of the physical world” (i.e., a thing that is real). (Ex. 1006, 6:45-48.) As discussed for limitation [1.a], the application data packets (“data”) are associated with that real-world object and form the data representation of that object, but are merely information stored on application servers and thus are distinct from the real-world object itself. (*See* Section IX.A.1.ii.)

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

91. Skagerwall discloses this limitation. As discussed for limitation [1.b], each of Skagerwall’s directory servers (“location server”) includes a CPU (“processor”). (*See* 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.”**

92. Skagerwall in view of Vingralek discloses and/or suggests this limitation. **First**, Skagerwall discloses “programming logic stored on the location server” that is configured to perform any of the actions a “location server” may perform. As discussed for limitations [1.b] and [6.b], each of Skagerwall’s directory servers (“location server”) has a CPU that controls the operations of the server. (*See* Sections IX.A.1.iii, IX.C.1.iii.) A person of ordinary skill in the art would have understood that a CPU would control the operations of the directory server by being programmed to perform each of the server’s operations. Thus, each directory server would store programming logic (“programming logic stored on the location server”) to perform each of the operations of the directory server, including those recited in limitation [6.c] and discussed below. (*Id.*)

93. **Second**, Skagerwall discloses and/or suggests “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.” As discussed for limitation [1.b], the primary directory server (“location server”) receives a request (“location query related to a desired entity”) from the access server for information concerning the location of application data packets related to an object (“entity”), where that request includes the corresponding object tag. (*See* Section IX.A.1.iii.) The primary directory server then determines whether the application server addresses (“location information”) associated with the object/object tag (“for the desired entity”) are located at the primary directory server. (*Id.*) If the directory server (“location server”) does contain application server addresses (“location information”) associated with that object tag (“for the desired entity”), those addresses are retrieved and subsequent steps for providing the user with the desired information are performed. (*Id.*) For example, Skagerwall discloses that the obtained addresses (“location information”; “at least one location”) may be sent to the access server (“return...a location message”),

which the access server may use to determine where to find the application data packets associated with an object tag (“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...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”). (Ex. 1006, 9:42-48 (“The obtained addresses IS; IS1-ISp and optionally other information can be returned to the access server AS; AS1-ASr, which...retrieves the specified application data packets from the identified application servers”).) A person of ordinary skill in the art would have understood the location information is returned in a message because Skagerwall discloses that other communications from directory server to access server are “message[s].” (*Id.*, 13:5-6, 13:64-65.) A person of ordinary skill in the art would have been motivated to format the message returned to the access server containing the application server addresses (“location message”) so that it would have been in conformance with HTTP (“in conformance with the transfer protocol”) because as discussed for limitation [6.a], a person of ordinary skill in the art would have been motivated to format communications between an access server

and directory server, e.g., the location message discussed above, in accordance with HTTP. (*See* Section IX.C.1.ii.)

94. **Third**, Skagerwall discloses that “if the location server lacks the location information for the desired entity,” a directory server (“location server”) is configured to determine “a list of at least one other location server known to have the location information for the desired entity.” As discussed for limitation [1.b] with respect to Figure 2a, a primary directory server is determined in step S22 based on the application of a hash function to an object tag, with the result used as an index for searching in the hash table and finding the corresponding directory server. (Section IX.A.1.iii.) As further discussed for limitation [1.b], if in step S26 the primary directory server (“location server”) determines that it does contain the requested application server addresses associated with an object/object tag (“location information for the desired entity”), then that object data is retrieved by the access server in step S27. (Section IX.A.1.iii.) However, if the primary directory server determines in step S26 that it does not contain the application server addresses related to the object/object tag (“if the location server lacks the location information for the desired entity”), then the information request is forwarded to another directory server in step S28, which is determined based on the application of the hash function. (Ex. 1006, 12:63-13:5.)

95. Skagerwall discloses that the primary directory server (“location server”) may be the component that determines the location of a “correct” directory server (i.e., the directory server that has the requested application server addresses associated with the object tag) (“a list of at least one other location server known to have the location information for the desired entity”) using the hash function. (*Id.*, 9:18-28 (disclosing that the access server “send[s] the request including the object tag T; T1-Tw to one of the directory servers DS1; DS2;...; DS_n, which in turn, using the hashfunction and hashtable H, determines the directory server for storing the object tag T; T1-Tw and forwards the request to that particular directory server”), 10:22-25 (“[I]t is preferred, that one of the directory servers DS1; DS2;...; DS_n identifies the directory servers storing information related to a requested object tag.”).) Skagerwall discloses that there may be multiple directory servers storing the application server addresses associated with the object tag (“a list of at least one other location server known to have the location information for the desired entity”) because directory servers may be replicated so as to provide a robust system. (*Id.*, 5:47-49, 11:60-64, 16:36-38.)

96. Thus, Skagerwall discloses that “if the location server lacks the location information for the desired entity,” a directory server (“location server”) is

configured to determine “a list of at least one other location server known to have the location information for the desired entity.”

97. I have been asked to consider as scenario where Skagerwall does not explicitly disclose “**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.**” In that scenario, a person of ordinary skill in the art would have been motivated to implement this feature in view of Vingralek.

98. Like Skagerwall, 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 that are 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 Skagerwall, which discloses

that the database of object data is distributed based on a hash function over a plurality of directory servers, so that the number of objects may be scaled by adding new directory servers. (Ex. 1006, 8:15-19, 8:43-54; *see also* 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 Skagerwall's system/method.

99. Vingralek additionally discloses propagating updates to clients' and other servers' address tables in a deferred manner. (Ex. 1007, 6.) To limit communication overhead, clients learn about changes to the address table only when a client request reaches a wrong server, in which case the server forwards the client's request to the correct server and provides updates to the client's address table. (*Id.*, 5 ("To limit the communication overhead, clients learn about bucket splits and changes to the server-locating hash function only when a client request is directed to the 'wrong' server," in which 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."), 6 ("Similarly to [the] LH* [approach], updates to the address table are propagated to clients and other servers in a deferred manner; forwarding is used to locate buckets when a client has a stale address table.")) Thus, Vingralek discloses 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.

100. Vingralek's method of deferred address table updates operates as follows. A hash function is applied to a key, which is used as an index to an address table listing available servers. (*Id.*, 9.) Each client and server has its own copy of the address table, but each table may have stale information if changes have been made to the servers since the table was last updated. (*Id.*, 10 ("The client's and the server's perception of the file may or may not coincide with the current picture of the file (i.e., they may have stale information on...the address table)."); *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 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 (referring to "client requests and server responses" as "messages").)

101. 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 at least 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 containing the requested information) would have gradually brought the client's perception of the address table up-to-date. Thus, 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.

102. Additionally, Vingralek discloses that “[t]he most salient feature of [the LH*] approach,” where a client’s hash function and table are 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.” (*Id.*, 5.) Thus, Vingralek discloses implementing a redirection mechanism similar to that of the LH* approach and additionally includes improved features to minimize the number of servers and reduce the cost of scaling up a distributed system. (*Id.*, 6, 8.)

103. A person of ordinary skill in the art would have been motivated to include features similar to those disclosed by Vingralek when implementing Skagerwall by providing a deferred address/hash table update to the client to limit communication overhead associated with updating hash tables in a scalable system. Indeed, Skagerwall itself explains that, while an access server’s hash table is not always up to date, it is preferably not updated every time there is a change to the hash table, so as to conserve resources. (Ex. 1006, 10:12-20.) As discussed above, Vingralek’s method of updating the client’s address table only when a request reaches the wrong server (i.e., in a deferred manner) is beneficial because it limits communication overhead, and improves the scalability of the network because communication overhead is independent of number of servers and clients in the

system. (Ex. 1007, 5.) Thus, a person of ordinary skill in the art would have found it beneficial to include Vingralek's method of deferred hash table updates, via a redirect message, when implementing Skagerwall's system/process.

104. A person of ordinary skill in the art would have further understood that it would also be beneficial for Vingralek's method of updating the client's address table via redirect message to provide the addresses of "correct" location servers to an access server. Skagerwall discloses that "[i]ntermediate steps are possible, where the look up request is relayed to further devices before it reaches the correct directory server." (Ex. 1006, 9:39-41.) Indeed, Skagerwall discloses an arrangement in which the first directory server ("location server") calculates the location of a correct directory server ("a list of at least one other location server known to have the location information for the desired entity") and the access server forwards the request to the correct directory server, but may involve an extra set of servers (referred to as broker servers), and does not explicitly disclose a specific process for transmitting the address of the correct directory server to the access server. (*Id.*, 16:47-51, 18:33-43.) Consistent with Vingralek's disclosure of updating the client's address table, a person of ordinary skill in the art would have been motivated to implement a system similar to that of Skagerwall where it includes in the redirect message only the portion of the address table corresponding

to the location server(s) that contain the requested location information (“redirect message comprising a list of at least one other location server known to have the location information for the desired entity”). This would allow the access server to update the relevant portions of its own hash/address table, consistent with Vingralek’s teachings regarding the benefits of a deferred address table update, as well as provide the access server with the addresses of the location server(s) that should receive the forwarded request. A person of ordinary skill in the art would have also understood that implementing Vingralek’s redirect message feature would have reduced communication bandwidth in several ways, consistent with Vingralek’s teachings of limiting communication overhead: by directly conveying the address of a correct directory server to the access server (without involving of an extra set of broker servers), and by including in the redirect message only the portion of the address table corresponding to the location server(s) that contain the requested location information, instead of the entire mapping/table. 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 directory server). (Ex. 1007, 5, 7, 8; Ex. 1006, 5:28-37, 10:15-20, 17:1-5, 18:54-58.)

105. A person of ordinary skill in the art would have understood that this implementation, where a first directory server calculates a correct directory server and the access server forwards the request to that server (e.g. as disclosed by Skagerwall), would have been one of only several options, i.e., calculating and forwarding at either the access server or the directory server that processes the information request. Thus, a person of ordinary skill in the art would have been motivated to implement the system to calculate the addresses of correct directory server(s) at a directory server and transmit a redirect message, including those addresses, to the access server for the access server to forward the request to that correct directory server. As such, a person of ordinary skill in the art would have been motivated to include features similar to Vingralek's "redirect message" features when implementing Skagerwall.

106. A person of ordinary skill in the art would have had a reasonable expectation of success combining the teachings of Skagerwall and Vingralek. As discussed above, both references are in the same field of art. (*See above.*) Furthermore, Skagerwall discloses an "error message" that is provided if none of the directory servers stores the requested object data (Ex. 1006, 13:5-6, 13:62-65), akin to the claimed "redirect message" that is provided if the location server does not contain the requested information. In fact, a person of ordinary skill in the art

would have appreciated that the concept of providing a message when requested information is not found was well known. (*Id.*, 2:40-43 (“[A] called address or link for retrieving a certain service from the network may no longer be valid and an error message is returned to the user.”).) Moreover, Skagerwall itself suggests that its system/process is flexible and additional processes may be included, particularly with respect to the process of relaying a lookup request to a correct directory server. (*Id.*, 9:39-41 (explaining that “[i]ntermediate steps” may be implemented when relaying a lookup request to a correct directory server).)

107. Additionally, a person of ordinary skill in the art would have understood that implementing the combination would have involved modifying the Skagerwall system in a manner that Skagerwall suggests is feasible—implementing a means of occasionally updating a hash table to reduce communication overhead, and implementing a means of providing addresses of correct directory servers to the access server. (*Id.*, 10:12-20.) Moreover, such a modification would have involved applications of known technologies (e.g., known system of managing data and associated location information (Skagerwall)) according to known methods (e.g., known methods of occasionally updating 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, Skagerwall in view of Vingralek discloses and/or suggests this limitation.

108. Consistent with the above and in such a combined system, for example, the access server would apply the hash function to the object tag and its own copy of the hash table, and send a request to the primary directory server (“location server”), which would apply the hash function to the object tag and its own copy of the hash table in order to determine whether it has the requested location information. If the primary directory server does not have the requested location information, e.g., application server addresses associated with an object tag (“if the location server lacks the location information for the desired entity”), the directory server would calculate the location of another directory server (perhaps multiple, if servers are replicated for robustness, as described above) that has the requested location information (“at least one other location server known to have the location information for the desired entity”). The directory server would then send a portion of the hash table corresponding to those directory servers in a redirect message (“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 hash table) would allow the access server to update its stale hash table, and additionally provide the access server with the addresses of correct directory servers, to which the access server then forwards the request.

2. **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.”**

109. The Skagerwall-Vingralek combination discloses and/or suggests this limitation. Skagerwall discloses that a hash table maps directory servers in a network (“plurality of location servers in the network”) to a hashed object tag (i.e., the output of a hash function applied to an object tag). (Ex. 1006, 4:66-5:10, 8:38-67, 10:27-11:11, FIG. 2.) As discussed for limitation [1.b] with respect to the location information retrieval process, the hash table indicates which directory servers store location information associated with a given object tag. (*See* Section IX.A.1.iii.) As discussed for limitations [1.pre] and [1.a], object tags 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 directory servers in the network (“a list of a plurality of location servers in the network”), along with a corresponding list of hashed object tags (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”). (*See* Section IX.A.1.iii.) 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 access server that contains a portion of the hash table corresponding to the “correct” directory servers that include the requested application server addresses (“location information”). (*See* Section IX.C.1.iv.) As further discussed for limitation [6.c], the directory servers may be replicated for robustness, such that there would be multiple “correct” directory servers containing the requested information. (*See* Section IX.C.1.iv.) 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 the hash table (“the list of at least one other location server”) also comprises a list of the correct directory servers (“a list of a plurality of location servers in the network”) and a corresponding list of hashed object tags 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”).

3. **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.”**

110. Skagerwall discloses this limitation. As discussed for limitation [1.b], Skagerwall discloses that each of the directory servers in the network (“each of a plurality of location servers in the network”) stores only a portion of the application server addresses (“stores only a portion of the data location information”). (*See* Section IX.A.1.iii.)

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

111. Skagerwall discloses this limitation. Skagerwall discloses that each directory server (“location server) has a memory means (“location store”) that stores application server addresses (“location information”) along with corresponding object tags. (Ex. 1006, 15:15-20 (“[D]irectory server DS1...further comprises address memory means AM for storing a plurality of address data blocks AB1-AB_k, each comprising one of the application identifiers AI1-AI_m, the corresponding application server address IS; IS1-IS_p and, optionally, corresponding object tags T; T1-T_w.”), 15:49-51 (“[T]he directory server may comprise single memory means, in which case all object data related to an objected are stored in the provided memory means.”).) A person of ordinary skill in the art would have understood that the memory means is an “indexed location store”

because each object tag may be used as an index to retrieve all associated application server addresses (“location information”).

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

112. Skagerwall discloses this limitation. As discussed for limitation [1.a], a person of ordinary skill in the art would have understood that each of Skagerwall’s object tags is a string. (*See* Section IX.A.1.ii.) A person of ordinary skill in the art would have further understood that Skagerwall’s application server addresses are also strings defining the locations of the data location server (e.g., via URLs), and thus the memory means (“indexed location store”) for storing object tags and application server addresses would have been understood to comprise a “string store.” (*Id.*) Second, Skagerwall discloses that the memory means (“indexed location store comprises a string store”) is “indexed by a hash table.” As discussed for claim 9, the memory store (“indexed location store”) uses an object tag as an index to retrieve all associated application server addresses. (*See* Section IX.C.4.) Skagerwall discloses that the indices (i.e., the object tags) included in a directory server’s memory means are arranged/distributed based on a hash function/hash table (“indexed by a hash table”) that maps object tags to one or more directory servers. (Ex. 1006, 8:43-54 (“[T]he hashfunction is used to assign

object data related to a group of objects to each of the directory servers. Thus, each of the object tags is assigned to a target directory server using the hashfunction. Preferably, the hashfunc[tion] distributes the object tags evenly over the available directory servers.”.)

6. 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.”

113. Skagerwall discloses this limitation. As discussed for claim 10, the memory means is an “indexed location store [that] comprises a string store indexed by a hash table.” (See Section IX.C.5.) Skagerwall discloses a system that may have multiple access servers (“a plurality of clients”). (Ex. 1006, 9:14-17.) A person of ordinary skill in the art would have understood that the hash table is distributed across the plurality of access servers (“hash table distributed across a plurality of clients”) in at least two ways. First, the hash table is distributed across the plurality of access servers because Skagerwall discloses that the hash table may be stored on each access server. (*Id.*, 10:12-14 (“[C]opies of the hashtable can be stored in the at least one access server.”).) Second, the hash table is distributed across the plurality of access servers because each access server is associated with a subgroup of objects, and thus a portion of the object tags represented in the hash table. (*Id.*, 9:14-17 (“[A] plurality of arbitrarily located access servers AS; AS1-

ASr may be provided, each serving a group of users, e.g.,...serving a subgroup of the plurality of objects.”.)

114. Skagerwall 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.*) In particular, a person of ordinary skill in the art would have recognized that the access server (“client”) is a hardware and/or software component attached to a network given that Skagerwall explains that all components in the system may be “connected via a network N for communication with each other.” (*Ex. 1006, 16:34-36.*) As further discussed for limitation [1.b], the access server also sends a request (i.e., location request message) to a directory server (i.e., location server) to initiate lookup of mappings of object tags (i.e., identifiers) to application server addresses (which provide location information about application data packets). (*See Section IX.A.1.iii.*)

7. 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.”

115. Skagerwall discloses this limitation. As discussed for claim 10, the memory means is an “indexed location store [that] comprises a string store indexed

by a hash table.” (See Section IX.C.5.) A person of ordinary skill in the art would have understood that the hash table is distributed across the plurality of directory servers (“hash table distributed across a plurality of servers”) in at least two ways. First, the hash table is distributed across the plurality of directory servers because Skagerwall discloses that the hash table may be stored on each directory server. (Ex. 1006, 10:9-12 (“Copies of the hashtable may additionally be stored in each of the directory servers for performing look up operations upon receiving a request regarding the correct directory server.”).) Second, the hash table is distributed across the plurality of directory servers because the hash function distributes the object tags in the hash table across the directory servers in order to assign related object data to those servers. (*Id.*, 8:43-54 (“[T]he hashfunction is used to assign object data related to a group of objects to each of the directory servers. Thus, each of the object tags is assigned to a target directory server using the hashfunction. Preferably, the hashfunc[tion] distributes the object tags evenly over the available directory servers.”).)

8. **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.”**

116. The Skagerwall-Vingralek combination discloses and/or suggests this limitation. **First**, Skagerwall discloses “wherein the location query identifying the desired entity comprises a unique identifier for the desired entity.” As discussed for limitations [1.b] and [6.c], Skagerwall discloses that the request for location information (“location query”) sent from access server to directory server identifies the object for which information is sought (“identifying the desired entity”) by including the object tag (“identifier”) that identifies the object (“for the desired entity”) in the request. (See Sections IX.A.1.iii, IX.C.1.iv.) As discussed for limitation [1.a], the object tag uniquely identifies an object (“a unique identifier for the desired entity.” (See Section IX.A.1.ii.)

117. **Second**, the Skagerwall-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],

Skagerwall discloses programming logic on a directory server (“location server”) that is configured to perform any of the directory server’s operations. (*See* Section IX.C.1.iv.) As further discussed, in the Skagerwall-Vingralek combination, the directory server (“location server”) is configured to apply a hash function (“index function”) to the object tag (“unique identifier”) as part of the process for retrieving the application server addresses (“at least a portion of the locations”) associated with the object tag (“unique identifier”) that are stored in the memory means (“indexed location store”) of the directory 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.C.1.iv.) A person of ordinary skill in the art would have recognized that an index function is a function that returns the index to a table, and thus a hash function, which returns a value that is used as an index to the hash table, is an “index function.”

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

118. Skagerwall discloses this limitation. As discussed for claim 10, a directory server’s memory means (“indexed location store”) comprises a “string store.” (*See* Section IX.C.5.) As discussed for claim 13, the index function

applied to the object tag is a hash function (“index function comprises a hash function”). (*See* Section IX.C.9.)

10. 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:”**

119. 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, as discussed for limitation [1.pre], Skagerwall discloses a communication system that links real-world objects to information related to the objects. (*See* Section IX.A.1.i.) As discussed for limitation [1.b], Skagerwall discloses that the system includes a network of multiple directory servers (“network comprising a plurality of location servers”)¹¹ that store application server addresses (“data location information”) identifying the location of application data packets (“data”). (*See* Section IX.A.1.iii.) As further discussed, Skagerwall discloses that the system performs a method of handling requests for

¹¹ Skagerwall 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.)

application server addresses (“location queries”) pertaining to a specific object tag, where the request for that location information is sent to a directory server (“location queries in a network”). (See Section IX.A.1.iii.)

- 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;”**

120. Skagerwall discloses this limitation. **First**, Skagerwall discloses “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.” As discussed for limitation [1.b], Skagerwall discloses a network of directory servers (“the network”), and discloses that each directory server maintains a set of identifier/location mappings, where each object tag (“identifier”)¹² is associated with corresponding application server addresses (“locations”) in the server (“correlating each one of a plurality of locations in the network”). (See Section IX.A.iii.) As discussed for limitation

¹² Skagerwall 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.)

[1.a], Skagerwall discloses that each object tag (“each one of the plurality of identifiers”) identifies an object’s application data packets (“identifying a respective one of a plurality of data entities”), where application packets associated with multiple objects (“plurality of data entities”) are stored on a network of application servers. (*See* Section IX.A.1.ii.)

121. **Second**, Skagerwall discloses “wherein the data entities are stored in corresponding locations in the network.” As discussed for limitation [1.b] and above, the application data packets (“data entities”) are stored at locations on the application servers, which are identified by application server addresses that correspond to an object tag (“wherein the data entities are stored in corresponding locations” in a network). (*See* Section IX.A.iii.) Skagerwall further discloses that these application servers may in the same network (e.g., a known network or the Internet) as the directory servers (“in the network”). For example, Skagerwall discloses that both the directory servers and application servers may both communicate via the Internet (i.e., on the same network). (Ex. 1006, 6:40-41 (“application servers may be part of a known datacom-or telecom network, such as the Internet”), 17:9-12 (system kernel’s components, which include directory servers, “can be connected by telecommunication networks or by existing computer networks, such as the Internet”).) Alternatively, Skagerwall also

discloses that the entire interconnected system of application servers and directory servers may communicate over some other known network. (*Id.*, 22:45-48 (“the system is communicating over a known datacom- or telecom network and connect[] *services* provided by the known network”), 4:1-5 (referring to information stored on application servers as “information *services*”).)

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;”

122. Skagerwall discloses this limitation. As discussed for limitation [1.b], the access server (“client”) transmits the information request (“location query”) to a directory server (“receiving a location query from a client at one of the plurality of location servers”). (*See* Section IX.A.1.iii.) As further discussed, the request (“location query”) is for application server addresses (“location information”) associated with the object tag in the request (“location query requesting location information”), and the application server addresses identify the locations of the application data packets associated with the object tag (“identifying a location of a data entity included in the data entities”). (*Id.*) As discussed for limitations [1.a] and [15.a], the application data packets associated with a given object tag form a “data entity,” and the group of all packets, which are associated with many

different tags, thus represent multiple “data entities.” (See Sections IX.A.1.ii, IX.C.10.ii.)

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

123. Skagerwall discloses this limitation. As discussed for limitation [6.c], Skagerwall discloses that if, for example, the wrong directory server (“location server”) receives a request (i.e., if the directory server does not have the requested application server addresses (“location information”)), the directory server may determine which other directory server has that requested information (“determining which of the plurality of location servers includes the location information”). (See Section IX.C.1.iv.)

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”

124. Skagerwall discloses this limitation. As discussed for limitation [6.c], Skagerwall discloses that if a directory server (“location server”) has the requested application server addresses (“in response to determining the one of the plurality of location servers includes the location information”), the directory server returns a message to the access server (“sending a location response message to the client”)

with the requested application server addresses (“the location response message comprising the location information”). (*See* Section IX.C.1.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.”**

125. The Skagerwall-Vingralek combination discloses and/or suggests this limitation. As discussed for limitation [6.c], the Skagerwall-Vingralek combination discloses and/or suggests that if a directory server (“location server”) does not have the application server addresses (“location information”) that were requested (“in response to determining the one of the plurality of location servers fails to include the location information”), the directory server returns a message to the access server (“sending a redirect message to the client”) that identifies which of the other directory servers includes the requested application server addresses (“the redirect message identifying which of the plurality of location servers includes the location information”). (*See* Section IX.C.1.iv.)

11. 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.”

126. Skagerwall discloses this limitation. As discussed for claim 13, a person of ordinary skill in the art would have understood that Skagerwall’s disclosed hash function is an index function (“indexing function”). (*See* Section IX.C.8.) As discussed for limitation [15.a], Skagerwall’s disclosed object tag (“identifier”) identifies the application data packets (“data entity”) associated with an object. (*See* Section IX.C.10.ii.) And as discussed for limitation [6.c], Skagerwall discloses that the directory server (“location server”) applies a hash function (“indexing function”), using an object tag (“identifier”) as an input (“applying an identifier of the data entity to an indexing function”), in order to determine which of the other directory servers includes the requested application server addresses (“to determine which of the location servers includes the location information”). (*See* Section IX.C.1.iv.)

12. **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.”**

127. Skagerwall discloses this limitation. **First**, Skagerwall discloses “wherein each one of the plurality of location servers comprises a portion of an indexed table of the identifiers and corresponding associated locations.” Skagerwall describes a hash function that is applied to an object tag (“identifier”) to generate an index to a hash table, where the hash table maps the object tags (“identifiers”) to corresponding addresses of directory servers (“corresponding associated locations”) that store information pertaining to a certain object tag. (Ex. 1006, 10:21-11:15, 13:15-57.) A person of ordinary skill in the art would have recognized that the hash table is therefore “an indexed table of the identifiers and corresponding associated locations.” As discussed for claim 2, Skagerwall discloses that each directory server (“each one of the plurality of location servers”) stores location information (e.g., application server addresses) that comprise a portion of a hash table. (See Section IX.A.2.) Thus, a person of ordinary skill in the art would have recognized that each directory 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.

128. **Second**, Skagerwall 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], Skagerwall discloses determining the location of a directory server that stores the requested application server addresses (“determining which of the plurality of location servers includes the location information”) by using a hash function that takes an object tag 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 directory server addresses (“to look up corresponding associated locations”) in the hash table (“in the indexed table”). As discussed for limitation [1.a], Skagerwall discloses that an object tag (“identifier”) identifies the corresponding application data packets (“data entity”) associated with an object. (*See* Section IX.A.1.ii.) Thus, a person of ordinary skill in the art would have

understood that an object tag in a location request (“the one of the identifiers”) identifies the application data packets that will ultimately be retrieved (“identifying the data entity”).

II. CONCLUSION

129. 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