

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

**PETITION FOR *INTER PARTES* REVIEW
OF U.S. PATENT NO. 7,814,170**

TABLE OF CONTENTS

I.	INTRODUCTION	1
II.	MANDATORY NOTICES	1
III.	PAYMENT OF FEES UNDER 37 C.F.R. §§ 42.15 AND 42.103	3
IV.	GROUND FOR STANDING.....	3
V.	PRECISE RELIEF REQUESTED AND GROUNDS RAISED.....	3
	1. Earliest Effective Filing Date of the '170 Patent.....	4
	2. Skagerwall and Vingralek Qualify as Prior Art.....	6
VI.	LEVEL OF ORDINARY SKILL	7
VII.	OVERVIEW OF THE '170 PATENT	7
VIII.	CLAIM CONSTRUCTION	10
IX.	DETAILED EXPLANATION OF GROUNDS.....	14
	A. Ground 1: Skagerwall Anticipates Claims 1, 2, 4, and 5	14
	1. Claim 1	14
	2. Claim 2	31
	3. Claim 4.....	32
	4. Claim 5	33
	B. Ground 2: Skagerwall Renders Claim 3 Obvious.....	35
	1. Claim 3	35
	C. Ground 3: Skagerwall in View of Vingralek Renders Claims 6- 17 Obvious	39
	1. Claim 6.....	39
	2. Claim 7	59
	3. Claim 8.....	60
	4. Claim 9.....	60
	5. Claim 10.....	61
	6. Claim 11.....	62
	7. Claim 12.....	63

8.	Claim 13	63
9.	Claim 14	65
10.	Claim 15	65
11.	Claim 16	69
12.	Claim 17	70
X.	DISCRETIONARY DENIAL IS NOT APPROPRIATE	72
A.	The Board Should Not Deny Institution Under § 325	72
B.	Discretionary Denial Under <i>General Plastic</i> Is Not Appropriate.....	74
C.	The Board Should Not Deny Institution Under <i>Fintiv</i>	75
XI.	CONCLUSION.....	77
XII.	APPENDIX A (CLAIM LISTING)	78
	Claim 1	78
	Claim 2	79
	Claim 3	79
	Claim 4	79
	Claim 5:	79
	Claim 6	80
	Claim 7	81
	Claim 8	81
	Claim 9	81
	Claim 10	82
	Claim 11	82
	Claim 12	82
	Claim 13	82
	Claim 14	83
	Claim 15	83
	Claim 16	84
	Claim 17	84

LIST OF EXHIBITS

EX1001	U.S. Patent No. 7,814,170 to Overton et al. (“the ’170 patent”)
EX1002	Declaration of Sandeep Chatterjee, Ph.D.
EX1003	Curriculum Vitae of Sandeep Chatterjee, Ph.D.
EX1004	Prosecution History of U.S. Patent No. 7,814,170
EX1005	WIPO Application No. 1997/043717 to Kahn et al.
EX1006	U.S. Patent No. 6,473,781 to Skagerwall et al.
EX1007	R. Vingralek, <i>Distributed file organization with scalable cost/performance</i> (1993)
EX1008- EX1010	RESERVED
EX1011	Japanese Patent Application Publication No. H09-231180 to Sato et al.
EX1012- EX1014	RESERVED
EX1015	M. C. Norrie et al., <i>The Database Research Group at ETH Zurich</i> , SIGMOD Record (March 1994)
EX1016- EX1019	RESERVED
EX1020	Declaration of Sylvia D. Hall-Ellis, Ph.D.
EX1021	U.S. Patent Application No. 09/111,896
EX1022	<i>Amazon Web Services, Inc. v. Kove IO, Inc.</i> , IPR2020-00276, Paper 23 (June 16, 2020)
EX1023	Kove’s Memorandum in Opposition to Amazon Web Services, Inc.’s Motion for Judgment on the Pleadings (Dkt. Nos. 145 and

Petition for *Inter Partes* Review
Patent No. 7,814,170

	145-1) in <i>Kove IO, Inc. v. Amazon Web Services, Inc.</i> , Case No. 1:18-cv-08175 (N.D. Ill. May 29, 2020)
EX1024	Memorandum Opinion and Order (Dkt. No. 484) in <i>Kove IO, Inc. v. Amazon Web Services, Inc.</i> , Case No. 1:18-cv-08175 (N.D. Ill. Dec. 17, 2021)
EX1025	Memorandum Opinion and Order (Dkt. No. 739) in <i>Kove IO, Inc. v. Amazon Web Services, Inc.</i> , Case No. 1:18-cv-08175 (N.D. Ill. Feb. 6, 2024)
EX1026- EX1032	RESERVED
EX1033	2020 to 2023 Northern District of Illinois Orders Regarding Motions to Stay
EX1034	Agreed Order Setting Amended Patent Case Schedule (Dkt. No. 49) in <i>Kove IO, Inc. v. Google LLC</i> , Case No. 1:23-cv-04244 (N.D. Ill. Feb. 20, 2024)
EX1035	Federal Court Management Statistics (December 2023)
EX1036	Memorandum by Director Vidal (June 21, 2022)
EX1037	Request for <i>Ex Parte</i> Reexamination of U.S. Patent No. 7,814,170 in <i>Ex Parte</i> Reexamination No. 90/019,035 (Nov. 19, 2021)
EX1038	Order Granting Request for <i>Ex Parte</i> Reexamination in <i>Ex Parte</i> Reexamination No. 90/019,035 (Jan. 12, 2022)
EX1039	Notice of Intent to Issue <i>Ex Parte</i> Reexamination Certificate in <i>Ex Parte</i> Reexamination No. 90/019,035 (Sept. 27, 2022)
EX1040	Request for <i>Ex Parte</i> Reexamination of U.S. Patent No. 7,814,170 in <i>Ex Parte</i> Reexamination No. 90/019,165 (Feb. 17, 2023)
EX1041	Order Granting Request for <i>Ex Parte</i> Reexamination in <i>Ex Parte</i> Reexamination No. 90/019,165 (Mar. 29, 2023)
EX1042- EX1051	RESERVED

EX1052	Order for Staying Discovery Process until July 15, 2024 (Dkt. No. 65) in <i>Kove IO, Inc. v. Google LLC</i> , Case No. 1:23-cv-04244 (N.D. Ill. May 1, 2024)
EX1053	Northern District of Illinois Local Patent Rules
EX1054	RESERVED
EX1055	Google’s Motion to Stay and for Expedited Briefing Schedule (Dkt. No. 64) in <i>Kove IO, Inc. v. Google LLC</i> , Case No. 1:23-cv-04244 (N.D. Ill. Apr. 30, 2024)
EX1056	Joseph B. Greene, <i>Oracle8 Server Unleashed</i> (1998)
EX1057	Joseph B. Greene, <i>Oracle DBA Survival Guide</i> (1995)
EX1058	Betty Salzberg, <i>File Structures</i> (1988)
EX1059	Andrew P. Black & Yeshayahu Artsy, <i>Implementing Location Independent Invocation</i> (1989)

I. INTRODUCTION

Google LLC (“Petitioner”) requests *inter partes* review of claims 1-17 (“the challenged claims”) of U.S. Patent No. 7,814,170 (“the ’170 patent”) (EX1001) assigned to Kove IO, Inc. (“Patent Owner”).

II. MANDATORY NOTICES

Real Parties-in-Interest: Pursuant to 37 C.F.R. § 42.8(b)(1), Petitioner identifies Google LLC as the real party-in-interest.¹

Related Matters: The ’170 patent is asserted in *Kove IO, Inc. v. Amazon Web Services, Inc.*, Case No. 1:18-cv-08175 (N.D. Ill.) (“the AWS Proceeding”), and *Kove IO, Inc. v. Google LLC*, Case No. 1:23-cv-04244 (N.D. Ill.). The ’170 patent was involved in ex-parte reexamination No. 90/019,035 (“the ’035 EPR”), and is currently pending in ex-parte reexamination No. 90/019,165 (“the ’165 EPR”).

The ’035 EPR was filed by third party Amazon Web Services (“AWS”) (EX1037) in November 2021, presenting SNQs on claims 1, 2, 6, 8, 9, 12, and 15 based on different combinations of prior art from the grounds here (*id.*, 11-14;

¹ Google LLC is a subsidiary of XXVI Holdings Inc., which is a subsidiary of Alphabet Inc. XXVI Holdings Inc. and Alphabet Inc. are not real parties-in-interest to this proceeding.

infra §§IV, IX); the request was granted (EX1038, 21-22) and the claims were confirmed (EX1039, 16). Petitioner had no involvement in the preparation or filing of the '035 EPR.

The '165 EPR was filed by third party AWS (EX1040) in February 2023, presenting SNQs on claims 1, 2, 6, 8, 9, 12, and 15 based on different combinations of prior art from the grounds here (*id.*, 19-21; *infra* §§IV, IX.). Petitioner had no involvement in the preparation or filing of the '165 EPR. Because the request raised substantial new questions of patentability, reexamination was ordered in March 2023. (EX1041, 19.) The examiner has not issued an Office Action at the time of the filing of the current Petition.

Petitioner is concurrently filing IPR petitions challenging related U.S. Patent Nos. 7,103,640 (“the '640 patent”) and 7,233,978 (“the '978 patent”), which are also at issue in the above-noted district court cases. The '978 patent was also at issue in *Amazon Web Services, Inc. v. Kove IO, Inc.*, IPR2020-00276 (“the AWS-IPR”).

Counsel and Service Information: Lead counsel is Naveen Modi (Reg. No. 46,224). Backup counsel are Joseph E. Palys (Reg. No. 46,508), Daniel Zeilberger (Reg. No. 65,349), Howard Herr (*pro hac vice* to be filed), Alexa Lowman (*pro hac vice* to be filed), and Puja Patel (*pro hac vice* to be filed). Service information is Paul Hastings LLP, 2050 M Street NW, Washington, D.C.

20036, Tel.: 202.551.1700, Fax: 202.551.1705, email: PH-Google-Kove-IPR@paulhastings.com. Petitioner consents to electronic service.

III. PAYMENT OF FEES UNDER 37 C.F.R. §§ 42.15 AND 42.103

The PTO is authorized to charge any fees due during this proceeding to Deposit Account No. 50-2613.

IV. GROUNDS FOR STANDING

Petitioner certifies under 37 C.F.R. § 42.104(a) that the '170 patent is available for review and Petitioner is not barred/estopped from requesting review on the grounds identified herein.

V. PRECISE RELIEF REQUESTED AND GROUNDS RAISED

Petitioner requests review of claims 1-17 of the '170 patent and cancellation of these claims as unpatentable on the following grounds:

Ground 1: Claims 1, 2, 4, and 5 are unpatentable under pre-AIA 35 U.S.C. § 102 as being anticipated by Skagerwall (EX1006).

Ground 2: Claim 3 is unpatentable under pre-AIA 35 U.S.C. § 103 as being obvious over Skagerwall.

Ground 3: Claims 6-17 are unpatentable under pre-AIA 35 U.S.C. § 103 as being obvious over Skagerwall and Vingralek (EX1007).

As discussed below, Skagerwall and Vingralek qualify as prior art and are properly relied upon for showing unpatentability of the '170 patent.

1. Earliest Effective Filing Date of the '170 Patent

The '170 patent issued from U.S. Patent Application No 11/354,224, filed on February 13, 2006, which claims priority to application No. 09/661,222 (now the '640 patent) filed on September 13, 2000; provisional application No. 60/153,709 ("the '709 application") filed on September 14, 1999; and application No. 09/111,896 ("the '896 application") filed on July 8, 1998. (EX1001, Cover, 1:6-15.) However, the '170 patent is not entitled to claim priority to the '896 application because the claims of the '170 patent are not fully supported by the '896 application. (EX1002, ¶¶32-37.) Therefore, the earliest possible effective filing date for the '170 patent is September 14, 1999.²

In order for a claim to be entitled to the benefit of the filing date of an earlier filed application, the subject matter of the claim must be disclosed in the earlier-filed application in accordance with the written description requirement of Section 112. *PowerOasis, Inc. v. T-Mobile USA, Inc.*, 522 F.3d 1299, 1306 (Fed. Cir. 2008). Here, none of the independent claims of the '170 patent are disclosed in the

² Petitioner does not concede the '170 patent is entitled to a September 14, 1999 priority date.

'896 application; therefore, none of the claims of the '170 patent are entitled to claim priority to the '896 application.³

The '896 application does not provide written description support for at least the following limitations of the '170 patent: limitations [1.b], [6.c], and [15.e].

(EX1002, ¶¶32-37.) 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). (EX1021, 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.)

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. Limitations [6.c] and [15.e] require if a data location server does not contain the requested data location information associated with the desired entity or data entity, the server

³ Patent Owner acknowledged in the AWS Proceeding that of all claims in the '170 patent and the related '640 and '978 patents, only claims 1 and 31 of the '978 patent are allegedly supported by the '896 application. (EX1023, 7 (n.4).)

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.” (EX1021, 15:23-26.) Although the '896 application describes server response messages that 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.) The examiner in the '165 EPR likewise found “claims 1, 2, 6, 8, 9, 12, and 15 do not appear to be fully supported by the ['896] application.” (EX1041, 10.)

2. Skagerwall and Vingralek Qualify as Prior Art

Skagerwall issued from an application filed October 27, 1998 (EX1006, Cover), and thus qualifies as prior art at least under pre-AIA 35 U.S.C. § 102(e).

Vingralek is a journal article published by ETH Zürich as early as December 1993, and thus is prior art under at least pre-AIA 35 U.S.C. § 102(b). The declaration of Dr. Sylvia D. Hall-Ellis (Ex. 1020, ¶¶1-66) confirms Vingralek was published, disseminated, and reasonably available to the public on or shortly after December 31, 1993. (Ex. 1020, ¶¶46-54; *see also id.*, ¶¶19-45.) For example, Dr.

Hall-Ellis explains that based on finding a copy of Vingralek that is dated December 1993 in the ETH Zürich Library and a MARC record in its online library catalog indicating the ETH Zurich journal was first published in 1974. (*Id.*, ¶¶51-54.) Moreover, Vingralek was cited by Ex. 1015 (“Norrie”), which was published on March 1, 1994. (*Id.*, ¶¶55-63.) It is her opinion that Vingralek was publicly available on or shortly after December 31, 1993, as shown on the cover. (*Id.*, ¶¶64-65.)

VI. LEVEL OF ORDINARY SKILL

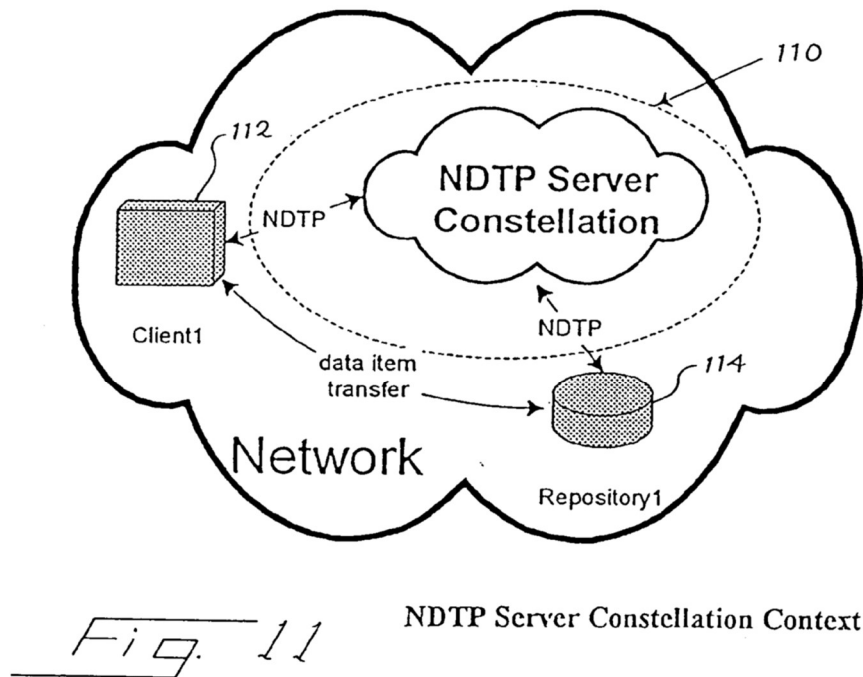
A person of ordinary skill in the art at the time of the alleged invention of the ’170 patent (“POSITA”) would have had an undergraduate degree in electrical engineering, computer engineering, computer science or a related field along with two years of work experience in the field of distributed computing systems. (EX1002, ¶¶20-21.)⁴ More education can supplement practical experience and vice versa. (*Id.*)

VII. OVERVIEW OF THE ’170 PATENT

The ’170 patent describes storing and retrieving information across a network (EX1001, 1:29-32, 1:58-60) and a network distributed tracking protocol

⁴ Petitioner submits the declaration of Dr. Sandeep Chatterjee (EX1002), an expert in the field of the ’170 patent. (EX1002, ¶¶1-28; EX1003.)

(NDTP) (*id.*, 2:14-16, 4:28-38). Figure 11 shows an NDTP server constellation where a client consults servers return strings responding to a client request. (*Id.*, 17:3-6.)



(*Id.*, FIG. 11; EX1002, ¶29.)

Server constellations have two organizational approaches: client-centric (Figure 12) and server-centric (Figure 13). In the client-centric approach, processing emphasis is toward clients (*id.*, 18:25-27), whereas in the server-centric approach, server-processing is emphasized (*id.*, 18:50-54).

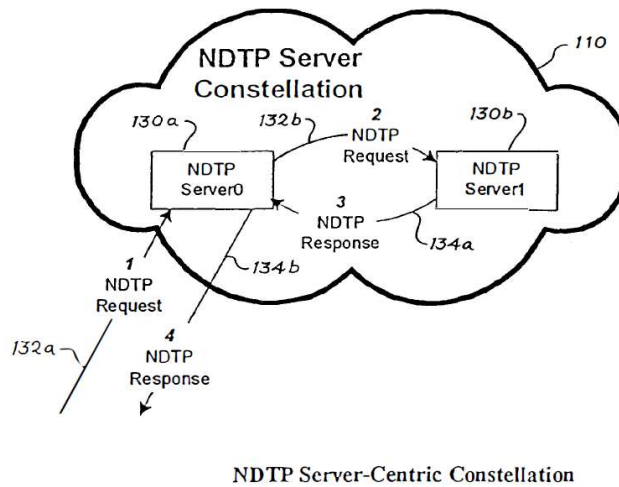
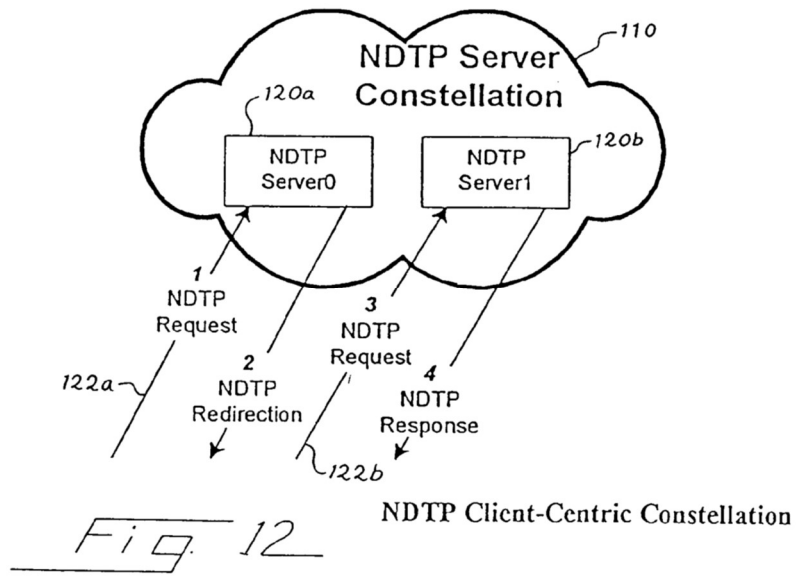


Fig. 13

(*Id.*, FIGS. 12, 13; EX1002, ¶30.)

In both approaches, a client transmits an identifier specifying an entity to a server in a client request. (EX1001, 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.*; EX1002, ¶31.)

As demonstrated below, all of the features recited in the challenged claims of the '170 patent were known in the prior art before the alleged invention date of the '170 patent. (*Infra* §IX; EX1002, ¶¶22-28, 38-44; EX1005, EX1011, EX1056-EX1059.)

VIII. CLAIM CONSTRUCTION

The Board only construes the claims when necessary to resolve the underlying controversy. *Toyota Motor Corp. v. Cellport Sys., Inc.*, IPR2015-00633, Paper No. 11 at 16 (Aug. 14, 2015). For purposes of this proceeding, Petitioner believes no special constructions are necessary to assess whether the challenged claims are unpatentable over the asserted prior art. (EX1002, ¶¶45-53.)⁵

As shown below, in the AWS-IPR, the Board construed a claim term of the related '978 patent (EX1022, 13), and in the AWS Proceeding, the district court

⁵ Petitioner reserves all rights to raise claim construction and other arguments in other venues.

construed several claim terms of the '170 patent. (EX1024, 35; EX1025, 19.)

While Petitioner does not acquiesce to those constructions, the prior art mappings in §IX explain how the '170 patent claims are disclosed and/or rendered obvious if those constructions are applied. (EX1002, ¶¶45-53.)

'170 Patent Terms	AWS Proceeding/IPR Construction
“identifier” ⁶	“a unique encoding that identifies an individual entity, and with which zero or more data location specifiers are associated in a location server” (EX1022, 13.)
“location information”	“information pertaining to one or more locations of data and/or the identities of one or more location servers” (EX1024, 35.)

⁶ Petitioner applies the construction of “identifier” recited in the '978 patent claims to '170 patent claims reciting “identifier string” because the '170 patent uses the two terms interchangeably (EX1001, 4:10-12) and describes the terms in substantially the same manner as the '978 patent. (*Compare* EX1001, 4:10-12, *with* EX1022, 10-11.)

'170 Patent Terms	AWS Proceeding/IPR Construction
"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" (EX1024, 35.)
"client"	"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" (EX1024, 35.)
"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</i>	"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

'170 Patent Terms	AWS Proceeding/IPR Construction
<i>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” ⁷	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” (EX1024, 35.)
“data pertaining to the entity”	“data pertaining to a person or thing (real, digital, or abstract) distinct from that data” (EX1024, 35.)
“plurality of location servers” ⁸	“location servers in non-hierarchical structures” (EX1025, 19.)

⁷ Petitioner applies only the bolded and italicized part of this construction, as the rest of the construction recites the exact claim language.

⁸ Petitioner also applies this construction to claims reciting “plurality of data location servers.”

IX. DETAILED EXPLANATION OF GROUNDS

As detailed below, each of the challenged claims are unpatentable.

(EX1002, ¶¶45-129.)

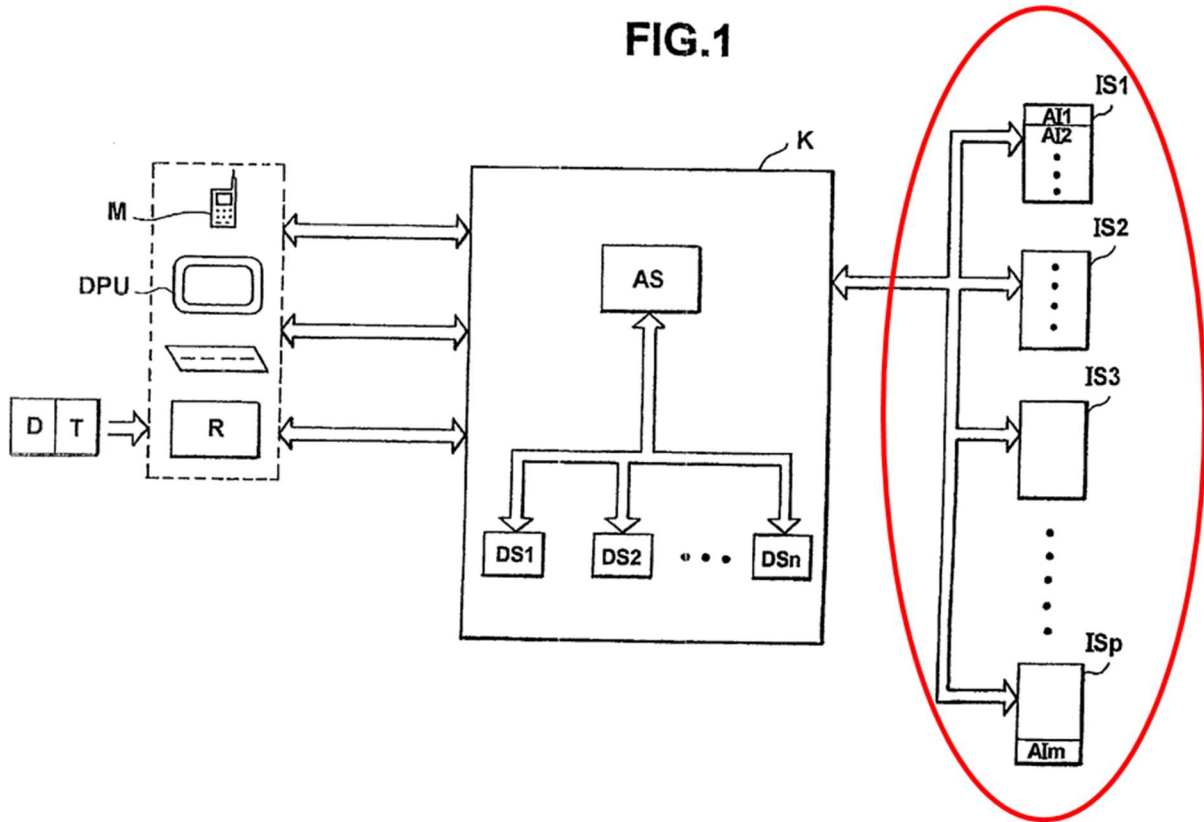
A. Ground 1: Skagerwall Anticipates Claims 1, 2, 4, and 5

1. Claim 1

i) [1.pre]

To the extent the preamble is limiting, Skagerwall discloses the limitations therein. (EX1002, ¶¶55-59.) Skagerwall discloses a system linking objects, e.g., items in the physical world, to information related to the objects. (EX1006, Abstract, 6:45-48.) The system links objects to information or application data packets (“data”) about those objects, where the application data packets are stored on application servers. (*Id.*, 4:1-11.) The application data packets include, e.g., text/programs. (*Id.*, 4:5-9, 7:13-17; EX1002, ¶55.)

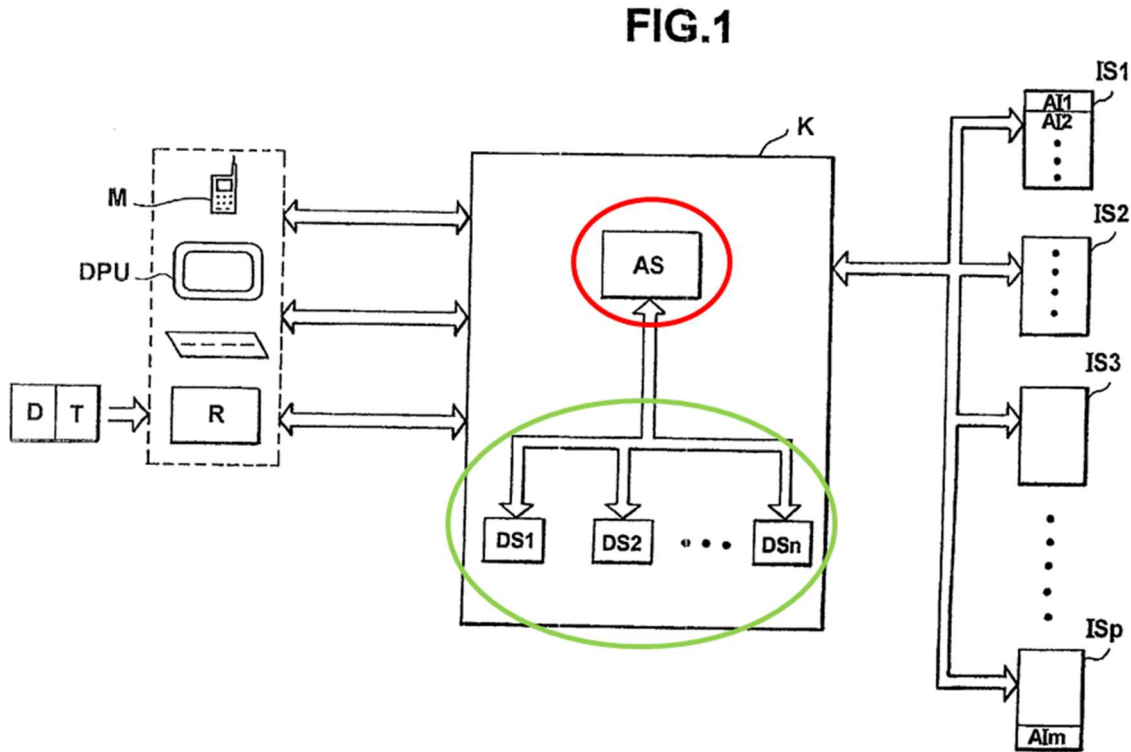
Application data packets (“data”) are stored in a distributed network of application servers (“distributed network”). As shown in Figure 1, application data packets, each identified by an application identifier (AI1-AIm), are distributed across application servers (red), each identified by an address (IS1-ISp). (EX1006, 6:34-37.)



(*Id.*, FIG. 1.) Consistent with Figure 1, a POSITA would have understood 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. (EX1002, ¶56.) The servers may be placed at arbitrary locations, similar to intranets/Internet, which a POSITA would have recognized are distributed networks. (EX1006, 7:4-8.)

Skagerwall's system "manag[es] data stored in a distributed network." (EX1002, ¶57.) Skagerwall discloses retrieving application data packets ("data stored in a distributed network"), which is a way of managing the application data

packets. As shown in Figure 1, the system includes an access server (red) and a network of directory servers (green). (EX1006, 6:32-34.)⁹



(*Id.*, FIG. 1.) Each directory server stores object data, which associates objects with application data packets (“data”), and is involved in retrieving application data packets from application servers (“data stored in a distributed network”). (*Id.*,

⁹ A POSITA would have understood 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., EX1006, 9:1-9.)

4:13-16, 8:19-22, Abstract, FIG. 1.) The object data is associated with an object tag and includes application server addresses indicating the storage location of the application data packets, and application identifiers of the packets. (*Id.*, 8:22-27.)

Application server addresses, indicating 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. (EX1006, 8:28-36; EX1002, ¶58.) The first step comprises identifying a directory server storing object data (including application server addresses) relating to an object and retrieving those addresses. (EX1006, 8:22-32, 9:5-9, 9:42-48, 14:7-11.) The second step involves using the object data to retrieve application data packets (“data”) from the application servers, or, transmitting the object data directly to the user. (*Id.*, 8:32-36.)

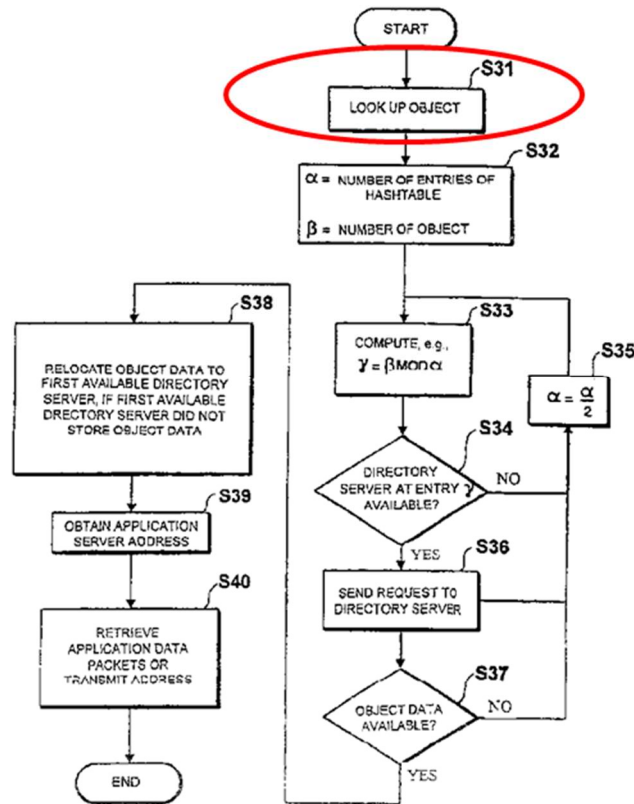
In addition to retrieving application data packets associated with a specific object, Skagerwall discloses storing application data packets (“data stored in a distributed network”) (EX1006, 14:54-56), which is also a way of managing the application data packets. (EX1002, ¶59.)

ii) [1.a]

Skagerwall discloses this limitation. (EX1002, ¶¶60-63.) **First**, Skagerwall discloses “a data repository configured to store a data entity.” As discussed for limitation [1.pre], application data packets (“data”) associated with an object are stored in a distributed network of application servers. (§IX.A.1.i.) A POSITA would have understood the application servers together form a “data repository” because they store the application data packets (“data”) associated with tagged objects. (EX1006, 4:1-12.) A POSITA would have also understood 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. (EX1002, ¶¶60; §IX.A.1.i.) Accordingly, Skagerwall discloses “a data repository configured to store a data entity.”

Second, Skagerwall discloses “wherein an identifier string identifies the data entity.” (EX1002, ¶¶61.) Regarding the process for retrieving application data packets, the system kernel includes an access server that receives an information request about an object (step S31 of Figure 3), where the request at least includes an object tag (“identifier string”). (EX1006, 9:1-5, 12:27-32, 13:23-26, 13:29-30, FIGs. 2-3.)

FIG.3



(*Id.*, FIG. 3.) 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, 7:22-23.) Upon receiving this request for application data packets associated with an object, which includes the object tag (“identifier string”), the access server performs operations (e.g., 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.)

A POSITA would have understood an object tag, e.g., 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 POSITA would have further understood an object tag is an “identifier string” because the tag identifies a given object. (EX1002, ¶62; EX1006, 2:53-54.) A POSITA would have similarly understood 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. (§IX.A.1.i.)

The object tag is an “identifier string” under the AWS-IPR construction of related term “identifier.” (§VIII; EX1001, 4:10-12.) A POSITA would have understood the object tag is an encoding that uniquely identifies both an object (i.e., individual entity) and its data representation (“data entity”) because, as discussed above, the tag is included in a query to identify the specific object for which information is sought. As discussed for limitation [1.b], the object tag (“identifier string”) is associated with zero/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 construed in the AWS Proceeding). (*Infra* §IX.A.1.iii; EX1002, ¶63.)

iii) [1.b]

Skagerwall discloses this limitation. (EX1002, ¶¶64-73.) **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 directory servers (“a data location server network comprising...data location servers”). (§IX.A.1.i; EX1006, 8:15-19.) The application server addresses (“data location information”) are stored on directory servers (“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”). (§IX.A.1.i; EX1006, 8:19-27.) As discussed for limitation [1.pre], Skagerwall discloses 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”). (§IX.A.1.i; EX1006, 8:22-27; EX1002, ¶64.)

Skagerwall discloses the claimed “data location server[]” under the AWS Proceeding’s construction of “location server.” (§VIII; EX1001, 4:17-20.) Skagerwall discloses each directory server is a component attached to a distributed network. (EX1006, 8:15-19, 16:34-36, FIG. 1; §IX.A.1.i.) 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 object tags and associated object data are distributed across directory servers based on a hash function, and each directory server has a memory means associating object tags with corresponding object data (including application server addresses). (EX1002, ¶65; EX1006, 8:43-63, 15:15-20, 15:49-52.)¹⁰ Further, Skagerwall discloses the directory server returns identifier/location mappings in response to a request from the access server for location information associated with an object tag (i.e., in response to location request messages from a client). 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. (EX1001, 2:36-40, 2:60-65, 6:63-67, 16:53-56.)

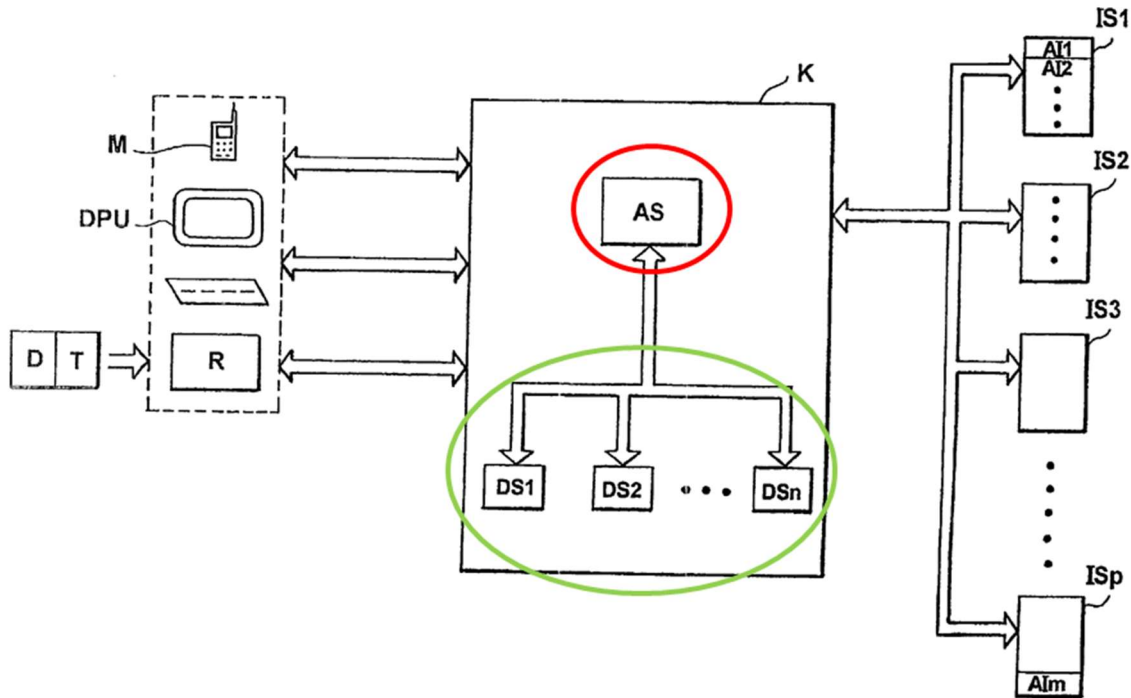
¹⁰ Skagerwall’s application server address likewise discloses the “location” (or similar terms) as described in the ’170 patent (EX1001, 4:12-14) for reasons discussed above.

Skagerwall also discloses the corresponding object tags (i.e., identifiers) are returned with the application server addresses. (EX1006, 19:42-46, 19:51-55.)

Skagerwall further discloses identifier/location mappings may be updated/modified responding 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; EX1001, 6:50-53.) Thus, Skagerwall discloses the claimed “data location server[]” under the AWS Proceeding’s construction of “location server.”

Skagerwall discloses the claimed “plurality of data location servers” under the AWS Proceeding’s construction of “plurality of location servers.” (§VIII.) A POSITA would have understood 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 in Figure 1—there is no hierarchy of intermediary servers between the access server (red) and the directory servers (green), and the directory servers themselves are not arranged in a hierarchy. (EX1002, ¶66.)

FIG.1



(EX1006, FIG. 1.) Skagerwall discloses 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 POSITA would have thus understood because a hash function is used for distribution across directory servers, these servers are arranged in a non-hierarchical configuration. Consistent with the above, Patent Owner argued in the AWS Proceeding the hash function feature requires a non-hierarchical configuration. (EX1025, 15.)

Skagerwall discloses the claimed “data location information” and “location information” under the AWS Proceeding’s construction of “location information”

(§VIII) as each of the application server addresses identifies the location of application data packets (“data”). (EX1006, 8:22-27, 15:15-52; EX1002, ¶67.)

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.” (EX1002, ¶68.)

Skagerwall discloses each directory server (“each one of the plurality of data location servers”) has a central processing unit (“comprises a processor”).

(EX1006, 15:1-7.) The database of object data, which includes application server addresses (“data location information”), is distributed across directory servers, such that each (“each one of the plurality of data location servers”) stores object data related to certain object tags. (*Id.*, 8:15-27.) Further, 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...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, 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 the full database of

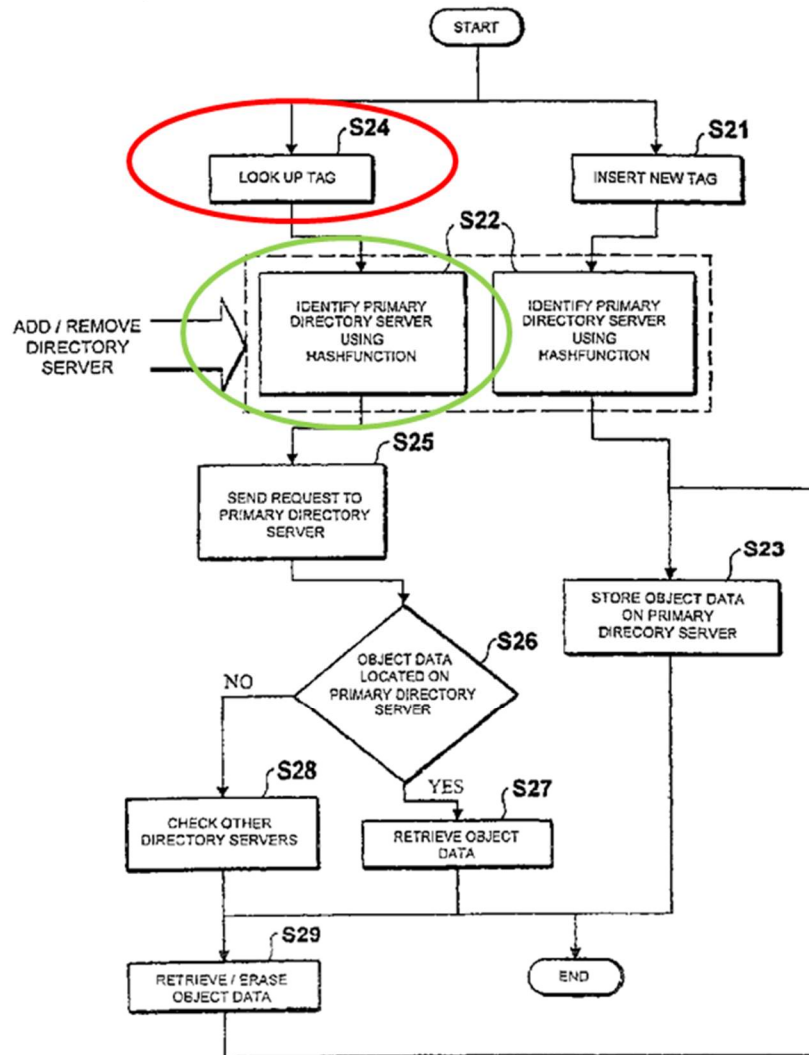
object data is distributed across the plurality of directory servers according to a hash function/hash table that assigns object data to corresponding directory servers (“the portion of the data location information included in a corresponding one of the data location servers is based on a hash function used to organize the data location information across the plurality of data location servers”).

Skagerwall discloses 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 AWS Proceeding’s construction. (§VIII.) Skagerwall discloses 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. (EX1006, 8:43-54; EX1002, ¶69.)

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.” As discussed below, Skagerwall discloses as part of the process for retrieving application data packets, a directory server (“data location server”) applies a hash function to an object tag (“identifier string”) to determine the directory server that stores information specifying the location of the application data packets. (EX1002, ¶70.)

Skagerwall discloses 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. (EX1006, 9:18-22.) Skagerwall discloses with respect to Figure 2a 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, 10:28-11:16, 12:8-12, 13:15-57; EX1002, ¶71.)

FIG.2a

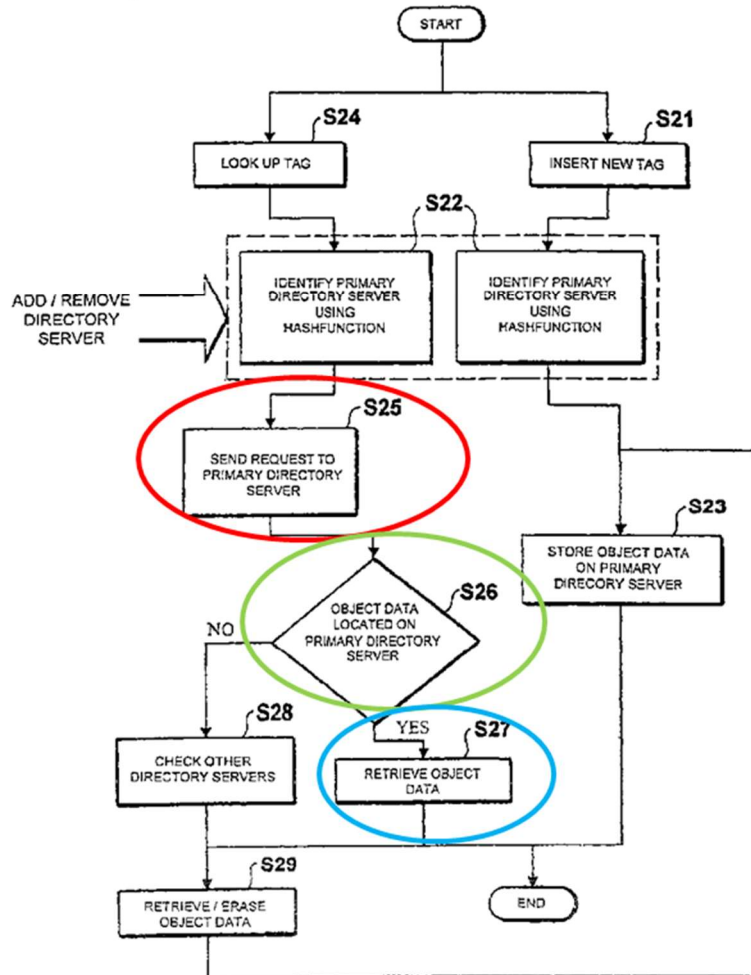


(EX1006, FIG. 2a.)

Skagerwall explains 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.*, 8:22-27, 9:9-13, 9:23-25, 12:37-40, FIG. 2a.) 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.)

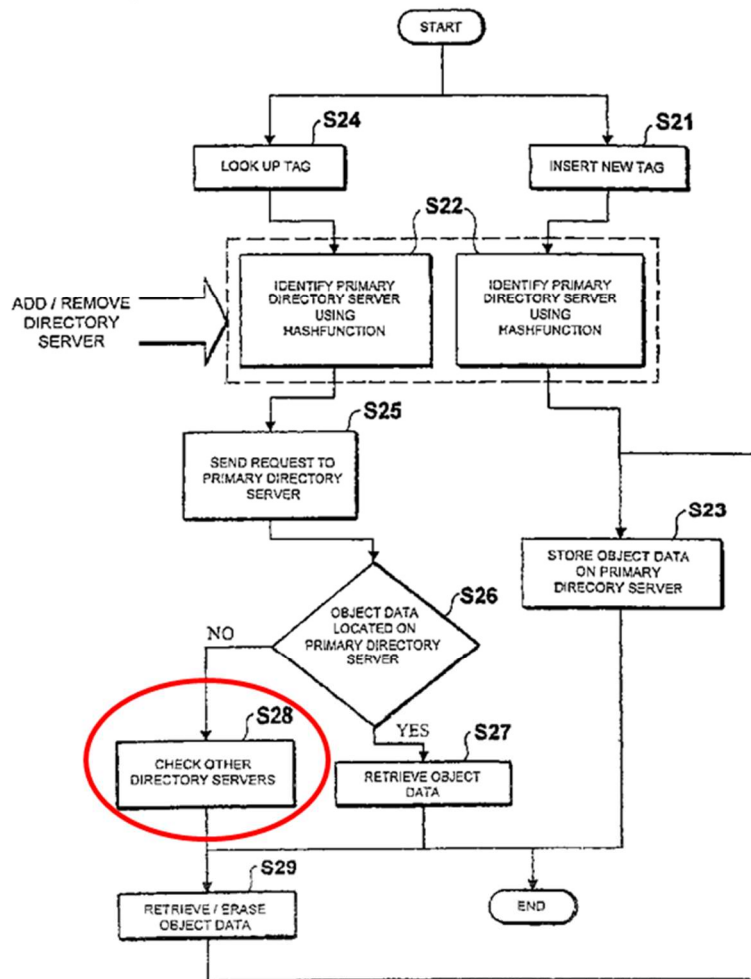
FIG.2a



(*Id.*, FIG. 2a; EX1002, ¶72.)

If the directory server determines in step S26 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. (EX1006, 12:63-13:5.)

FIG.2a



(*Id.*, FIG. 2a.) 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”). (EX1002, ¶73.) Skagerwall discloses 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”). (EX1006, 9:18-31, 10:22-11:16.)

2. Claim 2

Skagerwall discloses this limitation. (EX1002, ¶74.) As discussed for limitation [1.b], Skagerwall discloses a hash function applied to an object tag, with the result serving as an index to a table of directory servers (i.e., a hash table). (§IX.A.1.iii; EX1006, 10:28-11:16, 13:15-57.) As further discussed, 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

the full database of object data is distributed across the directory servers according to a hash function/hash table that assigns object data to directory servers based on object tag. (§IX.A.1.iii; EX1006, 8:15-27, 8:38-54, 8:55-59.) A POSITA would have therefore understood 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 directory servers because each includes object data associated with a portion of the identifiers in the hash table. A POSITA would have also understood 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. (EX1002, ¶74.)

3. Claim 4

Skagerwall discloses this limitation. (EX1002, ¶75.) 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”). (§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”). (EX1006, 16:1-4; *see also id.*, 9:34-38; §IX.A.1.iii.)

4. Claim 5

Skagerwall discloses this limitation. (EX1002, ¶¶76-77.) **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”)¹¹ 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”). (§IX.A.1.iii.) Skagerwall discloses 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

¹¹ Petitioner assumes the claimed “string identifier” is intended to refer to the “identifier string” in claim 1. (§IX.A.1.ii.)

identifier of the at least one of the plurality of data location servers”) of the corresponding directory server. (EX1006, 10:31-38; EX1002, ¶76.)

Second, Skagerwall discloses “wherein the location information includes a hash table.” (EX1002, ¶77.) As discussed for claim 2, the “location information” on a given directory server “comprises any portion of a hash table,” and thus a POSITA would have understood all the “location information” collectively comprises a hash table. (§IX.A.2.) To the extent claim 5 is read to require the claimed “hash table” *is* “location information,” a POSITA would have understood 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 the hash table is “location information” under the AWS Proceeding’s construction. (§VIII.) 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. Ground 2: Skagerwall Renders Claim 3 Obvious

1. Claim 3

Skagerwall discloses and/or suggests this limitation. (EX1002, ¶¶78-82.)

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”) storing application data packets (“data”) comprises multiple application servers (“wherein the data repository comprises a plurality of data repository servers”). (§IX.A.1.ii; EX1002, ¶78.)

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 directory servers. (EX1006, 12:3-26, 14:29-53.) 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 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; EX1002, ¶79.)

To the extent 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, a POSITA would have found it obvious to implement this feature when, e.g., 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 where the data/information services (e.g., application data packets) for an object and associated object data (e.g., location information) are subject to change. (EX1006, 2:30-40, 4:1-12, 8:15-19, 8:59-67, 11:16-26, 12:47-51.) A POSITA would have understood, 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. (EX1002, ¶80.) 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. (EX1006, 5:28-37, 10:15-20, 17:1-5, 18:54-58.)

Consistent with Skagerwall's process for identifying application servers for retrieving application data packets, as discussed in limitation [1.b], a POSITA would have understood identifying the application servers for storing application data packets would have likewise involved identifying/retrieving the application server addresses associated with an object tag. (EX1002, ¶81; §IX.A.1.iii; EX1006, 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). (§IX.A.1.iii.) Thus, a POSITA would have found it obvious 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 ("a request to store data in the data repository"), each directory server ("each...of data location servers") would be configured 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...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. (EX1006, 14:54-56; *id.*, 2:30-40, 4:1-12, 8:15-19, 8:59-67, 11:16-26, 12:47-51.)

A POSITA 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 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. *KSR Int’l Co. v. Teleflex Inc.*, 550 U.S. 398, 416 (2007). Accordingly, Skagerwall discloses and/or suggests this limitation. (EX1002, ¶82.)

C. Ground 3: Skagerwall in View of Vingralek Renders Claims 6-17 Obvious

1. Claim 6

i) [6.pre]

Skagerwall discloses this limitation. As discussed for limitation [1.pre], Skagerwall discloses a system (“system”) linking real-world objects to information related to the objects. (§IX.A.1.i.) As discussed for limitation [1.b], Skagerwall’s system performs processes for retrieving application data packets (“data”) associated with an object. (§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”). (§IX.A.1.iii.) A POSITA would have understood retrieving information based on a request is a way of managing information, and thus a system that performs processes retrieving application server addresses (“data location information”) is a “system for managing data location information.” (EX1002, ¶83.)

ii) [6.a]

Skagerwall discloses and/or suggests this limitation. (EX1002, ¶¶84-90.)

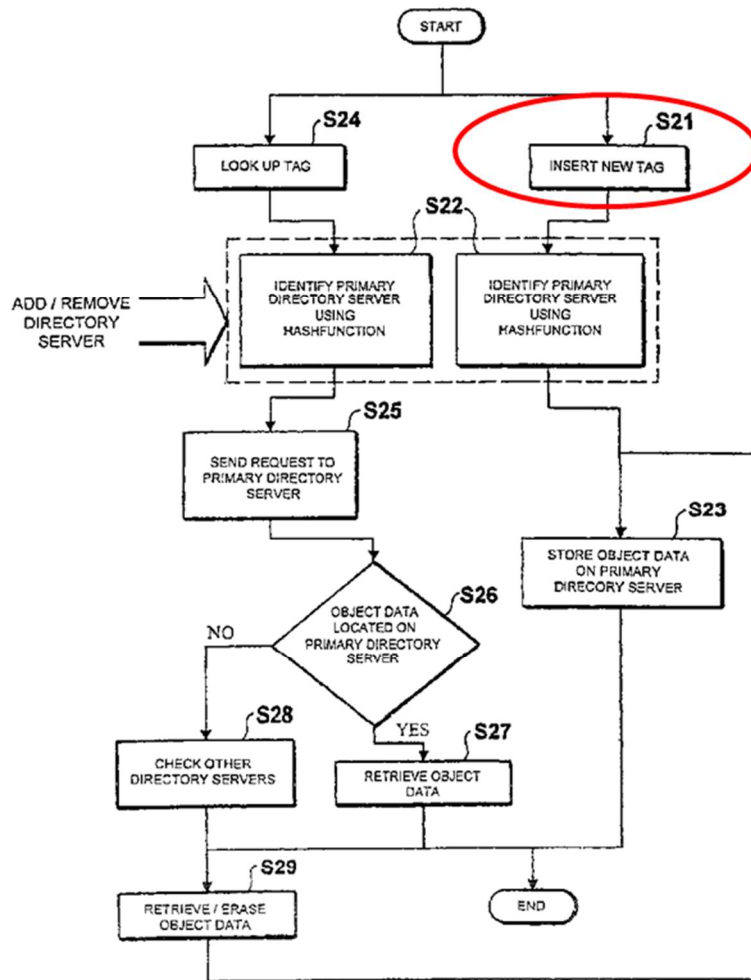
First, Skagerwall discloses “a location server configured to receive a location

addition request.” As discussed for limitation [1.b], Skagerwall discloses a network of directory servers (each a “location server”)¹² 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. (EX1006, 12:3-4.) Skagerwall discloses a request (“location addition request”) for insertion of new object data (e.g., application server addresses (“location[s]”)) corresponding to an object tag (“identifier”)¹³ is received by the system kernel in step S21 (red). (*Id.*, 12:5-7, 14:37-40.)

¹² Skagerwall discloses a “location server” as construed in the AWS Proceeding for the same reasons as discussed for limitation [1.b]. (§§ VIII, IX.A.1.iii.)

¹³ Skagerwall discloses an “identifier” as construed in the AWS-IPR for the same reasons as discussed for limitation [1.a]. (§§ VIII, IX.A.1.ii.)

FIG.2a



(*Id.*, FIG. 2a.) 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, 14:37-53.) In step S23, the request to store the object data is sent to the target directory 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; EX1002, ¶84.)

Second, Skagerwall discloses and/or suggests “the location addition request formatted in conformance with a transfer protocol.” A POSITA 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 its system may “use standard and/or amended protocols...for retrieving information associated with an object.” (EX1006, 9:59-62.) 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 POSITA 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 POSITA 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 POSITA would have found it obvious 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”). (EX1002, ¶85.)

A POSITA would have also had a reasonable expectation of success in implementing this feature because it would have involved modifying Skagerwall in a manner that Skagerwall suggests is feasible—implementing the system in accordance with a standard protocol such as HTTP. (EX1002, ¶86.) Such a modification would have involved applications of known technologies (e.g., known systems of Skagerwall) according to known methods (e.g., known transfer protocol used in Skagerwall or HTTP) to yield the predictable result of a system sending 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. *KSR*, 550 U.S. at 416.

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). (§IX.A.1.iii.) Thus, a POSITA would have found it obvious 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”). (EX1002, ¶87.) A POSITA 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. (*Id.*)

A POSITA 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. (EX1002, ¶88.) Skagerwall discloses storing the object data on the directory servers as part of the insertion process (EX1006, 12:1-26), and storing the object data alongside the associated object tag (*id.*, 8:43-63, 15:15-20, 15:49-52). 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. *KSR*, 550 U.S. at 416.

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 an object tag (“identifier”) identifies a real-world object (“entity”). (§IX.A.1.ii.) And as discussed for limitation [1.pre], Skagerwall discloses 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. (§IX.A.1.i; EX1002, ¶89.)

Skagerwall discloses “data in a network pertaining to the entity” under the AWS Proceeding’s construction of “data pertaining to an entity.” (§VIII.) Skagerwall discloses its system is for managing information associated with real-world objects, which may be “any products or items of the physical world” (i.e., real-world object/thing). (EX1006, 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. (§IX.A.1.ii; EX1002, ¶90.)

iii) [6.b]

Skagerwall discloses this limitation. (EX1002, ¶91.) As discussed for limitation [1.b], each of Skagerwall's directory servers ("location server") includes a CPU ("processor"). (§IX.A.1.iii.)

iv) [6.c]

Skagerwall in view of Vingralek discloses and/or suggests this limitation. (EX1002, ¶¶92-108.) **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 controlling the operations of the server. (§§IX.A.1.iii, IX.C.1.iii.) A POSITA 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 to perform each of the operations of the directory server, including those recited in limitation [6.c] and discussed below. (EX1002, ¶92.)

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.”

(EX1002, ¶93.) As discussed for limitation [1.b], the primary directory 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.

(§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 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.*) 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”).

(EX1006, 9:42-48.) A POSITA 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 POSITA would have found it obvious 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], it would have been obvious to format communications between an access server and directory server, e.g., the location message discussed above, in accordance with HTTP. (§IX.C.1.ii.)

Third, Skagerwall discloses “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.” (EX1002, ¶94.) 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. (§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. (§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. (EX1006, 12:63-13:5.)

Skagerwall discloses 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, 10:22-25.) Skagerwall discloses 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; EX1002, ¶95.)

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.” (EX1002, ¶96.)

To the extent that 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,**” a POSITA would have been motivated and found it obvious to implement this feature in view of Vingralek. (EX1002, ¶97.)

Like Skagerwall, Vingralek relates to storing data in a distributed network. Vingralek discloses a “distributed file organization system” where a file is “organized into buckets that are spread across multiple servers” (EX1007, 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). 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.) 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, 8.) Vingralek’s hash-based distributed file system is implemented using an address table, which a POSITA would have understood is akin to a hash table, to look up the server corresponding to a hashed key. (*Id.*, 4, 6.) Thus, Vingralek is similar to Skagerwall, disclosing 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. (EX1006, 8:15-19, 8:43-54; §IX.A.1.i.) A POSITA would have therefore been motivated to consider the teachings of Vingralek when implementing Skagerwall’s system/method. (EX1002, ¶98.)

Vingralek additionally discloses propagating updates to clients’ and other servers’ address tables in a deferred manner. (EX1007, 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.*, 5-6; EX1002, ¶99.) Thus, Vingralek discloses deferring updates to a client’s stale address table until a wrong first server is reached and the location of a second server needs to be calculated.

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. (EX1007, 9.) Each client and server has its own copy of

the address table, but, for example, the client's table may have stale information if changes have been made to the servers since the table was last updated. (*Id.*, 10; *see also id.*, 11-13.) When a client seeks to retrieve records associated with a given key, it applies the hash function to the key, identifies the server where that key is located by using the hashed key as an index into the current copy of the address table, and then sends the key in a request to that first server. (*Id.*, 9-10.) The first server that receives the key also applies the hash function to the key and identifies in its copy of the address table the server where the key resides (at least according to that first server's address table). (*Id.*, 10-11.) If the key was indeed sent to the correct server (i.e., if the identified server is the same as that first server), then that first server performs the requested operation. (*Id.*, 11.) But if the key was sent to the wrong server, the first server forwards the request to another server (i.e., the one identified by the above-described hash function application), and sends a message to the client to update the client's address table. (*Id.*; *see also id.*, 25 (“client requests and server responses” are “messages”); EX1002, ¶100.)

The key is eventually received by the correct server and the request from the client is executed. (EX1007, 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 POSITA 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. (EX1002, ¶101.)

Additionally, Vingralek discloses that the "most salient feature" of its approach, where a client's hash table is updated in a deferred manner, "is that its communication overhead is largely independent of the number of servers and clients in the system; thus, it is considered a scalable approach." (EX1007, 5.) Thus, Vingralek discloses implementing a redirection mechanism and additionally includes improved features to minimize the number of servers and reduce the cost of scaling up a distributed system. (*Id.*, 6, 8; EX1002, ¶102.)

A POSITA 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. (EX1002, ¶103.) Indeed, Skagerwall explains, 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 to conserve resources. (EX1006, 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. (EX1007, 5.) Thus, a POSITA 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. (EX1002, ¶103.)

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. (EX1002, ¶104.) 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." (EX1006, 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 POSITA 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”). (EX1002, ¶104.) 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. (*Id.*) A POSITA 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. (*Id.*) 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). (EX1007, 5, 7, 8; EX1006, 5:28-37, 10:15-20, 17:1-5, 18:54-58.)

A POSITA 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. (EX1002, ¶105.) Thus, calculating the addresses of correct directory server(s) at a directory server and transmitting a redirect message, including those addresses, to the access server for the access server to forward the request to that correct directory server, would have been obvious because it would have been one of a “finite number of identified, predictable solutions.” *Perfect Web Techs., Inc. v. InfoUSA, Inc.*, 587 F.3d 1324, 1331 (Fed. Cir. 2009). Therefore, a POSITA would have been motivated to include features similar to Vingralek's “redirect message” features when implementing Skagerwall.

A POSITA would have had a reasonable expectation of success combining the teachings of Skagerwall and Vingralek. (EX1002, ¶106.) Both references are in the same field of art. (*See supra.*) Furthermore, Skagerwall discloses an “error message” that is provided if no directory servers stores the requested object data (EX1006, 13:5-6, 13:62-65), akin to the claimed “redirect message” provided if the location server lacks the requested information. Additionally, the concept of providing a message when requested information is not found was well known. (*Id.*, 2:40-43.) Moreover, Skagerwall itself suggests that its system/process is flexible and additional processes may be included, e.g., the process of relaying a lookup request to a correct directory server. (*Id.*, 9:39-41.)

Additionally, a POSITA would have understood that implementing the combination would have involved modifying Skagerwall in a manner that itself suggests is feasible—implementing a means of occasionally updating a hash table to reduce communication overhead, and 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. *KSR*, 550 U.S. at 416. Accordingly, the Skagerwall-Vingralek combination discloses and/or suggests this limitation.

(EX1002, ¶107.)

Consistent with the above and in such a combined system, e.g., 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 to determine whether it has the requested location information. (EX1002, ¶108.)

If the primary directory server does not have the requested location information, e.g., application server addresses associated with an object tag (“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 *supra*) that has the requested location information (“at least one other location server known to have the location information for the desired entity”). (*Id.*) 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”). (*Id.*) 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 provide the access server with the addresses of correct directory servers, to which the access server then forwards the request. (*Id.*)

2. Claim 7

The Skagerwall-Vingralek combination discloses and/or suggests this limitation. (EX1002, ¶109.) Skagerwall discloses that a hash table maps directory servers (“location servers”) in a network to a hashed object tag (i.e., the output of a hash function applied to an object tag). (EX1006, 4:66-5:10, 8:38-67, 10:27-11:11, FIG. 2.) As discussed for limitation [1.b] regarding the location information retrieval process, the hash table indicates which directory servers store location information associated with a given object tag. (§IX.A.1.iii.) As discussed for limitations [1.pre] and [1.a], object tags identify real-world objects (“plurality of entities”). (§§IX.A.1.i-ii.) A POSITA would have understood that a hash table includes a list of directory servers (“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...entities having location information on the respective one...of location servers”). (§IX.A.1.iii.) As discussed for limitation [6.c], a POSITA would have found it obvious 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”). (§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. (§IX.C.1.iv.) A POSITA 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

Skagerwall discloses this limitation. (EX1002, ¶110.) As discussed for limitation [1.b], Skagerwall discloses that each of the directory servers (“each...of location servers”) in the network stores only a portion of the application server addresses (“only a portion of the data location information”). (§IX.A.1.iii.)

4. Claim 9

Skagerwall discloses this limitation. (EX1002, ¶111.) Skagerwall discloses that each directory server (“location server”) includes a memory means (“indexed location store”) storing application server addresses (“location information”) along with corresponding object tags. (EX1006, 15:15-20, 15:49-51.) A POSITA 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

Skagerwall discloses this limitation. (EX1002, ¶112.) First, Skagerwall discloses “the indexed location store comprises a string store.” As discussed for limitation [1.a], a POSITA would have understood that each of Skagerwall’s object tags is a string. (§IX.A.1.ii.) A POSITA 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 uses an object tag as an index to retrieve all associated application server addresses. (§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. (EX1006, 8:43-54.)

6. Claim 11

Skagerwall discloses this limitation. (EX1002, ¶¶113-114.) As discussed for claim 10, the memory means is an “indexed location store [that] comprises a string store indexed by a hash table.” (§IX.C.5.) Skagerwall discloses a system that may have multiple access servers (“a plurality of clients”). (EX1006, 9:14-17.) A POSITA would have understood that the hash table is distributed across the plurality of access servers (“distributed across a plurality of clients”) in at least two ways. (EX1002, ¶113.) First, the hash table is distributed across the access servers because Skagerwall discloses that the hash table may be stored on each access server. (EX1006, 10:12-14.) Second, the hash table is distributed across the 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.)

Skagerwall discloses the claimed “client” under the AWS Proceeding’s construction. (§VIII; EX1001, 4:14-17.) A POSITA 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.” (EX1006, 16:34-36; EX1002, ¶114.) 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). (§IX.A.1.iii.)

7. Claim 12

Skagerwall discloses this limitation. (EX1002, ¶115.) As discussed for claim 10, the memory means is an “indexed location store [that] comprises a string store indexed by a hash table.” (§IX.C.5.) A POSITA would have understood that the hash table is distributed across the plurality of directory servers (“a plurality of servers”) in at least two ways. First, the hash table is distributed across the directory servers because Skagerwall discloses that the hash table may be stored on each directory server. (EX1006, 10:9-12.) Second, the hash table is distributed across the directory servers because the hash function distributes the object tags in the hash table across the directory servers to assign related object data to those servers. (*Id.*, 8:43-54.)

8. Claim 13

The Skagerwall-Vingralek combination discloses and/or suggests this limitation. (EX1002, ¶¶116-117.) **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. (§§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.” (§IX.A.1.ii; EX1002, ¶116.)

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”) configured to perform the directory server’s operations. (§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”). (§IX.C.1.iv.) A POSITA 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.” (EX1002, ¶117.)

9. Claim 14

Skagerwall discloses this limitation. (EX1002, ¶118.) As discussed for claim 10, a directory server’s memory means (“indexed location store”) comprises a “string store.” (§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”). (§IX.C.9.)

10. Claim 15

i) [15.pre]

Skagerwall discloses this limitation. (EX1002, ¶119.) As discussed for limitation [1.pre], Skagerwall discloses a system that links real-world objects to information related to the objects. (§IX.A.1.i.) As discussed for limitation [1.b], the system includes a network of directory servers (“network comprising a plurality of location servers”)¹⁴ that store application server addresses (“data location information”) identifying the location of application data packets (“data”).

¹⁴ Skagerwall discloses a “location server” and “plurality of location servers” as construed in the AWS Proceeding for the same reasons as discussed for limitation [1.b]. (§§VIII, IX.A.1.iii.)

(§IX.A.1.iii.) As further discussed, the system performs a method of handling requests for 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”). (§IX.A.1.iii.)

ii) [15.a]

Skagerwall discloses this limitation. (EX1002, ¶¶120-121.) **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, and that each directory server maintains a set of identifier/location mappings, where each object tag (“identifier”)¹⁵ is associated with corresponding application server addresses (“locations”) in the server (“correlating each one of a plurality of locations in the network”). (§IX.A.iii.) As discussed for limitation [1.a], Skagerwall discloses that each object tag identifies an object’s application data packets (“identifying a respective one of a plurality of data entities”), where

¹⁵ Skagerwall discloses an “identifier” as construed in the AWS-IPR for the same reasons as discussed for limitation [1.a]. (§§VIII, IX.A.1.ii.)

application packets associated with multiple objects (“data entities”) are stored on a network of application servers. (§IX.A.1.ii; EX1002, ¶120.)

Second, Skagerwall discloses “wherein the data entities are stored in corresponding locations in the network.” (EX1002, ¶121.) 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 corresponding to an object tag (“stored in corresponding locations” in a network). (§IX.A.iii.) Skagerwall 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”), where directory and application servers may communicate via the Internet (i.e., on the same network). (EX1006, 6:40-41, 17:9-12.) Skagerwall also discloses that the entire interconnected system of application and directory servers communicate over some other known network. (*Id.*, 4:1-5, 22:45-48.)

iii) [15.b]

Skagerwall discloses this limitation. (EX1002, ¶122.) 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”). (§IX.A.1.iii.) 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.” (§§IX.A.1.ii, IX.C.10.ii.)

iv) [15.c]

Skagerwall discloses this limitation. (EX1002, ¶123.) As discussed for limitation [6.c], Skagerwall discloses that if 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”). (§IX.C.1.iv.)

v) [15.d]

Skagerwall discloses this limitation. (EX1002, ¶124.) 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”).

(§IX.C.1.iv.)

vi) [15.e]

The Skagerwall-Vingralek combination discloses and/or suggests this limitation. (EX1002, ¶125.) 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 (“identifying which of the plurality of location servers includes the location information”). (§IX.C.1.iv.)

11. Claim 16

Skagerwall discloses this limitation. (EX1002, ¶126.) As discussed for claim 13, a POSITA would have understood that Skagerwall’s disclosed hash function is an “index function.” (§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. (§IX.C.10.ii.) As discussed for

limitation [6.c], Skagerwall discloses that the directory server (“location server”) applies a hash function (“indexing function”), using an object tag as an input (“applying an identifier of the data entity to an indexing function”), 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”). (§IX.C.1.iv.)

12. Claim 17

Skagerwall discloses this limitation. (EX1002, ¶¶127-128.) **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 applied to an object tag (“identifier”) to generate an index to a hash table, where the hash table maps the object tags to corresponding addresses of directory servers (“associated locations”) that store information pertaining to a certain object tag. (EX1006, 10:21-11:15, 13:15-57.) A POSITA would have recognized the hash table is therefore “an indexed table of the identifiers and corresponding associated locations.” As discussed for claim 2, each directory server (“each...of location servers”) stores location information (e.g., application server addresses) that comprise a portion of a hash table. (§IX.A.2.) Thus, a POSITA would have recognized each directory server (“each...of location servers”) also comprises a

portion of the hash table (“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. (EX1002, ¶127.)

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 by using a hash function that takes an object tag as an input. (§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 in the hash table (“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. (§IX.A.1.ii.) Thus, a POSITA would have understood 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”). (EX1002, ¶128.)

X. DISCRETIONARY DENIAL IS NOT APPROPRIATE

A. The Board Should Not Deny Institution Under § 325

Discretionary denial under § 325(d) is not appropriate because this petition does not present the same or substantially the same prior art or arguments previously presented to the Office during prosecution. (EX1004; EX1001, Cover.) There are material differences between the art/arguments here and those during prosecution (*see, e.g.*, EX1004, 94-101), including features believed to be missing (*id.*, 132-136). Discretionary denial is also not warranted in view of the completed *ex parte* reexamination proceeding, since this petition does not present the same or substantially the same prior art or arguments presented in that reexamination, and there are material differences between the art/arguments here with those presented therein (*see, e.g.*, EX1037, 11-13), including features believed to be missing (EX1039, 11-16.) And the Office erred in a material manner by allowing the claims despite such features being known. *Advanced Bionics*, Paper 6 at 10-11; *Becton, Dickinson*, Paper 8.

Discretionary denial is also not warranted in view of the pending *ex parte* reexamination initiated by a different party, AWS. (EX1041; §II.) First, Petitioner had no involvement with the reexamination, and Petitioner developed the present petition independently of AWS.

Second, this petition and the reexamination are based on materially different grounds and positions. None of the positions in the reexamination relies on Vingralek, which is asserted in Ground 3 here. (EX1041, 13-19; §§IV, IX.) While Petitioner's Grounds 1 addresses claims 1 and 2 based on Skagerwall, Ground 1 additionally includes claims 4 and 5, and Ground 2 also addresses claim 3 based on Skagerwall, and claims 3-5 are not addressed in the reexamination. (EX1041, 13-19.)

Third, the pending claims of the reexamination and the challenged claims in this IPR are not coextensive. (§§II, IV; EX1041, 19.)

Fourth, denying institution in view of a third-party reexamination would be unfairly prejudicial. The reexamination does not challenge all of the claims that are challenged in this proceeding. Moreover, it is Patent Owner that decided to accuse separate companies of infringement, each with separate interests in challenging the patent. Given the statutory deadline to file an IPR, Google could not have waited until the conclusion of the reexamination. *Mueller Sys., LLC v. Rein Tech, Inc.*, IPR2020-00100, Paper 9 at 12-14 (P.T.A.B. May 12, 2020).

Fifth, the reexamination remains at an early stage, which counsels against discretionary denial. *Mueller* at 13.

B. Discretionary Denial Under *General Plastic* Is Not Appropriate

Petitioner has never before filed a Patent Office challenge to the '170 patent. Patent Owner may argue the '170 patent is already at issue in proceedings initiated by different parties. (§II.) The facts here, however, do not support a discretionary denial.

The Board has set forth seven factors in determining whether to exercise discretion to deny a petition due to an earlier proceeding. *Gen. Plastic Indus. Co., Ltd. v. Canon Kabushiki Kaisha*, IPR2016-01357, Paper 19 at 15-19 (Sept. 6, 2017) (precedential) (“*General Plastic*”).

With respect to the first factor, Petitioner is not a party, real party-in-interest, or privy to the other Patent Office proceedings and this is Petitioner’s first challenge to the '170 patent, which “weighs especially heavily against a discretionary denial.” *Unified Patents, Inc. v. Certified Measurement, LLC*, IPR2018-00548, Paper 7 at 7-8 (PTAB Sept. 5, 2018). Moreover, unlike *Valve*, AWS and Petitioner were sued at different times—four and a half years apart—and the AWS-challenged claims (§II) and the challenged claims in this IPR are not coextensive. *Valve Corp. v. Elec. Scripting Prods., Inc.*, IPR2019-00062, Paper 11 at 10 (PTAB April 2, 2019) (precedential).

The second to fifth factors “bear little relevance” here because Petitioner has never before challenged the ’170 patent. *Unified* at 7-8. Petitioner has no relationship to the earlier challenges to the ’170 patent.

As to the sixth factor, while the Board has finite resources, instituting this petition would be no more a burden on these finite resources than instituting any other petition. And for the seventh factor, there is no readily identifiable roadblock for the Board to issue a final determination within the statutory one-year limit.

C. The Board Should Not Deny Institution Under *Fintiv*

The merits of Petitioner’s arguments are compelling and the evidence in support is substantial. That “alone demonstrates that the PTAB should not discretionarily deny institution under *Fintiv*.” (EX1036, 4-5.) In any event, the six *Fintiv* factors do not justify denying institution.

The **first factor** (stay) weighs strongly against denial. Although Google has not yet moved to stay in district court pending IPR, the Northern District of Illinois has stayed a number of cases pending IPR proceedings since 2020. (EX1033, 1, 2, 5, 6, 10.)

The **second factor** (trial proximity) weighs strongly against denial, since “the median time-to-trial is...after the projected...final written decision.” (EX1036, 9.) The median time-to-trial in the Northern District of Illinois is close to 57 months. (EX1035, 47.) The district court case was filed on June 30, 2023,

so trial would occur in April 2028. On the other hand, with this petition filed in June 2024, a final written decision may be expected around December 2025.

Moreover, this petition was diligently filed shortly after Patent Owner served its infringement contentions identifying asserted claims on April 25, 2024. (EX1034, 1.)

The **third factor** (investment in parallel proceedings) also weighs strongly against denial. The district court case is in the early stages. (*Id.*, 1.) The date of claim construction hearing is still to be determined but is likely to be no earlier than April/May 2025. (*Id.*, 2.) Moreover, the district court's discovery process is currently stayed pending resolution of Google's Motion to Dismiss until July 15, 2024. (EX1052, 1; EX1055, 1.)

The **fourth factor** (overlap) also weighs against denial, since Petitioner challenges more claims than Patent Owner may ultimately assert in its final contentions in the district court litigation, i.e., 10 claims per patent and 20 claims overall from the '640, '170, and '978 patents. (EX1053, 10.) *Apple, Inc. v. SEVEN Networks, LLC*, IPR2020-00156, Paper 10 at 17, 21-22 (P.T.A.B. June 15, 2020).

The **sixth factor** (other circumstances) also weighs against denial as the merits of Petitioner's arguments are compelling. While the **fifth factor** (parties) may weigh slightly in favor of denial because the parties are the same as in district

court, that is true in almost every IPR, and under the “holistic view” of whether integrity of the system and efficiency is best served, institution is favored.

Samsung Elecs. Co. Ltd. v. Dynamics Inc., IPR2020-00505, Paper 11 at 15 (Aug. 12, 2020).

XI. CONCLUSION

Petitioner requests institution of IPR for the challenged claims based on the grounds specified in this petition.

Respectfully submitted,

Dated: June 28, 2024

By: /Naveen Modi/
Naveen Modi (Reg. No. 46,224)
Counsel for Petitioner

XII. APPENDIX A (CLAIM LISTING)

Claim 1

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

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

[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.”

Claim 2

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

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.”

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.”

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.”

Claim 6

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

[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;”

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

[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.”

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.”

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.”

Claim 9

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

Claim 10

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

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.”

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.”

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.”

Claim 14

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

Claim 15

[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:”

[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;”

[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;”

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

[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”

[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.”

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.”

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.”

CERTIFICATE OF COMPLIANCE

Pursuant to 37 C.F.R. § 42.24(d), the undersigned certifies that the foregoing
Petition for *Inter Partes* Review of U.S. Patent No. 7,814,170 contains, as
measured by the word-processing system used to prepare this paper, 13,995 words.
This word count does not include the items excluded by 37 C.F.R. § 42.24 as not
counting towards the word limit.

Respectfully submitted,

Dated: June 28, 2024

By: /Naveen Modi/
Naveen Modi (Reg. No. 46,224)
Counsel for Petitioner

CERTIFICATE OF SERVICE

I hereby certify that on June 28, 2024, I caused a true and correct copy of the foregoing Petition for *Inter Partes* Review of U.S. Patent No. 7,814,170 and supporting exhibits to be served via express mail on the Patent Owner at the following correspondence address of record as listed on Patent Center:

Reichman Jorgensen Lehman & Feldberg LLP
100 Marine Parkway
Suite 300
Redwood City, CA 94065

A courtesy copy was also sent via electronic mail to the Patent Owner's litigation counsel at the following addresses:

Christine E Lehman (clehman@reichmanjorgensen.com)
Jaime Francisco Cardenas-Navia (jcardenas-navia@reichmanjorgensen.com)
Gina H Cremon (gcremona@reichmanjorgensen.com)
Khue Hoang (khoang@reichmanjorgensen.com)

Reichman Jorgensen Lehman & Feldberg LLP

Respectfully submitted,

By: /Naveen Modi/
Naveen Modi (Reg. No. 46,224)
Counsel for Petitioner