

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

GOOGLE LLC,
Petitioner

v.

KOVE IO, INC.,
Patent Owner

Patent No. 7,103,640

**PETITION FOR *INTER PARTES* REVIEW
OF U.S. PATENT NO. 7,103,640**

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 '640 Patent.....	4
	2. Skagerwall and Vingralek Qualify as Prior Art.....	6
VI.	LEVEL OF ORDINARY SKILL	7
VII.	OVERVIEW OF THE '640 PATENT	7
VIII.	CLAIM CONSTRUCTION	10
IX.	DETAILED EXPLANATION OF GROUNDS.....	12
A.	Ground 1: Skagerwall in View of Vingralek Renders Claims 10-24 Obvious	13
	1. Claim 10.....	13
	2. Claim 11	46
	3. Claim 12	47
	4. Claim 13.....	48
	5. Claim 14.....	48
	6. Claim 15	49
	7. Claim 16.....	49
	8. Claim 17.....	50
	9. Claim 18	56
	10. Claim 19.....	62
	11. Claim 20.....	63
	12. Claim 21	63
	13. Claim 22.....	63
	14. Claim 23.....	64

15.	Claim 24.....	64
X.	DISCRETIONARY DENIAL IS NOT APPROPRIATE	65
A.	The Board Should Not Deny Institution Under § 325	65
B.	Discretionary Denial Under <i>General Plastic</i> Is Not Appropriate.....	68
C.	The Board Should Not Deny Institution Under <i>Fintiv</i>	69
XI.	CONCLUSION.....	72
XII.	APPENDIX A (CLAIM LISTING)	73

LIST OF EXHIBITS

Ex. 1001	U.S. Patent No. 7,103,640 to Overton et al. (“the ’640 patent”)
Ex. 1002	Declaration of Sandeep Chatterjee, Ph.D.
Ex. 1003	Curriculum Vitae of Sandeep Chatterjee, Ph.D.
Ex. 1004	Prosecution History of U.S. Patent No. 7,103,640
Ex. 1005	WIPO Application No. 1997/043717 to Kahn et al.
Ex. 1006	U.S. Patent No. 6,473,781 to Skagerwall et al.
Ex. 1007	R. Vingralek, <i>Distributed file organization with scalable cost/performance</i> (1993)
Exs. 1008-1010	RESERVED
Ex. 1011	Japanese Patent Application Publication No. H09-231180 to Sato et al.
Exs. 1012-1014	RESERVED
Ex. 1015	M. C. Norrie et al., <i>The Database Research Group at ETH Zurich</i> , SIGMOD Record (March 1994)
Exs. 1016-1019	RESERVED
Ex. 1020	Declaration of Sylvia D. Hall-Ellis, Ph.D.
Ex. 1021	U.S. Patent Application No. 09/111,896
Ex. 1022	<i>Amazon Web Services, Inc. v. Kove IO, Inc.</i> , IPR2020-00276, Paper 23 (June 16, 2020)
Ex. 1023	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. 7,103,640

	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)
Ex. 1024	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)
Ex. 1025	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)
Ex. 1026	Request for <i>Ex Parte</i> Reexamination of U.S. Patent No. 7,103,640 in <i>Ex Parte</i> Reexamination No. 90/014,552 (July 23, 2020)
Ex. 1027	Order Denying Request for <i>Ex Parte</i> Reexamination in <i>Ex Parte</i> Reexamination No. 90/014,552 (Sept. 9, 2020)
Ex. 1028	Request for <i>Ex Parte</i> Reexamination of U.S. Patent No. 7,103,640 in <i>Ex Parte</i> Reexamination No. 90/019,036 (Nov. 19, 2021)
Ex. 1029	Order Granting Request for <i>Ex Parte</i> Reexamination in <i>Ex Parte</i> Reexamination No. 90/019,036 (Dec. 23, 2021)
Ex. 1030	Notice of Intent to Issue <i>Ex Parte</i> Reexamination Certificate in <i>Ex Parte</i> Reexamination No. 90/019,036 (Sept. 22, 2022)
Ex. 1031	Request for <i>Ex Parte</i> Reexamination of U.S. Patent No. 7,103,640 in <i>Ex Parte</i> Reexamination No. 90/019,166 (Feb. 17, 2023)
Ex. 1032	Order Granting Request for <i>Ex Parte</i> Reexamination in <i>Ex Parte</i> Reexamination No. 90/019,166 (Apr. 4, 2023)
Ex. 1033	2020 to 2023 Northern District of Illinois Orders Regarding Motions to Stay
Ex. 1034	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)
Ex. 1035	Federal Court Management Statistics (December 2023)
Ex. 1036	Memorandum by Director Vidal (June 21, 2022)

Exs. 1037-1051	RESERVED
Ex. 1052	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)
Ex. 1053	Northern District of Illinois Local Patent Rules
Ex. 1054	RESERVED
Ex. 1055	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)
Ex. 1056	Joseph B. Greene, <i>Oracle8 Server Unleashed</i> (1998)
Ex. 1057	Joseph B. Greene, <i>Oracle DBA Survival Guide</i> (1995)
Ex. 1058	Betty Salzberg, <i>File Structures</i> (1988)
Ex. 1059	Andrew P. Black & Yeshayahu Artsy, <i>Implementing Location Independent Invocation</i> (1989)

I. INTRODUCTION

Google LLC (“Petitioner”) respectfully requests *inter partes* review of claims 10-24 (“the challenged claims”) of U.S. Patent No. 7,103,640 (“the ’640 patent”) (Ex. 1001) 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 ’640 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 ’640 patent was involved in two ex-parte reexaminations, Nos. 90/014,552 (“the ’552 EPR”) and 90/019,036 (“the ’036 EPR”), and is currently pending in ex-parte reexamination No. 90/019,166 (“the ’166 EPR”).

The ’552 EPR was filed by a third party (Ex. 1026) in July 2020, presenting SNQs on claims 17, 18, and 24 based on different combinations of prior art from the grounds here (*id.*, 21-24; *infra* Sections IV, IX); the request was denied in

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.

September 2020 (Ex. 1027, 1, 38). The '036 EPR was filed by third party Amazon Web Services (“AWS”) (Ex. 1028) in November 2021, presenting SNQs on claims 17, 18, and 24 based on different combinations of prior art from the grounds here (*id.*, 11); the request was granted (Ex. 1029, 21) and the claims were confirmed (Ex. 1030, 6). Petitioner had no involvement in the preparation or filing of either EPR.

The '166 EPR was filed by third party AWS (Ex. 1031) in February 2023, presenting SNQs on claims 17, 18, and 24 based on different combinations of prior art from the grounds here (*id.*, 19; *infra* Sections IV, IX.) Petitioner had no involvement in the preparation or filing of the '166 EPR. Because the request raised substantial new questions of patentability, reexamination was ordered in April 2023. (Ex. 1032, 20.) 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. 7,814,170 (“the '170 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 '640 patent is available for review and Petitioner is not barred or estopped from requesting review on the grounds identified herein.

V. PRECISE RELIEF REQUESTED AND GROUNDS RAISED

Petitioner requests review of claims 10-24 of the '640 patent and cancellation of these claims as unpatentable on the following grounds:

Ground 1: Claims 10-24 are unpatentable under pre-AIA 35 U.S.C. § 103 as being obvious over Skagerwall (Ex. 1006) and Vingralek (Ex. 1007).

As discussed below, Skagerwall and Vingralek qualify as prior art and are properly relied upon for showing unpatentability of the '640 patent.

1. Earliest Effective Filing Date of the '640 Patent

The '640 patent issued from U.S. Patent Application No. 09/661,222, filed on September 13, 2000, which claims priority to provisional application No. 60/153,709 (“the '709 application”) filed on September 14, 1999, and to application No. 09/111,896 (“the '896 application”) filed on July 8, 1998. (Ex. 1001, Cover, 1:6-12.) However, the '640 patent is not entitled to claim priority to the '896 application because the claims of the '640 patent are not fully supported by the '896 application. (Ex. 1002, ¶¶32-36.) Therefore, the earliest possible effective filing date for the '640 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 '640 patent are disclosed in the

² Petitioner does not concede that the '640 patent is entitled to a September 14, 1999 priority date.

'896 application; therefore, none of the claims of the '640 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 '640 patent: limitations [10.c], [17.c], and [18.e]. (Ex. 1002, ¶¶33-36.) Each of these limitations requires that if a data location server does not contain the requested location string/information associated with the identifier/identification string in the data query or data location request, a “redirect message” is transmitted to the client with information that the client may use to calculate a location of a second/different data location server. The '896 application does not provide support for any such redirect message, nor does it provide support for any of the operations associated with a redirect message. At most, the '896 application describes a “server response message[]” that “answers client queries, and sometimes also performs actions on behalf of the client called server actions.” (Ex. 1021, 15:23-26.) Although the '896 application 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

³ Patent Owner acknowledged that of all claims in the '640 patent and the related '170 and '978 patents, only claims 1 and 31 of the '978 patent are allegedly supported by the '896 application. (Ex. 1023, 7 (n.4).)

in claims 10, 17 and 18, as described above. (*Id.*, 15:27-24:24, 29:7-41:10.) The examiner in the '166 EPR likewise found that “at least claims 17 and 18 (the independent claims at issue)...are not supported by the ['896] patent application.” (Ex. 1032, 14.)

2. Skagerwall and Vingralek Qualify as Prior Art

Skagerwall issued from an application filed October 27, 1998 (Ex. 1006, 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 that 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 that 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.)

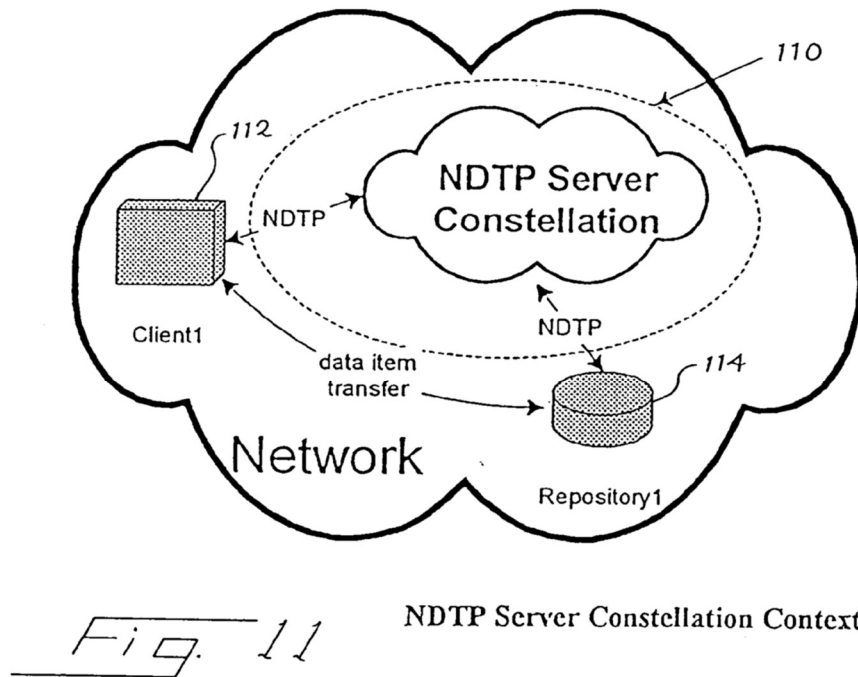
VI. LEVEL OF ORDINARY SKILL

A person of ordinary skill in the art at the time of the alleged invention of the '640 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. (Ex. 1002, ¶¶20-21.)⁴ More education can supplement practical experience and vice versa. (*Id.*)

VII. OVERVIEW OF THE '640 PATENT

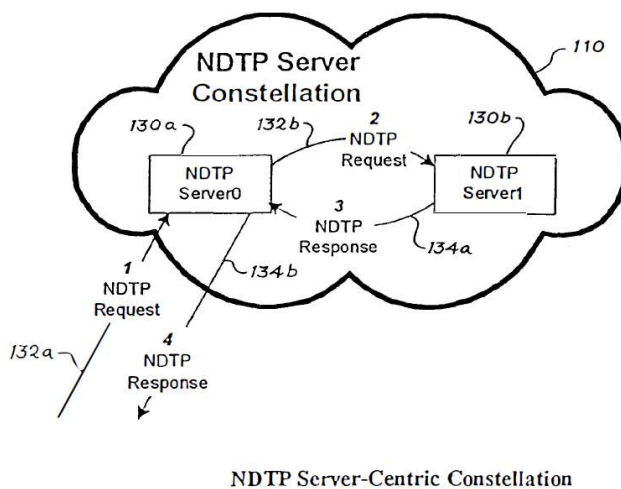
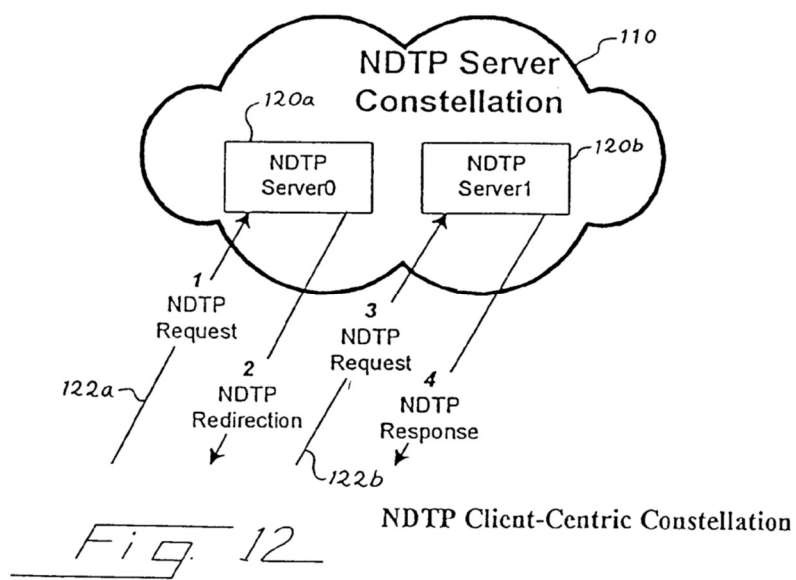
The '640 patent describes storing and retrieving information across a network (Ex. 1001, 1:26-30, 1:56-58) and a network distributed tracking protocol (NDTP) (*id.*, 2:12-14, 4:25-36). Figure 11 shows an NDTP server constellation where a client consults servers that return strings in response to a client request. (*Id.*, 16:53-56.)

⁴ Petitioner submits the declaration of Dr. Sandeep Chatterjee (Ex. 1002), an expert in the field of the '640 patent. (Ex. 1002, ¶¶1-28; Ex. 1003.)



(*Id.*, FIG. 11; Ex. 1002, ¶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.*, 18:11-13), whereas in the server-centric approach, server processing is emphasized (*id.*, 18:37-41).



(*Id.*, FIGS. 12, 13; Ex. 1002, ¶30.)

In both approaches, a client transmits an identifier specifying an entity to a server in a client request. (Ex. 1001, 2:12-44.) If the server contains location information specifying network locations of data associated with the entity, the server returns the location information. (*Id.*; Ex. 1002, ¶31.)

As demonstrated below, all of the features recited in the challenged claims of the '640 patent were known in the prior art before the alleged invention date of the '640 patent. (*Infra* Section IX; Ex. 1002, ¶¶37-43, 22-28; Exs. 1005, 1011, 1056-1059.)

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. (Ex. 1002, ¶¶44-50.)⁵

As shown below, in the AWS IPR, the Board construed a claim term of the related '978 patent (Ex. 1022, 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 '640 patent. (Ex. 1024, 35; Ex. 1025, 19.)

While Petitioner does not acquiesce to those constructions, the prior art mappings in Section IX explain how the '640 patent claims are disclosed and/or rendered obvious if those constructions are applied. (Ex. 1002, ¶¶44-50.)

'640 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" (Ex. 1022, 13.)
"location information"	"information pertaining to one or more locations of data and/or the identities of one or more location servers" (Ex. 1024, 35.)

⁶ Petitioner applies the construction of "identifier" recited in the '978 patent claims to '640 patent claims reciting "identification string"/"identifier string" because the '640 patent uses the three terms interchangeably (Ex. 1001, 2:19-24, 4:7-9) and describes the terms in substantially the same manner as the '978 patent. (*Compare* Ex. 1001, 4:7-9, *with* Ex. 1022, 10-11.)

'640 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” (Ex. 1024, 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” (Ex. 1024, 35.)
“plurality of location servers” ⁷	“location servers in non-hierarchical structures” (Ex. 1025, 19.)

IX. DETAILED EXPLANATION OF GROUNDS

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

⁷ Petitioner also applies this construction to claims reciting “plurality of data location servers.”

A. Ground 1: Skagerwall in View of Vingralek Renders Claims 10-24 Obvious

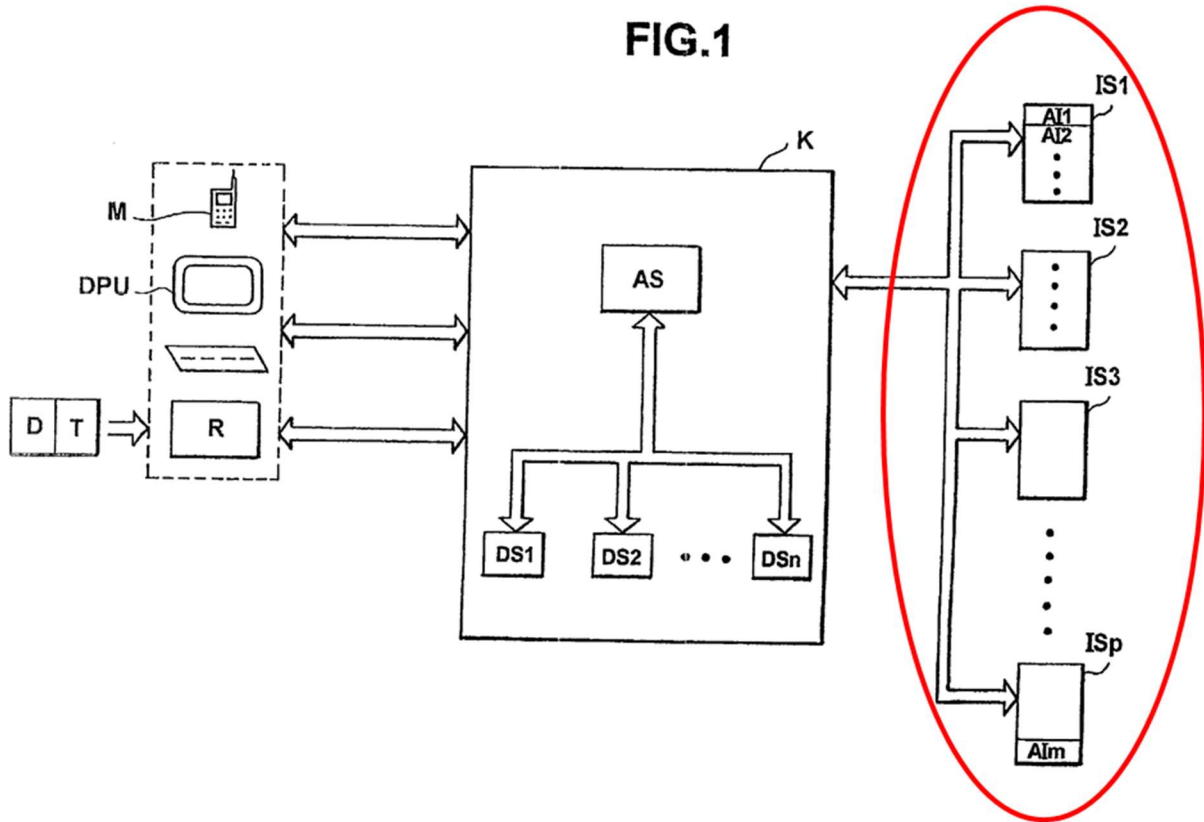
1. Claim 10

i) [10.pre]

To the extent the preamble of claim 10 is limiting, Skagerwall discloses the limitations therein. (Ex. 1002, ¶¶52-56.) To begin, Skagerwall discloses a system that links objects, e.g., products or items in the physical world, to different types of information related to the objects. (Ex. 1006, Abstract (“[c]ommunication system and method for linking each of a plurality of tagged object[s] to information services stored on a network”), 6:45-48 (“[t]he system is arranged to link the plurality of objects, which can, for example, be any products or items of the physical world to various kinds of information related to the objects”).) The system allows for linking objects to information or application data packets (“data”) about those objects, where the application data packets are stored on application servers. (*Id.*, 4:1-5 (“The invention advantageously allows to link...objects of the real world to information services stored on...application servers...”), 4:6-11 (“The...application servers preferably store a plurality of application data packets comprising data related to the services. Each of the plurality of application data packets may be associated with one of the plurality of objects of the real world, i.e., it contains information related to the object.”).) The

application data packets (“data”) may include text, video, or programs. (*Id.*, 4:5-9, 7:13-17; Ex. 1002, ¶52.)

Skagerwall discloses that the application data packets (“data”) are stored in a distributed network of application servers (“stored in a distributed network”). (Ex. 1002, ¶53.) For example, as shown in Figure 1, the application data packets, each identified by an application identifier (AI1-AIm), are distributed across a plurality of application servers (red), each identified by an address (IS1-ISp). (Ex. 1006, 6:34-37 (“The application servers each have an address IS; IS1-ISp and store a plurality of application data packets, each having an application identifier AI1-Aim.”).)



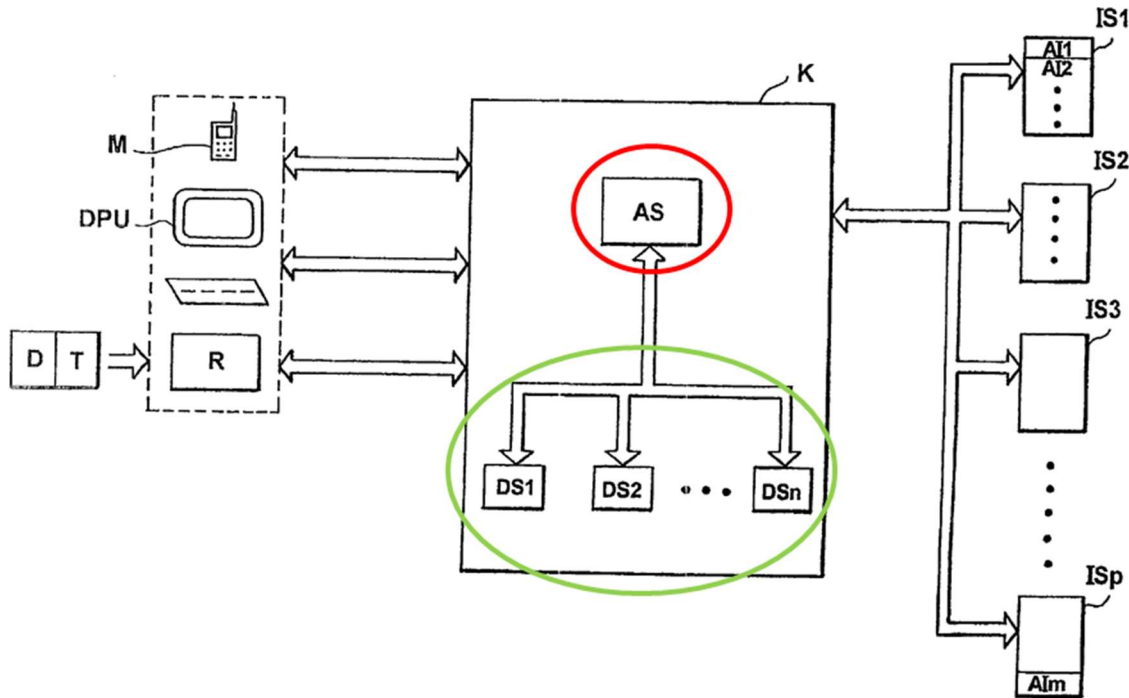
(*Id.*, FIG. 1.) Consistent with Figure 1, a POSITA would have understood that the application servers are arranged in a distributed network because computing resources and data are spread across servers that communicate with one another without going through a centralized point. (Ex. 1002, ¶53.) Indeed, the application servers may be placed at arbitrary locations, similar to intranets, company networks, or the Internet, which a POSITA would have recognized are distributed networks. (Ex. 1006, 7:4-8 (“The plurality of application servers with the addresses IS; IS1-ISp and storing data related to the objects D may be placed at

arbitrary locations in the world, similar to today's computer networks, such as intranets, company networks, or the Internet.”.)

Moreover, Skagerwall discloses a method for “retrieving data location information for data stored in a distributed network.” (Ex. 1002, ¶54.) As shown in Figure 1, the system includes an access server (red) and a network of directory servers (green). (Ex. 1006, 6:32-34 (“The embodiment of FIG. 1 comprises a system kernel K with at least one [access] server AS; AS1-ASr and a plurality of directory servers DS; DS1-DSn.”).)⁸

⁸ A POSITA would have understood that the reference to “server AS; AS1-ASr” as “application server[s]” is a typographical error, as these are referred to as “access server[s]” elsewhere in the specification, while the servers IS; IS1-ISp are referred to as the addresses of “application servers.” (See, e.g., Ex. 1006, 9:1-9.)

FIG.1



(*Id.*, FIG. 1.) Each directory server stores object data, which associates objects with application data packets (“data”), and is involved in the retrieval of application data packets from application servers (“data stored in a distributed network”). (*Id.*, 4:13-16 (“[E]ach of a plurality of directory servers stores part of an arbitrarily scaleable data base containing **object data associating each of the objects with at least one of the application data packets**. The plurality of directory servers serve as [a] key resource in retrieving an application data packet related to an object.”), 8:19-22 (“Each of the directory servers DS; DS1-DSn preferably stores object data related to a small number of the range of available object tags T; T1-Tw.”), FIG. 1; *see also id.*, Abstract (“The system reads an object

tag of one of the objects and retrieves information about the object stored on storage located at arbitrary places.”).) The object data is associated with an object tag and includes application server addresses (“data location information”) indicating the storage location of the application data packets, as well as application identifiers of the packets. (*Id.*, 8:22-27 (“[O]bject data related to an object tag preferably includes application identifiers AI1-AIm of application data packets related to the object as well as information about storage locations of the application data packets, e.g., the addresses of application servers IS; IS1-ISp storing the application data packets.”).) Skagerwall discloses the claimed “data location information” under the AWS Proceeding’s construction of “location information,” i.e., “information pertaining to one or more locations of data and/or the identities of one or more location servers,” because application server addresses identify the location of the application data packets (“data”). (*Id.*; *supra* Section VIII; Ex. 1002, ¶55.)

The application server addresses (“data location information”), which indicate the locations of application data packets (“for data”) stored on the network of application servers (“stored in a distributed network”), may be retrieved (“retrieving data location information for data stored in a distributed network”) as one part of a two-step process for retrieving application data packets associated with an object. (Ex. 1006, 8:28-36.) The first step comprises identifying a

directory server storing object data (including application server addresses) relating to an object, and retrieving those addresses (“retrieving data location information”) associated with an object’s application data packets (“for data stored in a distributed network”). (*Id.*, 8:22-32, 9:5-9 (“[T]he at least one access server AS; AS1-ASr is responsible for **retrieving addresses IS; IS1-ISp of application servers** obtained from at least one of the directory servers DS; DS1-DSn and/or retrieving corresponding application data packets.”), 9:42-48 (“**The obtained addresses IS; IS1-ISp and optionally other information can be returned to the access server AS; AS1-ASr**, which...retrieves the specified application data packets from the identified application servers”), 14:7-11.) The second step involves using the object data to retrieve application data packets (“data”) from the application servers, or alternatively, transmitting the object data directly to the user (e.g., a mobile telephone or data processing unit). (*Id.*, 8:32-36.) Accordingly, Skagerwall discloses this limitation. (Ex. 1002, ¶56.)

ii) [10.a]

Skagerwall discloses this limitation. (Ex. 1002, ¶¶57-65.) **First**, Skagerwall discloses “data associated with an identification string.” Skagerwall discloses a reader that reads an object tag (e.g., a barcode) (“identification string”) and transmits it to the system kernel. (Ex. 1006, 6:41-43 (“A plurality of objects D is marked with a plurality of tags T....A reader R is provided for reading data of an

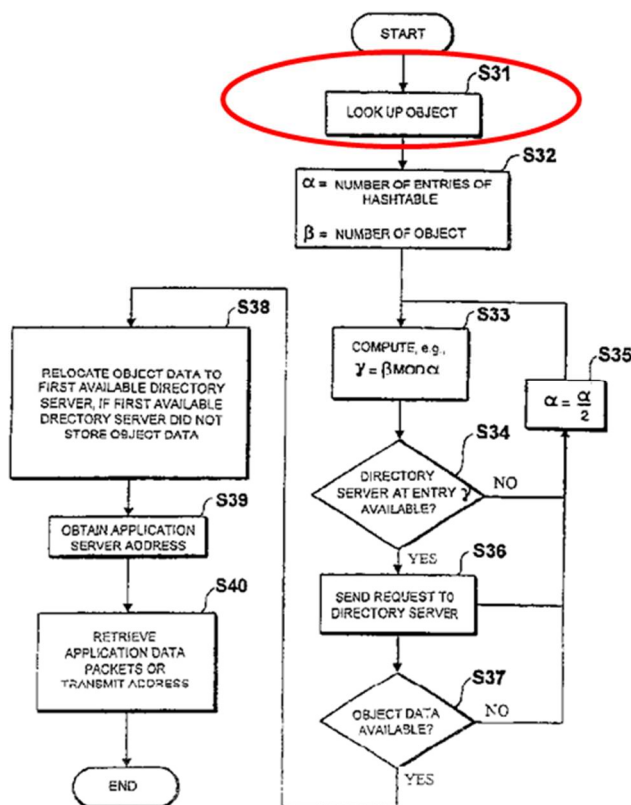
object tag and transmitting same [to] the system kernel K.”), 7:22-23 (“The reader may be a bar code reader or any other reading device.”).) A POSITA would have understood that an object tag such as a barcode is a “string,” because a barcode includes a string of characters, e.g., a human readable part or Arabic numerals below the bars and/or the bars themselves readable using a barcode reader. (Ex. 1002, ¶57.) A POSITA would have further understood that an object tag is an “identification string” because the tag identifies a given object. (*Id.*; Ex. 1006, 2:53-54 (“a plurality of objects (D) each identified by an object tag (T)”)).) Further, as described for limitation [10.pre], the object tag (“identification string”) is associated with the object’s application data packets (“data”) because the object tag is associated with the application server addresses, which provide the location of the application data packets associated with the tagged object. (*See* Section IX.A.1.i.)

Skagerwall discloses that the object tag is an “identification string” under the AWS IPR construction of related term “identifier,” i.e., “a unique encoding that identifies an individual entity, and with which zero or more data location specifiers are associated in a location server.” (*Supra* Section VIII; Ex. 1001, 4:7-9.) A POSITA would have understood that the object tag (e.g., a barcode) is an encoding that uniquely identifies an object (i.e., an individual entity) because, as discussed below, the tag is included in a query to identify the specific object for which

information is sought. (Ex. 1002, ¶58.) Further, as discussed below, zero or more application server addresses (data location specifiers, and as addressed below, the claimed “location string”), which specify the location of application data packets (“data”), are associated with the object tag in a directory server (i.e., a location server, as construed in the AWS Proceeding).

Second, Skagerwall discloses “receiving at a first client a data query for retrieving data associated with an identification string.” (Ex. 1002, ¶59.) Skagerwall discloses that the system kernel includes an access server (“first client”), which receives an information request (“data query”) to retrieve application data packets (“data”) about an object (“receiving at a first client a data query”), as shown in step S31 of Figure 3 below, where the request at least includes an object tag (“identification string”). (Ex. 1006, 9:1-5 (“[A]t least one access server AS; AS1-ASr...is responsible for receiving a request for information about an object including an object tag T; T1-Tw read by the reader R.”), 13:23-26 (“In a step S31 the system kernel K receives a request from a user, requesting information related to an object. A request at least includes an object tag T; T1-Tw of an object D...”), 13:29-30 (“Preferably, the request is received by one of the access servers AS; AS1-ASr.”), FIG. 3; *see also id.*, FIG. 2a (displaying in step S24 a lookup request for object data related to an object tag), 12:27-32 (describing the lookup request in step S24 of Figure 2a).)

FIG.3



(*Id.*, FIG. 3.)

Upon receipt of this request for application data packets associated with an object (“data query”), which includes the object tag (“identification string”), the access server (“first client”) performs operations (e.g., those involving directory servers) to identify application servers storing application data packets (“data”) related to the tagged object, retrieve the packets from the application servers, and supply it to the user. (*Id.*, 7:55-61, 9:5-13, 9:22-27.) Thus, Skagerwall discloses

“receiving at a first client a data query for retrieving data associated with an identification string.” (Ex. 1002, ¶60.)

Skagerwall discloses the claimed “first client” under the AWS Proceeding’s construction of “client,” i.e., “a network-attached component (which may be software or hardware) that initiates update or lookup of identifier/location mappings from a location server with location request messages.” (*Supra* Section VIII; *see also* Ex. 1001, 4:11-14.) In particular, a POSITA would have recognized that the access server (“first client”) is a hardware and/or software component attached to a network given that Skagerwall explains that all components in the system may be “connected via a network N for communication with each other.” (Ex. 1002, ¶61; Ex. 1006, 16:34-36.) As further discussed for limitation [10.b], the access server sends a request (i.e., location request message) to a directory server (i.e., location server) to initiate lookup of mappings of object tags (“identification string,” i.e., identifiers) to application server addresses, which provide location information about application data packets. (*See* Section IX.A.1.iii.)

Third, Skagerwall discloses “wherein the data is stored at a data repository in the distributed network.” As discussed for limitation [10.pre], Skagerwall discloses that the application data packets (“data”) associated with an object are stored in a distributed network of application servers. (*See* Section IX.A.1.i.) A POSITA would have further understood that each application server is a “data

repository” because it stores the application data packets (“data”) associated with tagged objects. (Ex. 1002, ¶62.)

Fourth, Skagerwall discloses “wherein a location string associated with the identification string of the data is stored in at least one of a plurality of data location servers.” (*Id.*, ¶63.) As discussed for limitation [10.pre], the directory servers (“at least one of a plurality of data location servers”) store application server addresses (“location string[s]”), which are associated with object tags (“associated with the identification string of the data”). (*See* Section IX.A.1.i; Ex. 1006, 8:19-27.) A POSITA would have understood that application server addresses are “location string[s]” because a server address may be formatted as a string of characters, and also provides the location where the application data packets are stored. (Ex. 1002, ¶63.) For reasons discussed above, the application server address discloses the “location string” (or similar terms) as described in the ’640 patent. (Ex. 1001, 4:9-11; Ex. 1002, ¶63.)

Skagerwall discloses the claimed “data location server[]” under the AWS Proceeding’s construction of “location server,” i.e., “a network-attached component that maintains a set of identifier/location mappings that are modified or returned in response to location request messages from clients.” (*Supra* Section VIII; *see also* Ex. 1001, 4:14-18.) Skagerwall discloses that each directory server is a component attached to a distributed network (i.e., a network-attached

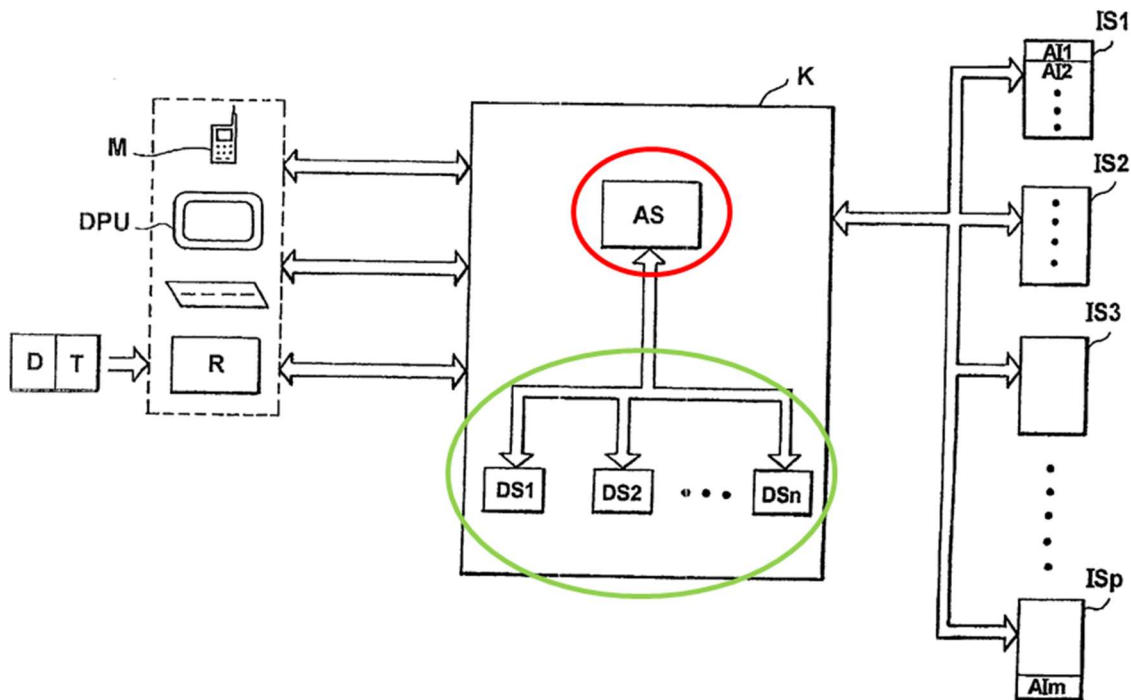
component). (Ex. 1006, 8:15-19 (“the data base of object data is distributed over a plurality of directory servers”), 16:34-36 (all components in the system may be “connected via a network N for communication with each other”), FIG. 1 (depicting a network of directory servers DS; DS1-DSn); *see also* Section IX.A.1.i (discussing Figure 1).) A POSITA would have also understood that each directory server maintains mappings from object tags (i.e., identifiers) to application server addresses (i.e., locations of application servers storing application data packets) because Skagerwall discloses that object tags and associated object data are distributed across directory servers based on a hash function, and that each directory server has a memory means associating object tags with corresponding object data (including application server addresses). (Ex. 1002, ¶64; Ex. 1006, 8:43-63 (“[T]he hashfunction is used to assign object data related to a group of objects to each of the directory servers. Thus, each of the object tags is assigned to a target directory server using the hashfunction....”), 15:15-20 (“[D]irectory server DS1 further comprises address memory means AM for storing a plurality of address data blocks AB1-ABk, each comprising one of the application identifiers AI1-Aim, the corresponding application server address IS; IS1-ISp and, optionally, corresponding object tags T; T1-Tw.”), 15:49-52 (“[T]he directory server may comprise single memory means, in which case all object data related to an object are stored in the provided memory means.”).) Further, Skagerwall discloses that

the directory server returns identifier/location mappings in response to a request from the access server (“first client”) for location information associated with an object tag (i.e., in response to location request messages from a client). For example, as discussed below for limitation [10.b], the application server addresses (i.e., locations) are returned to the access server if the directory server determines it contains the requested location information. (*See* Section IX.A.1.iii; Ex. 1001, 2:36-40, 2:60-65, 6:63-67, 16:53-56.) Skagerwall also discloses that the corresponding object tags (i.e., identifiers) are returned with the application server addresses. (Ex. 1006, 19:42-46 (“means (DS; DS1-DSn; AS; AS1-ASr) for...retrieving the object data corresponding to the object tag”), 19:51-55 (“object data corresponding to an object comprise an object tag, at least one application identifier and the address of the at least one application server storing the at least one application data packet identified by the at least one application identifier”).) Skagerwall further discloses that identifier/location mappings may be updated/modified in response to a request from the access server to add new location information (i.e., another type of location request message from the client). (*Id.*, 12:3-26, 14:29-56; Ex. 1001, 6:46-49.)

Skagerwall discloses the claimed “plurality of data location servers” under the AWS Proceeding’s construction of “plurality of location servers,” i.e., “location servers in non-hierarchical structures.” (*Supra* Section VIII.) A POSITA

would have understood that the directory servers (each a “data location server”) are arranged in a flat, cluster topology, rather than a hierarchical or tree-like structure where servers can only communicate with nodes immediately above or below it in the tree, because as shown below in Figure 1, there is no hierarchy of intermediary servers between the access server (red) and the directory servers (green), and the directory servers themselves are not arranged in a hierarchy. (Ex. 1002, ¶65.)

FIG.1



(Ex. 1006, FIG. 1.) Further, Skagerwall discloses that the hash function is used to distribute object tags and associated object data (e.g., application server addresses) among the directory servers. (*Id.*, 8:15-27, 8:38-54, 8:55-59.) A POSITA would have thus understood that because a hash function is used for distribution across

directory servers, Skagerwall's directory servers are arranged in a non-hierarchical configuration. (Ex. 1002, ¶65.) Indeed, consistent with the above, Patent Owner argued in the AWS Proceeding that the hash function feature requires a non-hierarchical server structure. (Ex. 1025, 15.) Accordingly, Skagerwall discloses this limitation.

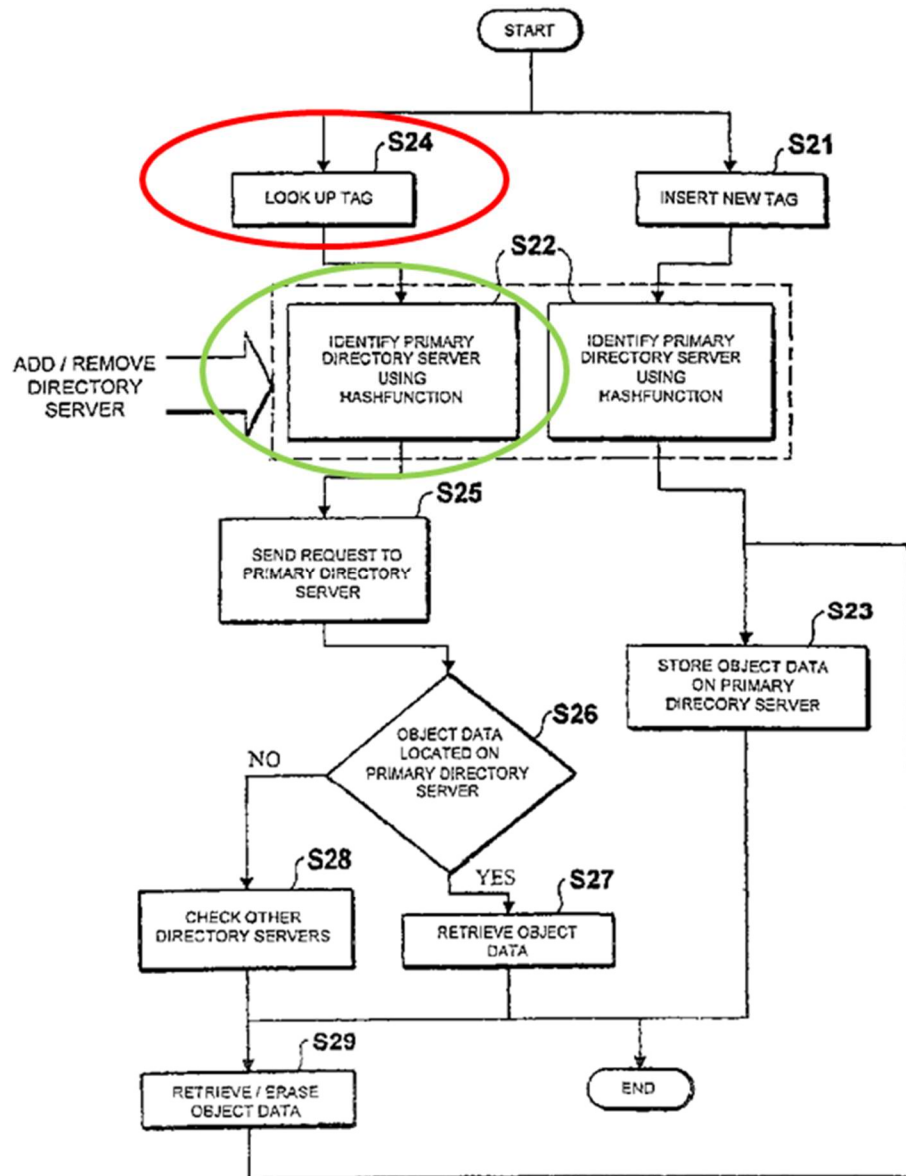
iii) [10.b]

Skagerwall discloses this limitation. (Ex. 1002, ¶¶66-68.) First, Skagerwall discloses “the identification string in the data query.” As discussed for limitation [10.a], Skagerwall discloses that the access server (“first client”) receives an information request (“data query”) for application data packets (“data”) relating to an object, with the object tag included in the request (“identification string in the data query”). (Section IX.A.1.ii; *see also* Ex. 1006, 9:1-5; Ex. 1002, ¶66.)

Skagerwall also discloses “transmitting a data location request from the first client to a first data location server to retrieve the location string associated with the identification string,” where “the data location request includ[es] the identification string.” (Ex. 1002, ¶67.) Skagerwall discloses that upon receipt of the information request (“data query”), the access server (“first client”) performs operations to identify the directory server (“first data location server”) that stores application server addresses (“location string[s]”) identifying the location of application data packets. (Ex. 1006, 9:18-22.) For example, Skagerwall discloses

with respect to Figure 2a that after an information request including an object tag is received in step S24 (red), a primary directory server (“first data location server”) is determined in step S22 (green) using a hash function. (*Id.*, 12:30-37.) The hash function is applied to the object tag from the information request and used as an index into a hash table listing the available directory servers. (*Id.*, 8:38-42 (“The first step of identifying the correct directory server may be performed via a look up in a hashtable H containing addresses or identifiers of all available directory servers DS1-DSn. To facilitate this look up procedure, a hashfunction is used.”), 12:8-12 (“In a step S22 the primary directory server...is computed based on, e.g., a number assigned to the new object tag as well as based on the number of available directory servers”); *see also id.*, 10:28-11:16, 13:15-57 (describing the application of a hash function to an object tag, with the result serving as an index into a table of directory servers).)

FIG.2a

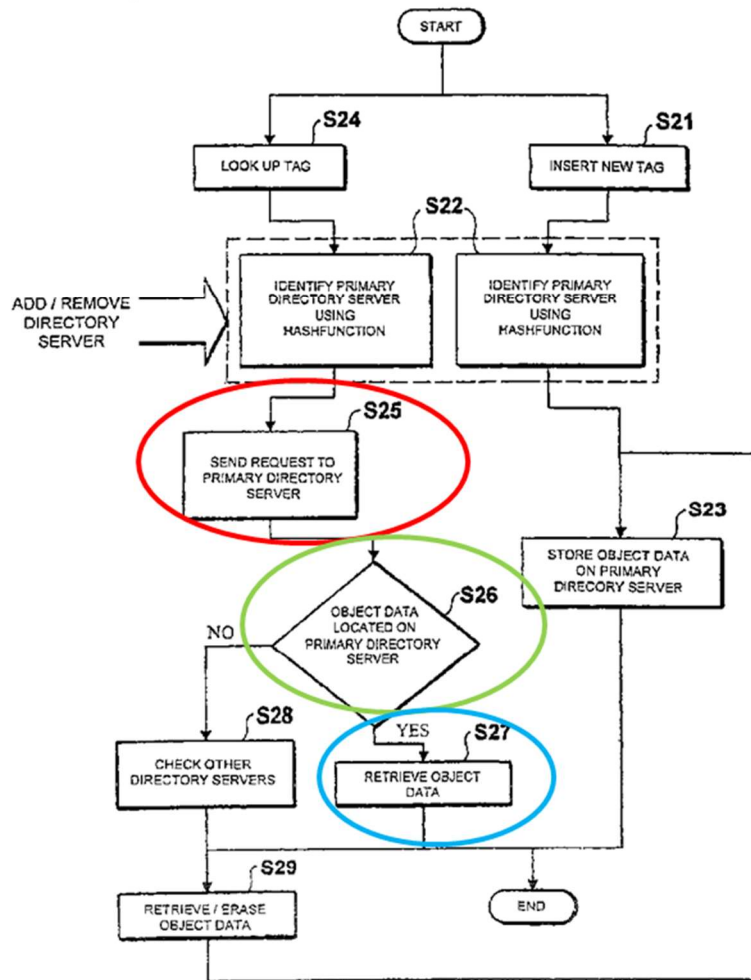


(*Id.*, FIG. 2a.)

Skagerwall explains that the access server (“first client”) then sends a request (“data location request”) to the first directory server (“first data location server”) in step S25 (red in Figure 2a below) to retrieve application server

addresses (“location string[s]”) associated with the object tag (“transmitting a data location request from the first client to a first data location server to retrieve the location string associated with the identification string”), where the request from access server to directory server includes the object tag (“the data location request including the identification string”). (*Id.*, 9:5-13 (“[T]he at least one access server AS; AS1-Asr is responsible for retrieving addresses IS; IS1-Isp of application servers obtained from at least one of the directory servers DS; DS1-DSn and/or retrieving corresponding application data packets....[T]he at least one access server AS; AS1-Asr executes the chain of requests connected to retrieving information from the system with respect to the object tag T; T1-Tw....”), 9:23-25 (“[The access server] send[s] the request including the object tag T; T1-Tw to one of the directory servers DS1; DS2; ...; DSn.”), 12:37-40, FIG. 2a; *see also id.*, 8:22-27 (object data including application server addresses).) In step S26 (green), the primary directory server (“first data location server”) determines whether it contains the application server addresses (“location string[s]”) associated with the object tag. (*Id.*, 12:41-43; *see also id.*, 8:22-27.) If the directory server does contain 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.) Accordingly, Skagerwall discloses this limitation. (Ex. 1002, ¶68.)

iv) [10.c]

v) [10.d]

vi) [10.e]

Skagerwall in view of Vingralek discloses and/or suggests these limitations.

(Ex. 1002, ¶¶69-81.) First, Skagerwall discloses “c) if the first data location server does not possess the location string...the first client to calculate a location of a

second data location server, wherein the second data location server contains the location string; d) calculating the location of the second data location server at the first client; and e) transmitting the data query from the first client to the second data location server.” As discussed for limitation [10.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 and the hash table. (Section IX.A.1.iii.) As further discussed therein, if in step S26 the primary directory server (“first data location server”) determines that it contains the requested application server addresses (“location string[s]”) associated with an object tag, then those addresses are retrieved by the access server (“first client”) in step S27. (Section IX.A.1.iii.) However, if the primary directory server (“first data location server”) determines in step S26 that it does not contain the application server addresses (“location string[s]”) related to the object tag (“if the first data location server does not possess the location string”), then the primary directory server calculates the location of a correct directory server using the hash function in step S28 (“calculating the location of the second data location server”) and forwards the request to that server (“transmitting the data query...to the second data location server”). (Ex. 1006, 9:18-28 (“After receiving a request for information including the object tag T, the access server AS; AS1-ASr performs operations to identify the directory server DS; DS1-DSn storing information concerning application data

packets related to the object. This preferably includes sending the request including the object tag T; T1-Tw to **one of the directory servers DS1; DS2; ...; DSn, which in turn, using the hashfunction and hashtable H, determines the directory server for storing the object tag T; T1-Tw and forwards the request to that particular directory server.**”), 12:63-13:5 (“If in step S26 it has been determined that object data related to the object tag is not located at the identified primary directory server, in a step S28, the lookup request is forwarded to other directory servers....[T]he lookup request may be forwarded to a directory server which is determined using the hashfunction....”); Ex. 1002, ¶69.)

Skagerwall discloses that the request may be forwarded to multiple directory servers before reaching the correct directory server. (*Id.*, 12:63-14:14, FIG. 2 (step S28), FIG. 3 (steps S32-S37).) A POSITA would have recognized that for any request for application data packets relating to an object that takes more than one attempt to find a correct directory server, there is some first directory server (“first data location server”) that receives a request but does not have the requested application server addresses, and some second directory server that does have the requested application server addresses (“second data location server contains the location string”). (Ex. 1002, ¶69.)

Skagerwall discloses that, alternatively, the above-described calculation and forwarding process for identifying the “correct” directory server that has the

requested application server addresses (“calculate a location of a second data location server, wherein the second data location server contains the location string”) may be performed by the access server (“first client”) rather than by a directory server. (Ex. 1006, 9:18-31 (“After receiving a request for information including the object tag T, the access server AS; AS1-ASr performs operations to identify the directory server DS; DS1-DSn storing information concerning application data packets related to the object. This preferably includes **sending the request including the object tag T; T1-Tw to one of the directory servers DS1; DS2; ...; DSn, which in turn, using the hashfunction and hashtable H; determines the directory server for storing the object tag T; T1-Tw and forwards the request to that particular directory server.** However it is also possible, that the **access server AS; AS1-ASr** or other components of the system kernel K **perform steps for identifying the correct directory server**”), 10:25-27 (in addition to a directory server, “**other components** of the system kernel K may serve as **means for identifying the correct directory servers.**”), 17:45-47 (“the identifying of **the correct directory server** may also be performed **by the access server.**”), 19:42-46 (“means (DS; DS1-DSn; **AS; AS1-ASr**) for identifying the directory server (DS; DS1-DSn) assigned to the read object tag **based on the hashtable and on the hashfunction**, and retrieving the object data corresponding to the object tag”); Ex. 1002, ¶70.) A POSITA would have understood that since a

directory server may perform steps including identifying the correct directory server, and Skagerwall discloses that the access server may instead perform those steps, then Skagerwall discloses that the access server may be the component that both calculates the location of a correct directory server containing the requested application server addresses, and also forwards the request to that directory server. (Ex. 1002, ¶70; Ex. 1006, 9:18-31.) Thus, Skagerwall discloses “c) if the first data location server does not possess the location string...the first client to calculate a location of a second data location server, wherein the second data location server contains the location string; d) calculating the location of the second data location server at the first client; and e) transmitting the data query from the first client to the second data location server.” (Ex. 1002, ¶70.)

To the extent that Skagerwall does not explicitly disclose “c) if the first data location server does not possess the location string, **transmitting a redirect message to the first client, the redirect message containing information for use by the first client to calculate a location of a second data location server,** wherein the second data location server contains the location string; d) calculating the location of the second data location server at the first client; and e) transmitting the data query from the first client to the second data location server,” a POSITA would have been motivated and found it obvious to implement this feature in view of Vingralek. (*Id.*, ¶71.)

Like Skagerwall, Vingralek relates to storing data in a distributed network. For example, Vingralek discloses a “distributed file organization system” where a file is “organized into buckets that are spread across multiple servers” (Ex. 1007, 4) and clients can modify the file’s distribution by issuing requests to insert/delete/retrieve records for a given key *K* (akin to an identifier of a record) in the file, such that the file grows or shrinks dynamically (*id.*, 9). 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 that corresponds to a hashed key. (*Id.*, 4, 6.) Thus, Vingralek is similar to Skagerwall, which discloses that the database of object data is distributed based on a hash function over a plurality of directory servers, so that the number of objects may be scaled by adding new directory servers. (Ex. 1006, 8:15-19, 8:43-54; *see also* Section IX.A.1.i.) A POSITA would have therefore been motivated to consider the

teachings of Vingralek when implementing Skagerwall's system/method. (Ex. 1002, ¶72.)

Vingralek additionally discloses propagating updates to clients' and other servers' address tables and hash functions in a deferred manner. (Ex. 1007, 6.) To limit communication overhead, clients learn about changes to the hash function and associated 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 hash function and address table. (*Id.*, 5-6; Ex. 1002, ¶73.) Thus, Vingralek discloses updates to a client's stale hash function and address table are deferred until a wrong first server is reached and the location of a second server needs to be calculated.

Vingralek's method of deferred hash function and 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. (Ex. 1007, 9.) Each client and server has its own copy of the address table, but each table may have stale information if changes have been made to the servers since the table was last updated. (*Id.*, 10 ("The client's and the server's perception of the file may or may not coincide with the current picture of the file (i.e., they may have stale information on...the address table).").) Similarly, each client and server may have stale information on the hash function. (*Id.*, 5 ("client's stale information on the hash function"), 9 (describing

the “file level” as a parameter derived from the index to the hash function), 10 (“The client’s and the server’s perception of the file may or may not coincide with the current picture of the file (i.e., they may have stale information on the file level...).”) When a client seeks to retrieve records associated with a given key, it applies its current 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 to that first server. (*Id.*, 9-10.) The first server that receives the key also applies its hash function to the key and identifies in its copy of the address table the server where the key resides. (*Id.*, 10-11.) If the key was indeed sent to the correct server, then that 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 it has identified through applying the hash function, and sends a message to the client to update the client’s hash function and address table. (*Id.*; *see also id.*, 25 (referring to “client requests and server responses” as “messages”); Ex. 1002, ¶74.)

The key is eventually received by the correct server and the request from the client is executed. (Ex. 1007, 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 and the parameters for the hash function are recalculated. (*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 and hash function each time a request is forwarded to the next server (until the request reaches a correct server containing the requested information, i.e., the "second data location server") would have gradually brought the client's perception of the address table and hash function up-to-date. (Ex. 1002, ¶75.) Thus, Vingralek 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 hash function and address/hash table, which the client uses when calculating the location of a server. (Ex. 1002, ¶75.)

Additionally, Vingralek discloses that the "most salient feature" of its approach, where a client's hash function and table are updated in a deferred manner, "is that its communication overhead is largely independent of the number of servers and clients in the system; thus, it is considered a scalable approach." (Ex. 1007, 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; Ex. 1002, ¶76.)

A POSITA would have been motivated to include features similar to those disclosed by Vingralek when implementing Skagerwall by providing a deferred hash function and address/hash table update to the client (for the client to calculate a location of a correct location server). (Ex. 1002, ¶77.) Indeed, Skagerwall itself explains that, while an access server's hash table is not always up to date, it is preferably not updated every time there is a change to the hash table, so as to conserve resources. (Ex. 1006, 10:12-20.) As discussed above, Vingralek's method of updating the client's hash function and address table only when a request reaches the wrong server (i.e., in a deferred manner) is beneficial because it limits communication overhead, and improves the scalability of the network because communication overhead is independent of number of servers and clients in the system. (Ex. 1007, 5.) Thus, a POSITA would have found it beneficial to include Vingralek's method of deferred hash function and table updates, via a redirect message, when implementing Skagerwall's system/process. (Ex. 1002, ¶77.) This is particularly true because Skagerwall discloses that "[i]ntermediate steps" may be implemented when relaying an information request to a correct server (Ex. 1006, 9:39-41), but does not explicitly disclose the specific processes, which Vingralek teaches. Such a mechanism would be consistent with Skagerwall's goal of providing a system that is both scalable (*id.*, 4:12-15, 11:20-23) and has the flexibility to implement processing steps at the client side (*id.*,

9:23-31, 10:25-27, 17:45-47, 19:42-46). As such, a POSITA would have been motivated to include features similar to Vingralek's "redirect message" features when implementing Skagerwall.

A POSITA would have been motivated to implement features similar to Vingralek's "redirect message" in Skagerwall's system such that the access server (rather than a directory server) is the component that calculates the location of the correct directory server (i.e., using the information provided by the "redirect message"). A POSITA would have understood that such an implementation is a design choice based on system efficiency/configuration and similar considerations. (Ex. 1002, ¶78; Ex. 1006, 5:28-37, 10:15-20, 17:1-5, 18:54-58.) For example, for systems where a lower processing burden on the directory servers is desired, it would have been advantageous to implement the system such that the access server ("first client") is used to calculate/identify the correct directory server ("second data location server") and transmit the information request to the correct directory server ("transmitting the data query from the first client to the second data location server"). A POSITA would have understood that this implementation, where the access server calculates a correct directory server and forwards the request to that server (e.g. as disclosed by Skagerwall), would have been one of only several options, i.e., calculating and forwarding at either the access server or the directory server that processes the information request. Thus, transmitting a redirect

message with information *for the access server* (“first client”) to calculate the location of a 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).

A POSITA would have also had a reasonable expectation of success combining the teachings of Skagerwall and Vingralek. (Ex. 1002, ¶79.) As discussed above, both references are in the same field of art. Further, Skagerwall itself discloses that the hash table on the access server is to be updated if there is a change to the directory servers (Ex. 1006, 10:12-20), and that the hash function may be updated as well because it depends on parameters that change as the hash table changes (*id.*, 13:15-57). Furthermore, Skagerwall discloses an “error message” that is provided if none of the directory servers stores the requested object data (*id.*, 13:5-6, 13:62-65), akin to the claimed “redirect message” that is provided if the location server does not contain the requested information. In fact, a POSITA would have appreciated that the concept of providing a message when requested information is not found was well known. (*Id.*, 2:40-43 (“[A] called address or link for retrieving a certain service from the network may no longer be valid and an error message is returned to the user.”).) Moreover, Skagerwall itself suggests that its system/process is flexible and additional processes may be included, particularly with respect to the process of relaying a lookup request to a

correct directory server. (*Id.*, 9:39-41 (explaining that “[i]ntermediate steps” may be implemented when relaying a lookup request to a correct directory server).)

Additionally, a POSITA would have understood that implementing the combination would have involved modifying the Skagerwall system in a manner that Skagerwall suggests is feasible—implementing a means of occasionally updating a hash table and function in order to save resources. (Ex. 1002, ¶80.) A POSITA would have also had the skill to implement and reasonable expectation of success in achieving such a modification, because it would have involved a combination 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 functions and tables using a redirect message (Vingralek)) to yield the predictable result of a system that updates hash functions and tables in a manner that limits communication overhead and allows for scalability, as discussed above. *KSR Int’l. Co. v. Teleflex Inc.*, 550 U.S. 398, 416 (2007). Accordingly, Skagerwall in view of Vingralek discloses and/or suggests limitations [10.c]-[10.e]. (Ex. 1002, ¶80.)

Consistent with the above and in such a combined system, for example, the access server (“first client”) would apply its current hash function to its own copy of the hash table, and send a request to the primary directory server (“first data location server”), which would apply its current hash function to its own copy of

the hash table in order to determine whether it has the requested location information. (*Id.*, ¶81.) If the primary directory server does not have the requested application server addresses (“location string”) associated with an object tag (“if the first data location server does not possess the location string”), the directory server would send to the access server a redirect message (similar to Vingralek’s disclosed message to update a client’s hash function and table) that includes information for updating its hash function and table, thus allowing the access server to use the hash function to calculate a location of the correct directory server that it subsequently sends the request to (“transmitting a redirect message to the first client, the redirect message containing information for use by the first client to calculate a location of a second data location server, wherein the second data location server contains the location string”). (*Id.*) Once transmission of the redirect message is complete, the access server would then perform the calculation of the location of the correct directory server (“calculating the location of the second data location server at the first client”) and send the request to that correct directory server (“transmitting the data query from the first client to the second data location server”). (*Id.*) A POSITA would have recognized that while the above process may have to be repeated until a request reaches the correct directory server, for any request for application server addresses that takes more than one attempt to find a correct directory server, there is some first directory server (“first

data location server”) that receives a request but does not have the requested application server addresses and thus sends a redirect message to the access server with information the access server uses to calculate the location of some second directory server that does have the requested application server addresses (“second data location server contains the location string”).

2. Claim 11

The Skagerwall-Vingralek combination discloses and/or suggests this limitation. (Ex. 1002, ¶82.) To begin, Skagerwall discloses that the hash table includes an indexed list of the addresses of all available directory servers (“data location server table”). (Ex. 1006, 10:31-38, FIG. 2.) As discussed for limitations [10.c]-[10.e], in the Skagerwall-Vingralek combination, a first directory server that does not have the requested location information would send a message (“transmitting the redirect message”) to the access server (“first client”) in order to update the access server’s hash function and hash table. (*See* Sections IX.A.1.iv-vi.) A POSITA would have understood that, to update the access server’s hash table, the redirect message could include either the specific information to be updated or the entire table for replacing the access server’s table. (Ex. 1002, ¶82.) Thus, transmitting a “data location server table” to the access server (“the first client”) would have been obvious because it would have been one of a “finite number of identified, predictable solutions.” *Perfect Web Techs.*, 587 F.3d at 1331.

Indeed, Vingralek suggests that the entire hash table may be sent to the client in order to update a stale table. (Ex. 1007, 11 (“However, if [key] *K* has been sent to the wrong server, the server...updates the client’s view with...its address table.”).) Thus, the Skagerwall-Vingralek combination discloses and/or suggests “transmitting the redirect message comprises transmitting a data location server table to the first client.” (Ex. 1002, ¶82.)

3. Claim 12

The Skagerwall-Vingralek combination discloses and/or suggests this limitation. (Ex. 1002, ¶83.) As discussed for claim 11, it would have been obvious to implement the redirect message such that it contains a hash table (“data location server table”) that the access server (“first client”) uses to calculate the location of a correct directory server (“calculating the location of the second data location server”) using a hash function. (*See* Section IX.A.2.) As discussed for limitations [10.c]-[10.e], that hash function is commonly known to both the access server (“first client”) and the directory servers (including “first data location server”), since both access and directory servers apply the function in the retrieval process (the access server to determine where to send a request and the directory server to determine if it has location information associated with an object tag), and because updates to the function (e.g., because of changes in the buckets or servers) are shared among the access server and directory servers via the redirect

message so each server has an up-to-date perception of the function (“a function commonly known to the first data location server and the first client”). (Sections IX.A.1.iv-vi.) As further discussed therein, that hash function is applied by the access server to the newly updated hash table and the object tag (“based on the data location server table and the identification string”) in order to determine the location of the correct directory server (“calculating the location of the second data location server”). (See Sections IX.A.1.iv-vi; Ex. 1002, ¶83.)

4. Claim 13

The Skagerwall-Vingralek combination discloses and/or suggests this limitation. (Ex. 1002, ¶84.) As discussed for claim 12, the access server (“first client”) applies a hash function (“wherein the function comprises a hash function and wherein the first client applies the hash function”) to the object tag (“identification string”) and updated hash table (“data location server table”) in order to determine the location of the correct directory server (“obtain the location of the second data location server”). (See Section IX.A.3.)

5. Claim 14

The Skagerwall-Vingralek combination discloses and/or suggests this limitation. (Ex. 1002, ¶85.) As discussed for limitations [10.c]-[10.e], in the Skagerwall-Vingralek combination, if a first directory server that does not have the requested location information, it sends a message (“transmitting the redirect

message”) to the access server (“first client”) in order to update the access server’s hash function (“the redirect message comprises transmitting a function to the first client”) and hash table. (*See* Sections IX.A.1.iv-vi.)

6. Claim 15

The Skagerwall-Vingralek combination discloses and/or suggests this limitation. (Ex. 1002, ¶86.) As discussed for limitations [10.c]-[10.e], the access server performs the calculation of a location of a correct directory server (“calculating at the first client the location of the second data location server”) by applying the updated/transmitted hash function to the object tag (“applying the transmitted function to the identifier string”).⁹ (*See* Sections IX.A.1.iv-vi.)

7. Claim 16

The Skagerwall-Vingralek combination discloses and/or suggests this limitation. (Ex. 1002, ¶87.) Skagerwall discloses that its system may be used for “performing the lookup of a regular Internet address, such as a service via a HTTP request.” (Ex. 1006, 18:5-7.) Skagerwall further discloses that the system kernel, which includes the directory servers (“data location server[s]”), can be connected via the Internet. (*Id.*, 17:9-11 (“The system kernel K is preferably distributed,

⁹ Petitioner assumes the “identifier string” is intended to refer to the “identification string” recited claim 10. (Sections IX.A.1.ii-iii.)

possibly world-wide. Its components can be connected by telecommunication networks or by existing computer networks, such as the Internet.”); *see also id.*, 6:32-34 (“a system kernel K with...a plurality of directory servers DS; DS1-DSn”).) A POSITA would have understood that an Internet address according to the HTTP protocol is a URL, and thus the location of any server connected by the Internet (e.g., the directory servers) may be defined by such a URL. (Ex. 1002, ¶87.) Accordingly, the generation of any location of a directory server in the Skagerwall-Vingralek system—including, for example, the generation of the location of the correct directory server based on the access server’s application of the transmitted hash function (*see* Sections IX.A.1.iv-vi)—may be a URL (“wherein applying the transmitted function generates a URL of the second data location server”). (Ex. 1002, ¶87.)

8. Claim 17

i) [17.pre]

To the extent the preamble of claim 17 is limiting, Skagerwall discloses the limitations therein. (Ex. 1002, ¶88.) As discussed for limitation [10.pre], Skagerwall discloses a communication system (“system”) for retrieving application server addresses (“data location information”) that identify the location of application data packets (“data”) associated with the tagged object, where the

application data packets are stored on a distributed network of application servers (“stored in a distributed network”). (See Section IX.A.1.i; Ex. 1002, ¶88.)

ii) [17.a]

Skagerwall discloses this limitation. (Ex. 1002, ¶89.) As discussed for limitation [10.pre], Skagerwall discloses a network of application servers, each storing application data packets (“data”) associated with an object. (See Section IX.A.1.i.) A POSITA would have understood that each of the application servers is a “data repository” (together “a plurality of data repositories”) because it stores application data packets (“configured to store data”) associated with tagged objects. (Ex. 1002, ¶89.) As discussed for limitation [10.a], Skagerwall discloses that each application server (“each data repository”) contains application data packets (“data”), and that each packet (“data”) is associated with an object, and thus with a unique object tag (“respective identifier string”) that identifies that object (“wherein the data is associated with a respective identifier string in each data repository”).¹⁰ (See Section IX.A.1.ii.)

¹⁰ Skagerwall discloses an “identifier string” under the AWS IPR construction of “identifier” for the same reasons as discussed for limitation [10.a]. (See Sections VIII, IX.A.1.ii.)

iii) [17.b]¹¹

Skagerwall discloses this limitation. (Ex. 1002, ¶¶90-91.) First, Skagerwall discloses “a data location [server] network having a plurality of data location servers, each of the plurality of data location servers containing location strings associated with respective identifier strings.” As discussed for limitation [10.pre], Skagerwall discloses a distributed network of directory servers (“a data location [server] network having a plurality of data location servers”). (See Section IX.A.1.i.) As further discussed, each of the directory servers (“each of the plurality of data location servers”) stores application server addresses (“location string[s]”), which each are associated with an object, and thus with a unique object tag (“associated with respective identifier strings”). (See Section IX.A.1.i.)

Skagerwall also discloses “each of the plurality of data location servers having computer executable code configured to execute the following steps” of limitation [17.c]. (See *infra* Section IX.A.8.iv.) For example, Skagerwall discloses that each of the directory servers “comprises a central processing unit CPU for controlling operations of the directory server.” (Ex. 1006, 15:1-7.) Thus, a POSITA would have understood that a directory server (“data location server”)

¹¹ Petitioner assumes that “data location sewer network” is intended to refer to a “data location server network.”

uses a CPU to execute the operations that the directory server performs based on computer executable code, including the steps recited in limitation [17.c]. (Ex. 1002, ¶91; *see infra* Section IX.A.8.iv.)

iv) [17.c]

The Skagerwall-Vingralek combination discloses and/or suggests this limitation. (Ex. 1002, ¶¶92-95.) First, Skagerwall discloses, “receiving a data location request from a client to retrieve a location string associated with an identification string provided in the data location request.” As discussed for limitation [10.b], Skagerwall discloses that, upon receiving a query for information associated with a tagged object, the access server (“client”) performs operations to identify the directory server (“data location server”) that stores information concerning the location of application data packets related to the tagged object. (*See* Section IX.A.1.iii; Ex. 1002, ¶92.) As further discussed, the access server then sends a request to the identified directory server (“receiving a data location request from a client”) to retrieve application server addresses (“location string[s]”) associated with the object tag (“associated with an identification

string”),¹² where the object tag is included in the request sent from access server to directory server (“identification string provided in the data location request”). (*See* Section IX.A.1.iii; Ex. 1002, ¶92.)

Skagerwall also discloses, “the client to calculate a location of a different data location server in the plurality of data location servers, wherein the different data location server contains the location string.” (Ex. 1002, ¶93.) As discussed for limitations [10.c]-[10.e], the access server (“client”) may itself calculate the location of a correct directory server (“calculate a location of a different location server in the plurality of data location servers”) that contains the requested application server addresses (“location string[s]”). (*See* Sections IX.A.1.iv-vi.)

To the extent that Skagerwall does not explicitly disclose, **“in response to receiving a data location request from a client to retrieve a location string associated with an identification string provided in the data location request, transmitting a redirect message to the client if the identification string is not associated with a location string at the data location server, wherein the redirect message contains information for use by the client to calculate a**

¹² Petitioner assumes that “identification string” is intended to refer to the “identifier string” recited in earlier limitations of claim 17. (*See* Sections IX.A.8.ii-iii.)

location of a different data location server in the plurality of data location servers, wherein the different data location server contains the location string,” a POSITA would have been motivated and found it obvious to implement this feature in view of Vingralek for reasons similar to those as discussed for limitations [10.c]-[10.e]. (Ex. 1002, ¶94; Sections IX.A.1.iv-vi.)

For example, as discussed for limitations [10.c]-[10.e], in the Skagerwall-Vingralek combination, after the primary directory server receives the request (“in response to receiving a data location request from a client”), the directory server applies the hash function to the object tag and to its own copy of the hash table to determine whether it has the requested application server addresses (“location string[s]”) associated with the object tag (“identification string”). (See Sections IX.A.1.iv-vi.) As further discussed, if the primary directory server does not have the requested application server addresses associated with the object tag (“if the identification string is not associated with a location string at the data location server”), the directory server would send a message to the access server (“transmitting a redirect message”) to update the access server’s hash function and hash table, thus allowing the access server to use the hash function to calculate the correct directory server that it may send the request to (“wherein the redirect message contains information for use by the client to calculate a location of a different data location server in the plurality of data location servers, wherein the

different data location server contains the location string”). (See Sections IX.A.1.iv-vi.) Accordingly, the Skagerwall-Vingralek combination discloses and/or suggests this limitation. (Ex. 1002, ¶95.)

9. Claim 18

i) [18.pre]

Skagerwall discloses this limitation for the reasons given for limitation [17.pre]. (Ex. 1002, ¶96; *see* Section IX.A.8.i.)

ii) [18.a]

Skagerwall discloses this limitation for the reasons given for limitation [17.a]. (Ex. 1002, ¶97; *see* Section IX.A.8.ii.)

iii) [18.b]

Skagerwall discloses this limitation. (Ex. 1002, ¶98.) As discussed for limitation [10.a], Skagerwall discloses that the access server (“client”) receives and responds to an information request about an object (“responsive to a data query”) by sending a data location request containing an object tag to a directory server (“to query a data location server”), in order to retrieve application server addresses

(“location information”)¹³ associated with the object tag (“associated with the identifier string”).¹⁴ (See Section IX.A.1.ii; Ex. 1002, ¶98.)

iv) [18.c]

Skagerwall discloses this limitation. (Ex. 1002, ¶¶99-101.) First, Skagerwall discloses “a data location server network comprising a plurality of data location servers, at least one of the plurality of data location servers containing location information associated with the identifier string.” As discussed for limitation [10.pre], Skagerwall discloses a distributed network of directory servers (“a data location server network comprising a plurality of data location servers”). (See Section IX.A.1.i.) As further discussed, each of the directory servers (“at least one of the plurality of data location servers”) stores application server addresses (“location information”), which are associated with the object tag (“associated with the identifier string”). (See Section IX.A.1.i; Ex. 1002, ¶99.)

¹³ Skagerwall discloses “location information” as construed in the AWS Proceeding for the same reasons as discussed for limitation [10.pre]. (See Sections VIII, IX.A.1.i.)

¹⁴ Skagerwall discloses an “identifier string” under the AWS IPR construction of “identifier” for the same reasons as discussed for limitation [10.a]. (See Sections VIII, IX.A.1.ii.)

Skagerwall also discloses “wherein each of the plurality of data location servers comprises computer executable code configured to execute the following steps” of limitations [18.d] and [18.e]. (*See infra* Sections IX.A.9.v-vi.)

Skagerwall discloses that each of the directory servers “comprises a central processing unit CPU for controlling operations of the directory server.” (Ex. 1006, 15:1-7.) Thus, a POSITA would have understood that a directory server (“data location server”) uses a CPU to execute operations based on computer executable code, including the steps recited in limitations [18.d] and [18.e]. (Ex. 1002, ¶100; *see infra* Sections IX.A.9.v-vi.)

Skagerwall also discloses that the steps of limitations [18.d] and [18.e] are “execute[d]...in response to receiving a data location request from the client.” (*See infra* Sections IX.A.9.v-vi.) As discussed for limitations [10.b] and further below, Skagerwall discloses that the access server (“client”) receives an information request about an object and its object tag, and upon receipt, further executes a chain of requests with respect to the object tag to retrieve application server addresses (“location string[s]”) from the directory server (“data location server”). (*See* Section IX.A.1.iii.) As further discussed for limitation [10.b], Skagerwall explains that the access server (“client”) performs operations to identify the directory server (“data location server”) that stores application server addresses indicating the location of application data packets related to the object,

and then requests those addresses by sending a request including the object tag to one of the directory servers (“receiving a data location request from the client”).

(*See* Section IX.A.1.iii; Ex. 1002, ¶101.)

v) [18.d]

Skagerwall discloses this limitation. (Ex. 1002, ¶102.) As discussed for limitation [10.b], the primary directory server (“data location server”) receives a request from the access server (“client”) for application server addresses indicating the location of application data packets related to an object, where that request includes the corresponding object tag (“the identification string provided in the data location request”).¹⁵ (*See* Section IX.A.1.iii.) The primary directory server then determines whether the application server addresses (“location string[s]”) associated with the object tag are located at the primary directory server. (*See* Section IX.A.1.iii.) If the directory server does contain the application server addresses (“if the data location server contains the location string associated with the identification string provided in the data location request”), those addresses are retrieved and subsequent steps for providing the user with the desired application data packets are performed. (*See* Section IX.A.1.iii.) For example, Skagerwall

¹⁵ Petitioner assumes that “identification string” is intended to refer to the “identifier string” recited in earlier limitations of claim 18. (Sections IX.A.9.ii-iv.)

discloses that the obtained application server addresses (“location information”) may be sent to the access server (“client”), which the access server may use to calculate where to find the application data packets (“data”) associated with an object tag (“the data location server transmits location information for use by the client to calculate a location of the data associated with the identification string”). (Ex. 1006, 9:42-48 (“The obtained addresses IS; IS1-ISp and optionally other information can be returned to the access server AS; AS1-ASr, which...retrieves the specified application data packets from the identified application servers ...”); *see also* Section IX.A.1.i.)

vi) [18.e]

The Skagerwall-Vingralek combination discloses and/or suggests this limitation. (Ex. 1002, ¶¶103-105.) First, Skagerwall discloses, “the client to calculate a location of a different data location server in the plurality of data location servers, wherein the different data location server contains the location string.” As discussed for limitations [10.c]-[10.e], the access server (“client”) may calculate the location of a correct directory server (“calculate a location of a different location server in the plurality of data location servers”) that contains the requested application server addresses (“location string[s]”). (*See* Sections IX.A.1.iv-vi; Ex. 1002, ¶103.)

To the extent that Skagerwall does not explicitly disclose, **“if the data location server does not contain the location string associated with the identification string, the location server transmits a redirect message to the client, wherein the redirect message contains redirect information for use by the client to calculate a location of a different data location server in the plurality of data location servers, wherein the different data location server contains the location string,”** a POSITA would have found it obvious in view of Vingralek. (Ex. 1002, ¶104.)

As discussed for limitations [10.c]-[10.e], in the Skagerwall-Vingralek combination, after the primary directory server receives the request, the directory server applies the hash function to the object tag and to its copy of the hash table to determine whether it has the requested application server addresses (“location string[s]”). (See Sections IX.A.1.iv-vi.) As further discussed, if the primary directory server does not have the requested application server addresses associated with the object tag (“if the data location server does not contain the location string associated with the identification string”), the directory server would send a message to the access server (“the location server transmits a redirect message to the client”) with an updated hash function and hash table (“redirect information”) to update the access server’s hash function and hash table. (See Sections IX.A.1.iv-vi.) This step would allow the access server to use the contents

of the redirect message (i.e., the updated hash table and hash function) to calculate the location of the correct directory server (i.e., the directory server containing the requested application server addresses) (“wherein the different data location server contains the location string”), that it may send the request to (“wherein the redirect message contains redirect information for use by the client to calculate a location of a different data location server in the plurality of data location servers”). (*See* Sections IX.A.1.iv-vi.) Accordingly, the Skagerwall-Vingralek combination discloses and/or suggests this limitation. (Ex. 1002, ¶105.)

10. Claim 19

The Skagerwall-Vingralek combination discloses and/or suggests this limitation. (Ex. 1002, ¶106.) As discussed for limitations [10.c]-[10.e], in the Skagerwall-Vingralek combination, a hash function is known to both the access server and the directory servers, since both access and directory servers apply the function in the retrieval process and updates to the function are shared among the access server and directory servers (“wherein the client and the plurality of data location servers each comprise a function commonly known to the client and the plurality of data location servers”). (*See* Sections IX.A.1.iv-vi.) As further discussed for limitations [10.c]-[10.e] and claim 11, it would have been obvious to implement the redirect message such that it contains a hash table (“redirect information”) that the access server (“client”) uses to determine a correct directory

server by applying the hash function to the object tag and hash table (“wherein the client is configured to apply the commonly known function to the...redirect information”). (*See* Sections IX.A.1.iv-vi, IX.A.2.)

11. Claim 20

The Skagerwall-Vingralek combination discloses and/or suggests this limitation for the reasons given for claim 11. (Ex. 1002, ¶107; *see* Section IX.A.2.)

12. Claim 21

The Skagerwall-Vingralek combination discloses and/or suggests this limitation. (Ex. 1002, ¶108.) As discussed for claim 19, the hash function known to the access server and directory servers (“wherein the commonly known function comprises a hash function”) is applied by the access server (“client”) to the object tag (“identification string”) and hash table (“data location server table”) in order to determine the location of a correct directory server (“wherein the client is configured to apply the hash function to the identification string and the data location server table to obtain the location of the different data location server”). (*See* Section IX.A.10.)

13. Claim 22

The Skagerwall-Vingralek combination discloses and/or suggests this limitation. (Ex. 1002, ¶109.) As discussed for claims 14 and 15, the directory

server transmits a message (“redirect message”) that includes an updated hash function (“transmitted function”) for use by the access server (“client”) in calculating the location of the correct directory server. (Sections IX.A.5, IX.A.6.)

14. Claim 23

The Skagerwall-Vingralek combination discloses and/or suggests this limitation. (Ex. 1002, ¶110.) As discussed for claim 15, the access server calculates the location of another directory server (“wherein the client is configured to calculate the location of the different data location server”) by applying the hash function transmitted in the redirect message to the object tag (“by applying the transmitted function to the identifier string”). (Section IX.A.6.)

15. Claim 24

The Skagerwall-Vingralek combination discloses and/or suggests this limitation. (Ex. 1002, ¶111.) Skagerwall discloses that the object tags and associated object data (including application server addresses) are distributed across a plurality of directory servers based on the hash function and hash table, such that each directory server stores object data related to certain object tags. (Ex. 1006, 8:15-27 (“[T]he data base of object data is distributed over a plurality of directory servers....Each of the directory servers...preferably stores object data related to a small number of the range of available object tags...”), 8:38-59 (“[T]he hashfunction is used to assign object data related to a group of objects to

each of the directory servers. Thus, each of the object tags is assigned to a target directory server using the hashfunction.”). In other words, Skagerwall discloses that the object data (including application server addresses (“location information”)) on a given directory server is only a portion of the full database of object data distributed over the directory servers based on the hash function/table. A POSITA would have therefore understood that the application server addresses (“location information”) on a given directory server comprise a portion of the full hash table distributed over the plurality of directory servers (“comprises a portion of a hash table distributed over the plurality of data location servers”) because the application server addresses are associated with a portion of the object tags in the hash table. (Ex. 1002, ¶111.)

X. DISCRETIONARY DENIAL IS NOT APPROPRIATE

A. The Board Should Not Deny Institution Under § 325

Discretionary denial under § 325(d) is not appropriate here because this petition does not present the same or substantially the same prior art or arguments previously presented to the Office during prosecution. (Ex. 1004; Ex. 1001, Cover.) *Advanced Bionics, LLC v. Med-El Elektromedizinische Geräte GMBH*, IPR2019-01469, Paper 6 at 8 (Feb. 13, 2020) (precedential). Moreover, there are material differences between the art/arguments here and those during prosecution (*see, e.g.*, Ex. 1004, 442-449, 492-498, 142, 150, 184, 190), including features

believed to be missing (*id.*, 458-461, 521-526). Discretionary denial is also not warranted in view of the completed *ex parte* reexamination proceedings, since this petition does not present the same or substantially the same prior art or arguments presented in those reexams, and there are material differences between the art/arguments here with those presented therein (*see, e.g.*, Ex. 1026, 21-24; Ex. 1028, 11), including features believed to be missing (Ex. 1027, 21 (“Jaeger...does not appear to teach a redirect message that is sent to a client”), 31 (“None of the [references] are seen as teaching a redirect message”), 36 (“Combining the [references] still fails to provide a teaching of redirect message”); Ex. 1030, 6 (“The cited prior art fail to teach individually or in combination: ‘...transmitting a redirect message to the client...’”)). 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. (Ex. 1032; Section 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 the only ground of the petition. (Ex. 1032, 6-7, 16-

20; Sections IV, IX.) Thus, there is no substantial overlap between the grounds here and those remaining in the pending EPR.

Third, the pending claims of the reexamination and the challenged claims in this IPR are not coextensive. (Sections II, IV; Ex. 1032, 20.)

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. And, given the statutory deadline to file an IPR, Google could not have waited until the conclusion of the reexamination. *See Mueller Sys., LLC v. Rein Tech, Inc.*, IPR2020-00100, Paper 9 at 12-14 (P.T.A.B. May 12, 2020) (“Petitioner would be barred from challenging claims...following the Reexamination Proceeding.”).

Fifth, the reexamination remains at an early stage, which counsels against discretionary denial. *See Mueller* at 13. Accordingly, discretionary denial is not warranted in view of the EPR. *See Pfizer Inc. v. Hoffman-LA Roche Inc.*, IPR2018-01219, Paper 12 at 5-10 (P.T.A.B. Dec. 18, 2018) (instituting IPR where “the grounds differ” from those in the pending EPR).

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

Petitioner has never before filed a Patent Office challenge to the '640 patent. Patent Owner may argue that the '640 patent is already at issue in proceedings initiated by different parties. (Section II.) The facts here, however, do not support a discretionary denial based on those proceedings.

The Board has set forth seven factors in determining whether to exercise discretion to deny a petition under 35 U.S.C. § 314(a) due to an earlier proceeding involving the same patent. *Gen. Plastic Indus. Co., Ltd. v. Canon Kabushiki Kaisha*, IPR2016-01357, Paper 19 at 15-19 (Sept. 6, 2017) (precedential) (“*General Plastic*”). The *General Plastic* factors confirm that this Petition should be considered on the merits.

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 '640 patent, which “weighs especially heavily against a discretionary denial.” *See 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 (Section 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 ’640 patent. *Unified* at 7-8 (“Once resolution of factor 1 indicates that Petitioner had not previously filed a petition against the same patent, factors 2-5 bear little relevance....”) (internal citation omitted). Petitioner has no relationship to the earlier challenges to the ’640 patent. And there is no indication that AWS was aware of the prior art combination at issue here—the Skagerwall-Vingralek combination, which discloses and/or suggests many of the challenged claims. In fact, the district court litigation against Petitioner was filed in June 2023, several months after AWS filed its currently pending reexamination request in February 2023. (Ex. 1031, 234.)

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*.” (Ex. 1036, 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. (Ex. 1033, 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.” (Ex. 1036, 9.) The median time-to-trial in the Northern District of Illinois is close to 57 months. (Ex. 1035, 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, the Board has relied on various justifications, such as diligence. *Verizon Business Network Servs., Inc. v. Huawei Techs. Co.*, IPR2020-01141, Paper 12 (Jan. 14, 2021). This petition was diligently filed shortly after Patent Owner served its infringement contentions identifying asserted claims on April 25, 2024. (Ex. 1034, 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. (Ex. 1052, 1; Ex. 1055, 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. (Ex. 1053, 10.) Thus, there is no complete overlap of claims/issues between the present proceeding and the district court litigation. The Board has found that reduced issues in district court litigation, including the reduced number of asserted claims, “tilt[] in favor of institution.” *Apple, Inc. v. SEVEN Networks, LLC*, IPR2020-00156, Paper 10 at 17, 21-22 (P.T.A.B. June 15, 2020). Moreover, institution here will serve the “interest of overall system efficiency and integrity” because it would allow this proceeding to “continue in the event” the district court litigation “fails to resolve the patentability question[s] presented in” this IPR. *Fintiv*, Paper 11 at 15.

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 10

[10.pre] “A method for retrieving data location information for data stored in a distributed network, comprising the steps of:”

[10.a] “a) receiving at a first client a data query for retrieving data associated with an identification string, wherein the data is stored at a data repository in the distributed network and wherein a location string associated with the identification string of the data is stored in at least one of a plurality of data location servers;”

[10.b] “b) transmitting a data location request from the first client to a first data location server to retrieve the location string associated with the identification string in the data query, the data location request including the identification string;”

[10.c] “c) if the first data location server does not possess the location string, transmitting a redirect message to the first client, the redirect message containing information for use by the first client to calculate a location of a second data location server, wherein the second data location server contains the location string;”

[10.d] “d) calculating the location of the second data location server at the first client; and”

[10.e] “e) transmitting the data query from the first client to the second data location server.”

Claim 11

“The method of claim 10, wherein transmitting the redirect message comprises transmitting a data location server table to the first client.”

Claim 12

“The method of claim 11, wherein calculating the location of the second data location server comprises calculating the location of the second data location server with a function commonly known to the first data location server and the first client and based on the data location server table and the identification string.”

Claim 13

“The method of claim 12, wherein the function comprises a hash function and wherein the first client applies the hash function to the identification string and the data location server table to obtain the location of the second data location server.”

Claim 14

“The method of claim 10, wherein transmitting the redirect message comprises transmitting a function to the first client.”

Claim 15

“The method of claim 14, wherein calculating at the first client the location of the second data location server comprises applying the transmitted function to the identifier string.”

Claim 16

“The method of claim 15, wherein applying the transmitted function generates a URL of the second data location server.”

Claim 17

[17.pre] “A system for retrieving data location information for data stored in a distributed network, the system comprising:”

[17.a] “a plurality of data repositories configured to store data, wherein the data is associated with a respective identifier string in each data repository;”

[17.b] “a data location sewer [sic] network having a plurality of data location servers, each of the plurality of data location servers containing location strings associated with respective identifier strings

and each of the plurality of data location servers having computer executable code configured to execute the following steps:”¹⁶

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

Claim 18

[18.pre] “A system for retrieving data location information for data stored in a distributed network, the system comprising:”

[18.a] “a data repository configured to store data, wherein the data is associated with an identifier string;”

¹⁶ Petitioner assumes that “data location sewer network” is intended to refer to a “data location server network.”

[18.b] “a client responsive to a data query to query a data location server for location information associated with the identifier string;”

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

[18.d] “if the data location server contains the location string associated with the identification string provided in the data location request, the data location server transmits location information for use by the client to calculate a location of the data associated with the identification string;”

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

Claim 19

“The system of claim 18, wherein the client and the plurality of data location servers each comprise a function commonly known to the client and the plurality of data location servers and wherein the client is configured to apply the commonly known function to the location information or redirect information.”

Claim 20

“The system of claim 19, wherein the redirect message comprises a data location server table.”

Claim 21

“The system of claim 20, wherein the commonly known function comprises a hash function and wherein the client is configured to apply the hash function to the identification string and the data location server table to obtain the location of the different data location server.”

Claim 22

“The system of claim 19, wherein the redirect message comprises a transmitted function for use by the client.”

Claim 23

“The system of claim 22, wherein the client is configured to calculate the location of the different data location server by applying the transmitted function to the identifier string.”

Claim 24

“The system of claim 18, wherein the location information comprises a portion of a hash table distributed over the plurality of data location servers.”

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. 7,103,640 contains, as
measured by the word-processing system used to prepare this paper, 13,443 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. 7,103,640 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