

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

MICROSOFT CORPORATION,

Petitioner

v.

SANDPIPER CDN, LLC,

Patent Owner.

Case IPR2026-00174

Patent No. 9,762,692

DECLARATION OF MARK R. LANNING

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

Date:

Dec. 16, 2025

By:

Mark R. Lanning

Mark R. Lanning

TABLE OF CONTENTS

I.	BACKGROUND AND QUALIFICATIONS.....	1
A.	Educational Background	1
B.	Professional Experience	2
II.	MATERIALS CONSIDERED	5
III.	LEGAL STANDARDS.....	5
IV.	TECHNOLOGY BACKGROUND.....	10
A.	The World Wide Web and the Internet	11
1.	Domain Names and URLs.....	11
2.	Web Request and Response	12
3.	Data Resources/Objects.....	12
4.	The HTTP and HTTPS Protocols.....	13
a.	HTTP/HTTPS Message Formats	13
5.	Domain Name Servers and Redirection	15
B.	Content Delivery Networks	16
V.	'692 PATENT	23
A.	Specification	23
B.	Prosecution History	28
VI.	PRIOR ART	32
A.	Seed (U.S. Pub. No. 2006/0015574)	32
B.	Swildens (U.S. Pub. No. 2004/0221019)	34
VII.	LEVEL OF ORDINARY SKILL.....	36
VIII.	CLAIM CONSTRUCTION.....	37
IX.	GROUND S	39
A.	Ground 1: Seed in view of Swildens (Claims 1, 2, 4-15, 17, 18).....	39
1.	Claim 1	39
a.	1[pre]	39

TABLE OF CONTENTS
(continued)

	Page
b. 1[a]	41
c. 1[b]	43
d. 1[c]	46
e. 1[d][i]	52
f. 1[d][ii]	54
g. 1[d][iii]	63
h. 1[d][iv]	64
2. Claim 2	66
3. Claim 4	67
4. Claim 5	68
5. Claim 6	69
6. Claim 7	71
7. Claim 8	71
8. Claim 9	72
9. Claim 10	73
a. 10[pre]	73
b. 10[a]	73
c. 10[b]	73
d. 10[c]	74
e. 10[d]	76
10. Claim 11	76
11. Claim 12	77
12. Claim 13	77
13. Claim 14	78
14. Claim 15	78
a. 15[pre]	78

TABLE OF CONTENTS
(continued)

	Page
b. 15[a]	79
c. 15[b]	80
d. 15[c]	85
e. 15[d]	86
15. Claim 17	86
16. Claim 18	87
B. Ground 2: Swildens in view of Seed (Claims 1, 2, 4-8, 10-15, 17, 18)	87
1. Claim 1	87
a. 1[pre]	87
b. 1[a]	90
c. 1[b]	93
d. 1[c]	98
e. 1[d][i]	103
f. 1[d][ii]	109
g. 1[d][iii]	111
h. 1[d][iv]	114
2. Claim 2	115
3. Claim 4	116
4. Claim 5	116
5. Claim 6	118
6. Claim 7	119
7. Claim 8	120
8. Claim 10	123
a. 10[pre]	123
b. 10[a]	124

TABLE OF CONTENTS
(continued)

	Page
c. 10[b]	124
d. 10[c]	124
e. 10[d]	125
9. Claim 11	126
10. Claim 12	126
11. Claim 13	127
12. Claim 14	128
13. Claim 15	128
a. 15[pre]	128
b. 15[a]	129
c. 15[b]	130
d. 15[c]	136
e. 15[d]	137
14. Claim 17	138
15. Claim 18	138
X. SUMMARY AND ADDITIONAL REMARKS.....	139

I, Mark R. Lanning, do hereby declare:

I. BACKGROUND AND QUALIFICATIONS

1. I have been retained by Petitioner Microsoft Corporation (“Microsoft” or “Petitioner”) to provide my opinions regarding claims 1, 2, 4-15, 17, and 18 of U.S. Patent No. 9,762,692 (“the ’692 patent,” Ex-1001).

2. I am currently the President of Telecom Architects, Inc., and Reticle Consulting, LLC.

3. Available at Ex-1004 is a copy of my current curriculum vitae, which provides further details about my academic and professional background and qualifications.

4. I am being compensated for my services and reimbursed for reasonable expenses that I may incur while working on this matter. I have no financial interest in the outcome of this matter or any other matter that may exist between Petitioners and Sandpiper LLC (“Sandpiper” or “Patent Owner”). This compensation does not depend in any way on the conclusions I reach.

A. Educational Background

5. I received a B.S. in Computer Science from Southern Methodist University (SMU) in 1983.

B. Professional Experience

6. I have over 40 years of experience working in the telecommunications industry that began in the U.S. Army Signal Corp.

7. In 1983, Digital Switch Corporation (DSC), now a part of Alcatel, hired me, where I was a software development manager on the team responsible for updating DSC's PSTN telephone switch into a Mobile Switching Center (MSC) for Motorola to sell as a part of their cellular product offering in the U.S. and many other countries.

8. In 1991, I began working as a consultant to Motorola for its "SuperCell" base station product and as a consultant to British Telecom to upgrade its current analog cellular network, and I was one of the network architects responsible for the design and rollout of its Global System for Mobile Communications (GSM) network known as Cellnet. Beginning in the early 1990s, I was responsible for implementation of the SMS service, including working with suppliers of the SMS Center (SMSC), Mobile Switching Centers (MSCs) and cellular phones to define and roll out the functionality that was to be provided.

9. I was personally involved with Nokia, Ericsson, Motorola and other equipment suppliers in this effort.

10. Since 1995, I have also provided second generation (2G) and third generation (3G) Code Division Multiple Access ("CDMA") network architecture

and equipment design and implementation consulting services to companies such as Sprint, Nextel, Nokia, and Ericsson. While consulting to Nextel, which has since become part of Sprint, as one of the network architects for its iDEN network, one of my responsibilities was to define the network and mobile phone functionality required to support MMS and advanced data communications capability.

11. I am a member of the Institute of Electrical and Electronics Engineers (IEEE), including the IEEE Standards Association. I am also a member of the Association for Computing Machinery (ACM). While employed at DSC, I was a member of the ANSI T1 and T1X1 standard groups responsible for the definition and standardization of the Advanced Intelligent Network (AIN) and Signaling System 7 (SS7) protocol.

12. I am currently the president of two consulting companies: Telecom Architects, Inc. and Reticle Consulting, LLC. Telecom Architects provides consulting services to fixed and wireless telecom service providers and their equipment suppliers. I have been President of Telecom Architects since 1999.

13. My experience specifically relevant to this declaration is that I have worked on the design of packet network topologies and their routing algorithms, the development of computer systems and communication protocols for packet networks since 1976, which is before the current Internet was “invented” in 1983.

Specifically, I began this work when I was assigned to a special project associated with the ARPANET while I was serving in the U.S. Army Signal Corp.

14. Since 1976, I have been involved with the design, development and implementation of many of the different types of computer systems and communication protocols used in the Internet. While working at Digital Switch Corporation (DSC), the engineering team that I managed designed and developed the software and hardware from scratch for at least the TCP/IP, X.25, Session and Transport communication protocols so the DSC telephone switches could connect to packet-based networks. Later, while working for Tandem Computers, my development team wrote the software for the HTTP communication protocol so that our computer systems could provide Internet server functionality.

15. Beginning in mid-2007, I worked with Google regarding the design and development of its extensive Content Delivery Network, named the Google Global Cache (GGC), that was mainly used at the time to cache Search requests/responses. During this time, Google had at least 500,000 servers in its network and used a multi-tier topology. Akamai Technologies, Inc., referenced by multiple prior art references, was one of Google's competitors.

16. As a technical expert on legal cases, I have evaluated various Internet packet routing methods, Internet network topologies, high-speed Internet routers, client/server computer systems, cellular network and Internet interworking

protocols, multi-media streaming across the Internet, specialized systems and methods for redirecting Internet users, and specialized CDNs for supplying real-time marketing data and transfer of very large files.

II. MATERIALS CONSIDERED

17. In developing my opinions below relating to the '692 patent, I have considered the materials cited herein, including those itemized in the table of "Materials Considered" attached as Appendix A to this declaration.

III. LEGAL STANDARDS

18. Counsel has informed me of the legal principles that apply in determining whether patent claims are unpatentable for obviousness. Paragraphs below beginning with "I understand," "I have been informed," or similar phrases capture my understanding of the law as provided to me by counsel. The other paragraphs are my opinion.

19. I understand that a claim may be unpatentable as obvious under § 103(a) if the subject matter described by the claim as a whole would have been obvious to a hypothetical "person of ordinary skill in the art" ("POSITA") in view of a prior-art reference, or combination of prior-art references at the time the claimed invention was made.

20. I understand that "the time the claimed invention was made" is often referred to as the priority or critical date.

21. I understand that, before considering the prior art, the claims are first construed to determine their scope. I understand that the claims are read from the perspective of POSITA at the time of the alleged invention. I understand that the words of a claim are generally given their ordinary and customary meaning as understood by a POSITA in light of the entire patent specification in which they appear, as well as the prosecution history of the patent.

22. I understand, however, that the words of a claim may be given a meaning other than their plain and ordinary meaning if: (1) a patentee expressly and unambiguously defined a term in a way that is different from its plain meaning; or (2) a patentee “disclaims” the full scope of a claim term’s plain meaning, for instance, by making clear and unambiguous statements during prosecution that a claim term does or does not encompass certain features.

23. I understand that, after construing the claims, the teachings of the prior art are considered. I understand that the prior art includes references that were known or used by others in this country, or patented or described in a printed publication in this or a foreign country, before the date the invention was allegedly made by the patent holder. For a printed publication to qualify as prior art, I understand that the Petitioner must demonstrate that the publication was disseminated or otherwise sufficiently accessible to the public. However, I also understand that a POSITA is

assumed to be aware of all relevant prior art in the field of endeavor covered by the patent-at-issue and all analogous prior art.

24. I understand that although the ultimate question of the obviousness of a claimed invention is a legal determination, obviousness is based on several factual inquiries, including: the scope and content of the prior art, the differences between the claimed subject matter and the prior art, the level of ordinary skill in the art at the time of the invention, and any “objective indicia” or “secondary considerations” of non-obviousness, which must all be considered.

25. I understand that, in determining the scope and content of the prior art, a reference is considered relevant prior art if it falls within the field of the inventor’s endeavor, or if it is reasonably pertinent to the particular problem that the inventor was trying to solve. A reference is reasonably pertinent if it logically would have commended itself to an inventor’s attention in considering his problem.

26. I understand that, to assess the differences between prior art and the claimed subject matter, the claimed invention must be considered as a whole. I understand that this involves showing that a POSITA, confronted by the same problems as the inventor and with no knowledge of the claimed invention, would have identified the elements in the prior art and combined them in the claimed manner.

27. I understand that there are several rationales for combining prior-art references. For instance, I understand that it is considered obvious to:

- combine prior-art elements according to known methods to yield predictable results;
- substitute one known element for another to obtain predictable results (also called a “simple substitution”);
- use the prior-art elements in a predictable way according to their established functions;
- apply a known technique to a known device (method or product) ready for improvement to yield predictable results (also called a “known technique”);
- choose from a finite number of identified, predictable solutions, with a reasonable expectation of success (also called “obvious to try”);
- or if there is some teaching, suggestion, or motivation in the prior art that would have led a POSITA to modify a prior-art reference or combine prior-art teachings to arrive at the claimed invention (also called “teaching-suggestion-motivation”).

I understand, however, that a POSITA does not need to have the same motivations or reasons to combine the prior art as the inventors, and can arrive at the claimed invention for completely different reasons than the inventors.

28. However, I also understand that the prior art cannot be modified or combined using “hindsight.” I understand that hindsight refers to situations in which the patent-at-issue is used as a framework to pick and choose pieces from the prior art to arrive at the claims, or adopting rationales identified by the patent-at-issue that those in the art would not have recognized. Instead, I understand that the obviousness inquiry is evaluated from the perspective of a POSITA as of the critical date of the patent-at-issue, with all the prior art before them, to determine whether the POSITA would have independently arrived at the claimed invention without the benefit of the patent-at-issue for guidance.

29. I understand that “objective indicia” or “secondary considerations” of non-obviousness include whether: (1) there was a long-felt need for the claimed invention; (2) the claimed invention has achieved commercial success; (3) there was copying of the claimed invention by others; (4) there were failed attempts by others to make the alleged invention; (5) there was praise of the invention in the field; and (6) there was skepticism the claimed invention could be achieved, among a few others.

30. As of the time that I prepared this declaration, the Patent Owner had not identified any secondary considerations of non-obviousness. Therefore, I reserve the right to elaborate on my understanding of these secondary considerations and to address any secondary considerations that Patent Owner may subsequently identify.

31. Finally, I understand that, in an *inter partes* review proceeding, the obviousness of a claim must be demonstrated by “a preponderance of the evidence,” and that this burden falls on the Petitioner. I understand that the preponderance of the evidence standard requires that a reasonable factfinder could find a material fact is more probable than the nonexistence of that fact. It does not allow for speculation regarding specific facts and is instead focused on whether the evidence more likely than not demonstrates the existence or non-existence of specific material facts. I understand that “preponderance of the evidence” is a lower standard than “clear and convincing evidence” (which requires a fact to be substantially more likely to be true than untrue), or “beyond a reasonable doubt” (which is an exceedingly high standard that I understand is generally reserved for criminal matters).

32. I have applied the “preponderance of the evidence” standard throughout my declaration.

IV. TECHNOLOGY BACKGROUND

33. The purported invention of the '692 Patent relates to content delivery and content delivery networks (CDNs). More specifically, “to frameworks and systems using CDNs.” (Field of the Invention, 1:30-34).

34. This section describes, at a high level, what a CDN is, typical CDN frameworks and components, how a CDN fits in the Internet and the protocols used to implement a CDN.

A. The World Wide Web and the Internet

35. At the inception of the World Wide Web (WWW), aka the Internet, in the early 1990s and up to the early 2000s, when an end user (client) wanted to access specific content over the Internet, they would typically use the browser on their computer to retrieve specific web pages, data, stream music or movies, etc. These requests would use a Uniform Resource Locator (“URL”) that was used to deliver the request to the correct web server.

1. Domain Names and URLs

36. These web servers were typically computer systems that had high-speed communication interfaces and databases for organizing and storing their content. In order to route the client’s request to the correct web server over the Internet that consisted of thousands of potential routers and paths, each of these servers was assigned a unique Internet Domain Name that was included in the URL. We have all used URLs with different Domain Names when we wanted to access many different types of data on the Internet. For example, <http://www.example.com/home> is a URL that consists of three main parts. The first part is called the “scheme” and it describes the protocol being used to access the server. For this example, the “http://” defines that the HTTP communication protocol is to be used (described in more detail below). The second part is the server’s Internet address, e.g., “www.example.com.” where the “www” stands for

world wide web and is a required part of the URL, the “example.com” portion is the domain name of the server. The third part of the URL names a specific resource on the web server, “/home” is referencing the home page on the server called “home.” The web server owners or their representatives typically maintained the content on their server with updates as they were needed.

2. Web Request and Response

37. The diagram below shows, at a very high level, a client (user with a browser) making a request to a web server for a specific resource and the server responds by providing the resource using the HTTP communication protocol to transfer both the request and response.

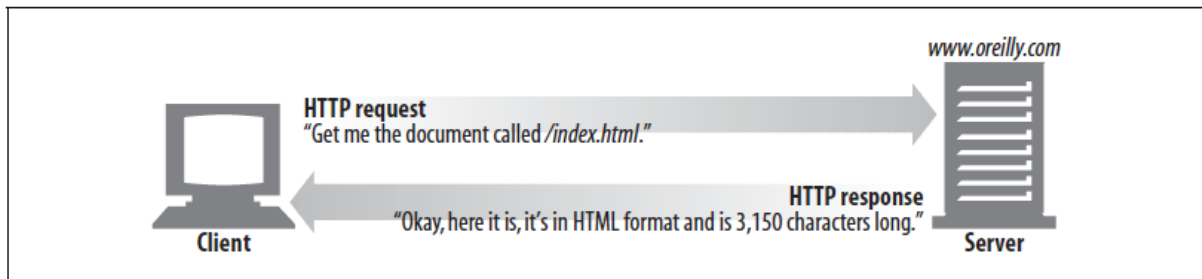


Figure 1-1. Web clients and servers

Ex-1007, 4.

3. Data Resources/Objects

38. Web servers host web resources (also referred to as objects). A web resource is the source of web content. “The simplest kind of web resource is a static file on the web server’s filesystem. These files can contain anything: they might be

text files, HTML files, Microsoft Word files, Adobe Acrobat files, JPEG image files, AVI movie files, or any other format you can think of.” Ex-1007, 4.

4. The HTTP and HTTPS Protocols

39. The Hypertext Transfer Protocol (HTTP) and the HTTP Secure (HTTPS) are request-response protocols that are used by programs to communicate over the Internet. Both are used to provide two-way communication between the web browser (client) and a web server. HTTPS provides essentially the same functionality as the HTTP protocol, but it is used to encrypt the plain text messages used by HTTP to prevent eavesdropping or tampering. The diagram below shows this relationship.

a. HTTP/HTTPS Message Formats

40. The structure of HTTP/HTTPS messages include three parts called: 1) the Start line; 2) the Header fields; and 3) the Body. The start line indicates “what to do for a request or what happened for a response.” The Header fields “follow the start line. Each header field consists of a name and a value, separated by a colon (:) for each parsing.” The Body follows the Header field, is optional and can contain “any kind of data. . . . the body can contain arbitrary binary data (e.g., images, videos, audio tracks, software applications). Of course, the body can also contain text.” Ex-1007, 11.

41. The start line of every HTTP message includes a request command called the “HTTP method.” The “method tells the server what action to perform (fetch a web page, run a gateway program, delete a file, etc.).” Ex-1007, 8. The table below shows five common HTTP methods.

Table 1-2. Some common HTTP methods

HTTP method	Description
GET	Send named resource from the server to the client.
PUT	Store data from client into a named server resource.
DELETE	Delete the named resource from a server.
POST	Send client data into a server gateway application.
HEAD	Send just the HTTP headers from the response for the named resource.

Id., 8-9.

42. Every HTTP response message is returned with a status code. “The status code is a three-digit numeric code that tells the client if the request succeeded, or if other actions are required. A few common status codes are shown in Table 1-3 below.” *Id.*, 9. The 302 Redirect code is used extensively to route requests to the correct server in a CDN network (described below).

Table 1-3. Some common HTTP status codes

HTTP status code	Description
200	OK. Document returned correctly.
302	Redirect. Go someplace else to get the resource.
404	Not Found. Can’t find this resource.

Id., 9.

43. The figure below shows the message details for a typical HTTP transaction between a client that is requesting information about a saw blade from joes hardware that is on special. I note the “Get” method is used in the request and the 200 “Status Code” is used in the response.

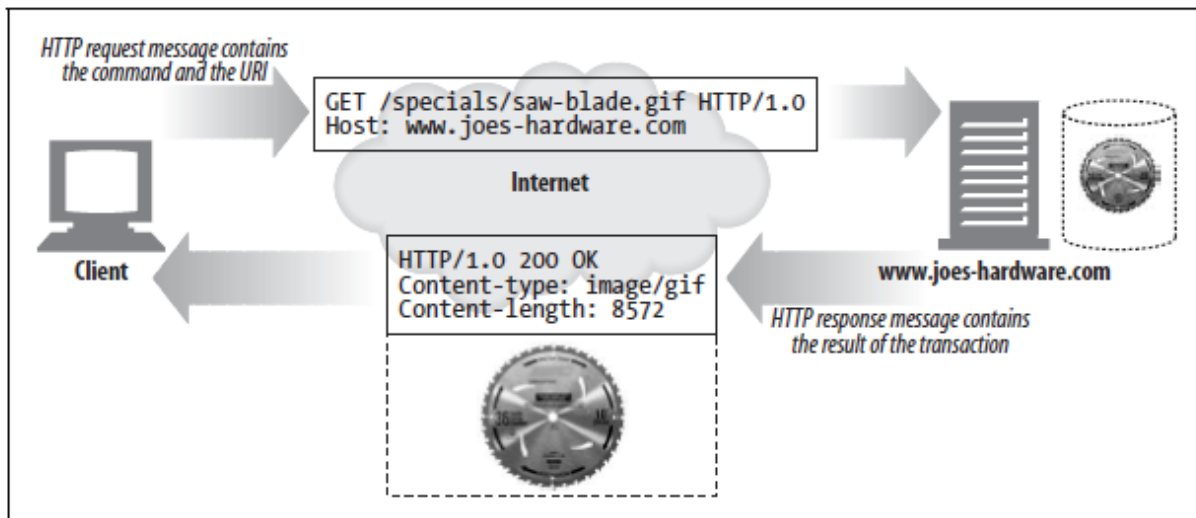


Figure 1-5. HTTP transactions consist of request and response messages
Ex-1007, 8.

5. Domain Name Servers and Redirection

44. The Internet routing protocol uses an IP address, which is a series of numbers, to route packets between the client and server and it cannot use the text www.joes.hardware. A special server in the Internet called a Domain Name Server (DNS) is used to convert these textual domain names that are received from the client into the correct IP addresses. A CDN uses a DNS to convert a requested URL into an IP address for an appropriate edge server, e.g., the CDN edge server that is closest to the requesting client. Thus, the request is not delivered to the real web server.

Instead, the CDN handles the request, which offloads the real web server (also called the origin server in most CDNs).

45. Redirection is an HTTP method that directs the requesting client to use a different URL for its request by returning the HTTP 302 response code described above along with the new URL it is to use. It can be used for many reasons. For Content Delivery Networks, the Redirect method is typically used to direct the client to a different CDN server for many different reasons that are described in the prior art and would have been understood by a person of ordinary skill in the art.

B. Content Delivery Networks

46. As internet usage grew exponentially in the early 2000s, those in the art recognized that “serving Web content from a single location...present[ed] serious problems for site scalability, reliability, and performance,” as servers would overload and crash if they received too many requests at one time. Ex-1008, 50.

47. Additionally, significant physical distance between a requesting network element and responding server often leads to greater latency (or delay) between a request and a response. *See generally* Ex-1009, 45, 51-52. For instance, if a user in Australia tries to stream a movie hosted on a United States-based server, the stream may experience notable delay and buffering as the movie data navigates globe-spanning connections to deliver the movie data to the end user.

48. To solve these problems, companies began building networks specifically intended to distribute content around the world faster while avoiding overloading a content provider's server. These networks would employ "different levels of local, regional, [and] international caches at geographically distributed locations." Ex-1009, 9, 16.

49. CDNs were well-established years before the '692 patent. *See, e.g.*, Ex-1010, Abstract, Figs. 1, 4, ¶¶5-9; Ex-1011, Abstract, 1:15-2:3, Fig. 1. Companies including Akamai, Digital Island, and Speedera were first to implement CDNs. *See, e.g.*, Ex-1010, Cover (22), ¶5 (describing conventional CDNs in the publication of a patent filed in 2001).

50. The content provider's server was typically referred to as an "origin" server, since it was the originating source of all content for a particular content provider. Ex-1009, 5, 37 ("The server where the definitive version of the content resides is called the origin server. It is updated by the content provider."). The content provider's origin server would then connect to a CDN—typically offered by a third-party company—which hosted "*caching and/or replica servers* located in different geographical locations to replicate content." *Id.*, 5. "A cache server makes copies (i.e. caches) of content at the edge of the network in order to bypass the need of accessing origin server to satisfy every content request." *Id.*, 37. These "caching" and/or "replica" servers were also called "edge," "parent," "surrogate," or other

names, depending on their purpose in the network. *See, e.g., id.; see also* Ex-1012, Fig. 2.

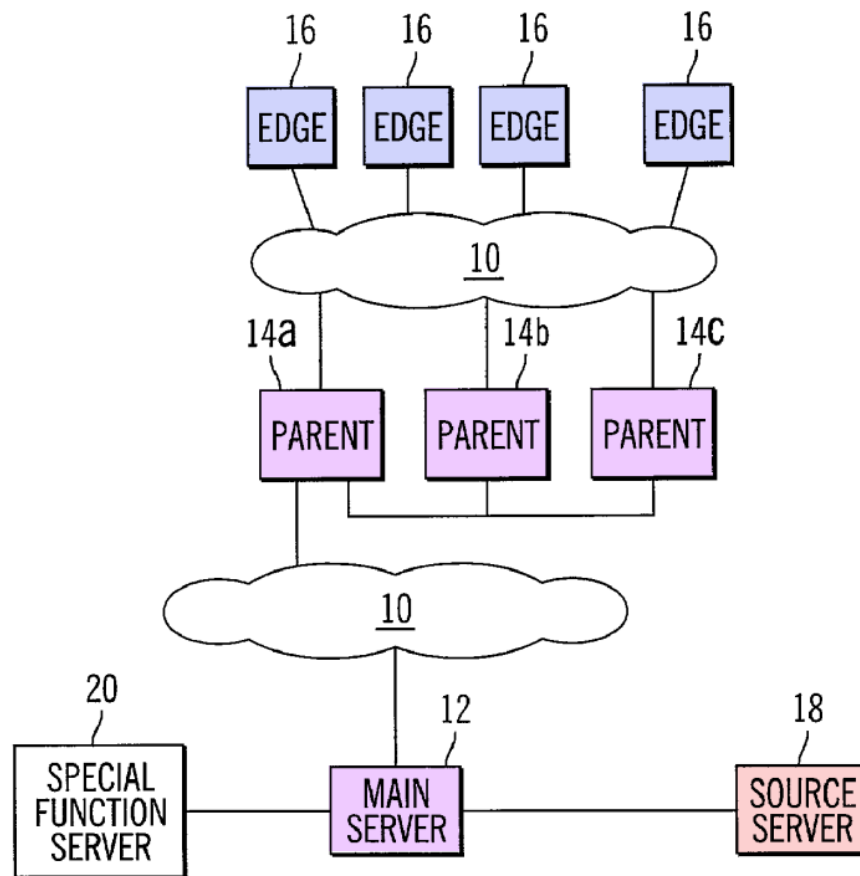


FIG. 2

Ex-1012, Fig. 2 (annotated).

51. To distribute the load of both requests and responses, as well as reduce delivery times, CDNs were often arranged in a hierarchy. Ex-1013, 57; Ex-1009, 9. Fig. 1.1 below is one example of a hierarchical CDN topology. The content provider's origin server (shown in red, below) would sit furthest from the end user in the hierarchy. The CDN would then include, for instance, intermediate servers—

called “surrogate” origin servers and/or parent servers (purple)—that may be geographically distributed and would cache at least some parts of the content owner’s library. These surrogate-origin servers and/or parent servers would then support clusters of also-geographically distributed “edge” or “child” servers located closest to the end user, which would also cache at least part some of the content owner’s library.

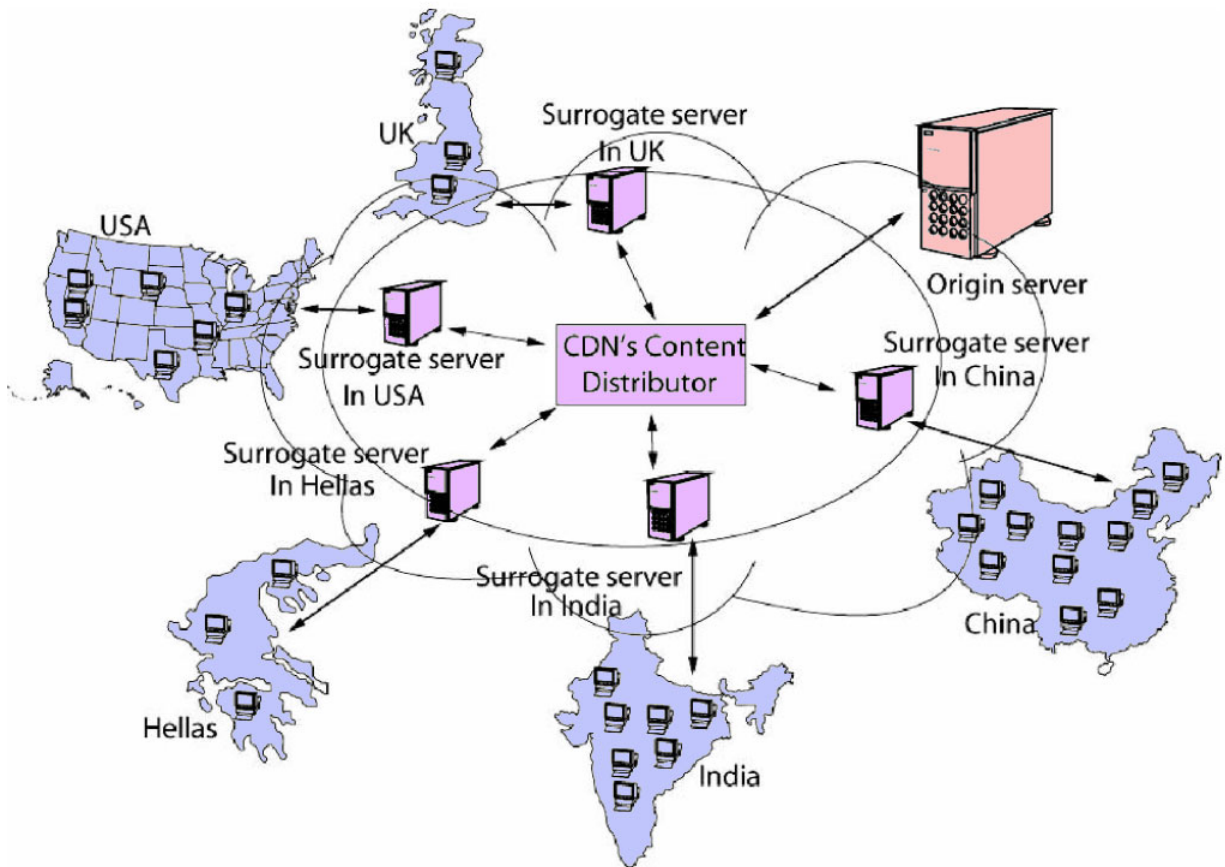


Fig. 1.1 Model of a CDN
Ex-1009, 5 (annotated).

52. Content would be replicated across servers based on a number of factors. Fig. 2.6 below provides one model for how data could be replicated across servers. For instance, content providers could choose between “full-site” caching (all content is hosted by the CDN) or “partial-site” caching (select content is hosted by the CDN). Ex-1009, 44. When “partial-site” caching is used, what content to host in the CDN can be selected based on empirical (heuristic-based manual selection), popularity (where “the most popular objects are replicated” throughout the CDN), objects (selection of specific types of objects to host), or clusters (hosting content that has been correlated based on specified factors). Ex-1009, 44.

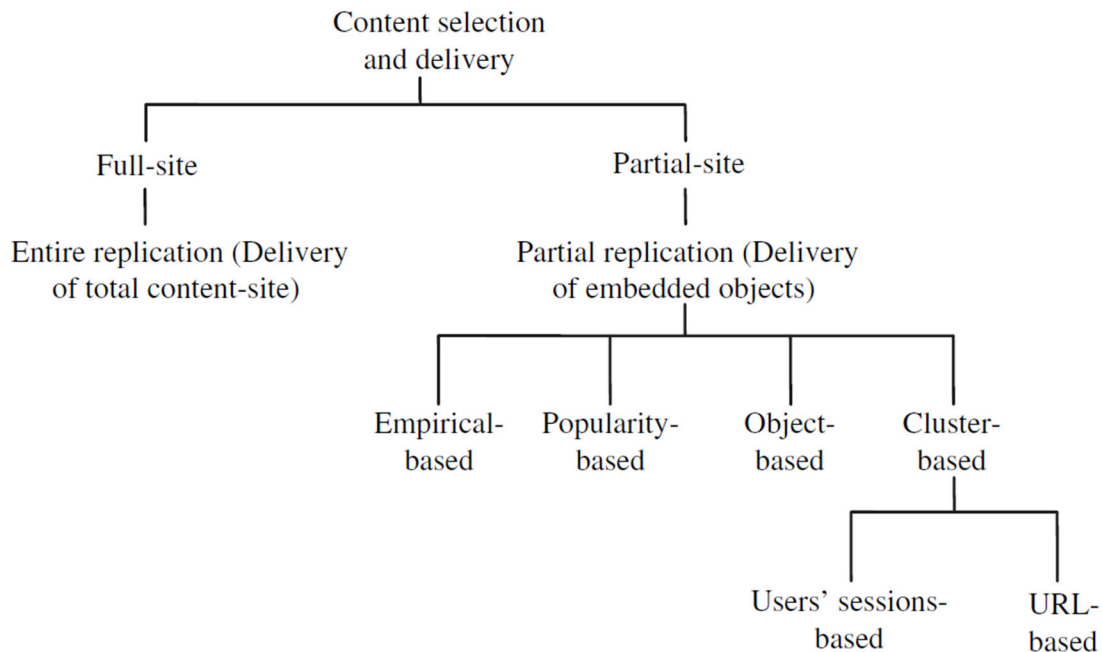
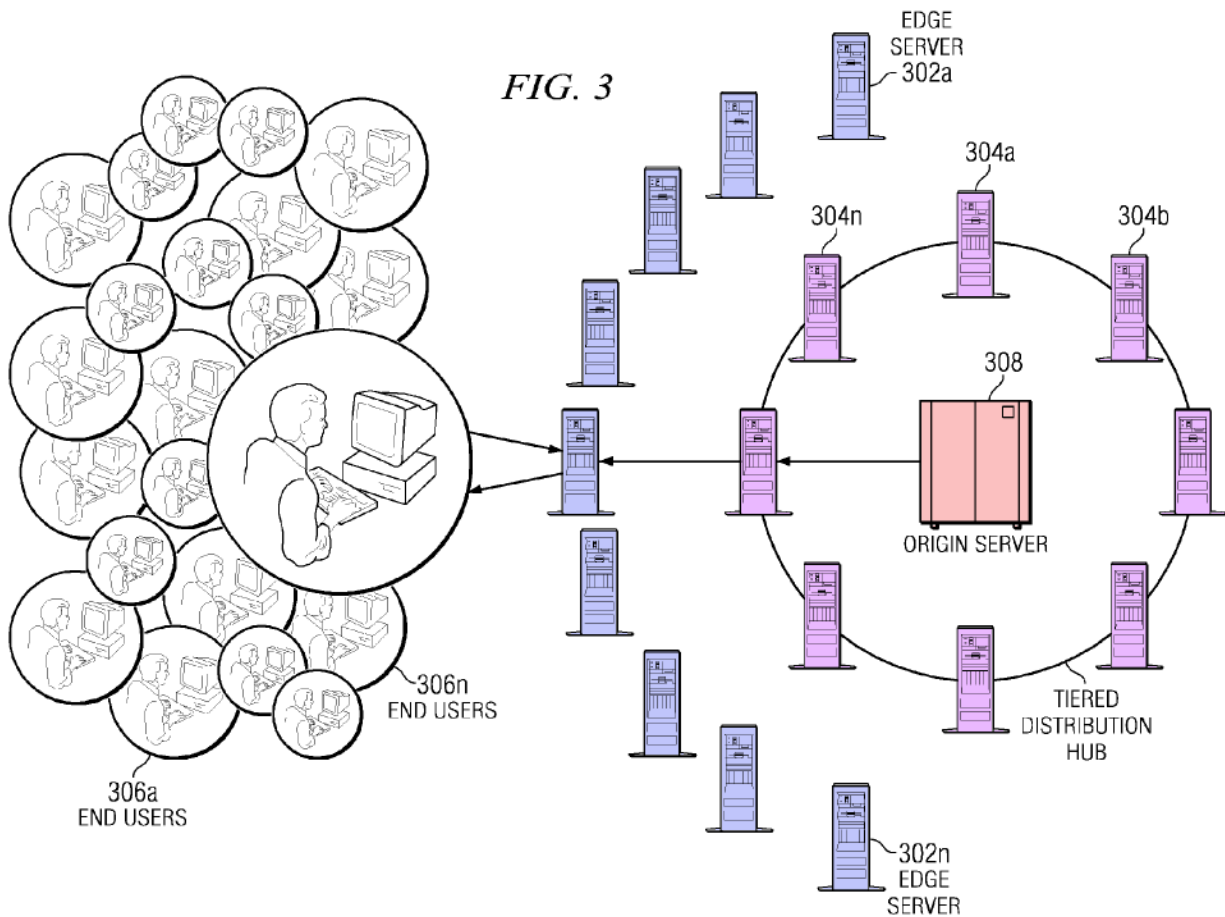


Fig. 2.6 Taxonomy of content selection and delivery

53. Fig. 3 below shows yet a different CDN hierarchical topology that was used. Thus, several different CDN topologies were known, and the exact

configuration, including the number of servers, placement of servers, storage procedures, request-routing procedures, and delivery procedures depended “on the content/services it provides to its users,” and factors such as cost, quality of service, latency, redundancy, and others. Ex-1009, 36; *see also id.*, 37-58, 63-64, 67-69 (describing considerations for each choice and providing example choices by certain CDN providers), 58-59 (summarizing key metrics that impacted design choices).



Ex-1011, Fig. 3 (annotated),¹ 3:5-9, 4:9-35.

¹ All annotations are added unless I note otherwise.

54. In hierarchical systems comprising edge, parent, and origin servers, when an end user requests the delivery of specific content, the request would typically be routed to the optimal edge server that would try to fill the request itself, minimizing the delivery time and avoiding burdening the content owner's origin server. Ex-1009, 6; Ex-1008, 50. But if the edge server could not locate the content in its cache, the edge server could either search for the content on, or redirect the request to, another server (such as a parent or surrogate origin server) that may store the requested content instead. Ex-1013, 57.

55. "To locate content in CDN peering, a centralized directory model, Distributed Hash Table (DHT), flooded request model, or document routing model can be used." Ex-1009, 58. A centralized directory model uses a centralized directory to lookup which server should be storing the requested content. *Id.* This approach, however, "is subject to a single point of failure," and scalability "is limited to the capacity of the directory." *Id.* With DHTs, "peers are indexed through hashing keys within a distributed system," which effectively builds in "load balancing by offloading excess loads to less-loaded peers." *Id.* Flooded request models involve servers that propagate requests to all of their connected peers until one peer in the network responds saying it has the requested content. *Id.* But this approach has drawbacks in that "it generates unnecessary network traffic and hence, it requires enormous bandwidth." *Id.* Finally, the document routing model uses requests that

contain a file identifier, which is forwarded to a peer whose identifier is most similar to the file identifier, which locates the file and provides it to the requesting peer.” *Id.*

56. If the requested content still could not be located, only then would the request be routed to the content provider’s origin server.

V. ’692 PATENT

57. The ’692 patent was originally filed as U.S. Application No. 12/880,324 on September 13, 2010. It claims priority as a “continuation-in-part” to another patent application (No. 12/408,681) filed on March 21, 2009—which issued as U.S. Patent No. 8,930,538—as well as a provisional patent application (No. 61/042,412) filed on April 4, 2008.

58. I have therefore been asked to assume that the ’692 patent is entitled to the earliest possible priority date of April 4, 2008 for this declaration and I have not performed any analysis or opinions regarding this date.

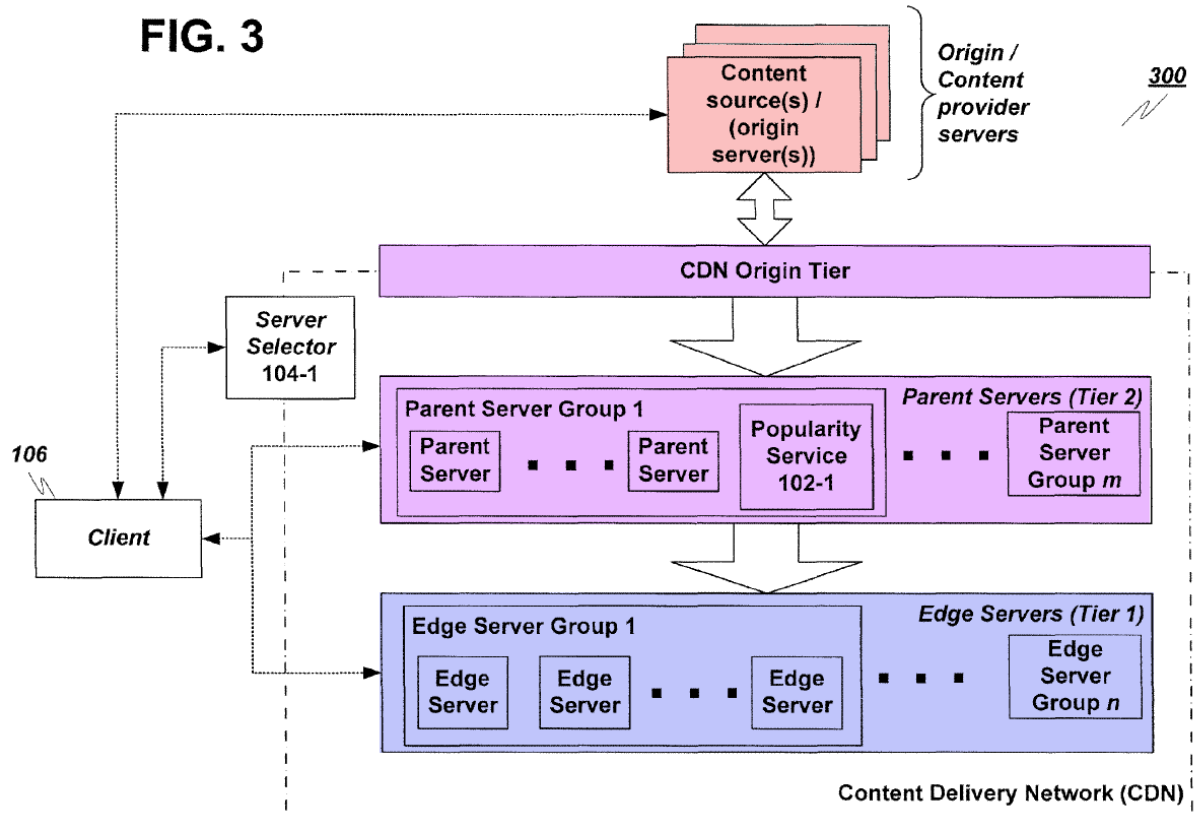
59. I have reviewed the ’692 patent and its file history and provide the following summary.

A. Specification

60. The ’692 patent is primarily concerned with CDNs, Ex-1001, 1:32-34, and specifically how content libraries are stored across the network, *id.*, 2:46-51.

61. The ’692 patent recognizes that CDNs provide servers in several tiers “organized hierarchically.” *Id.*, 2:52-53. The first tier is made up of “edge servers,”

and the second tier is made up of “parent servers.” *Id.*, 2:54-62. Moreover, “[a] content provider’s/customer’s server (or servers) are also referred to as origin servers,” and the CDN “may also include a CDN origin/content cache tier” that “obtain[s] content from the subscribers’ origin servers.” *Id.* 3:65-4:17. I have provided below an annotated version of Figure 3, which shows the hierarchical relationship between the content provider’s/customer’s server furthest from the client device (shown in red), the CDN origin servers that is in an intermediate tier (shown in purple), the CDN parent servers that is also an intermediate tier (also shown in purple), and the CDN edge servers that sit closest to the client (shown in blue):



62. The '692 patent explains that “[c]ertain publishers can have large content libraries in which only a small portion of the content...is popular enough to benefit from serving through a caching CDN, while the majority of the content...is accessed only occasionally.” *Id.*, 2:35-45. “[T]he CDN tracks popularity of certain content, and selectively migrates content toward the edge...as that content becomes popular.” *Id.*, 2:46-51. In other words, often-requested, popular content should be moved to servers closer to the end user for faster delivery, while less popular or unpopular content does not need to be stored right at the edge.

63. The '692 patent also explains that content libraries “may be logically partitioned across two or more servers” in servers other than the edge. *Id.*, 2:63-3:12. “For example, a first portion (or subset) of a content provider’s library may be stored on a first parent server...a second portion may be stored on a second server...and the last third may be stored on a third server” in order to provide “a type of preemptive load-balancing such that certain parent servers are only responsible for handling a pre-defined subset of a content provider’s library.” *Id.*

64. The '692 patent then details the operation of a few embodiments that pertain to the structure and functionality of a CDN based on the purported invention. In one embodiment “a first server in a first tier of servers obtains a request from a client.... [I]f the resource is not available at the first server or at a peer of the first server,” an evaluation is made whether the requested resource is “popular,” and if so, “the first server obtains the resource” from another server—such as a parent server or origin server—“and serves it to the client.” *Id.*, 11:55-12:2.² In other words, if a request comes into an edge server (a first tier server) that does not have the requested content cached, but the requested content is popular, the edge server

² The '692 patent also explains that “[t]he term ‘resource’ is sometimes used...to refer to content.”) *Id.*, 2:33-34.

should have a copy, so it will get a copy of the popular content (to store in its cache) from another server, and also serve a copy to the client.

65. But “if the resource is determined not to be popular...the client is directed to a second server in a second tier of servers.” *Id.*, 12:3-16. In other words, if the requested content is not already stored in cache on the edge server and also determined not to be popular, the edge server does not need to store a copy of it in its cache, so the edge server will direct the client to, e.g., a parent server to provide the client the requested content.

66. I have provided below Figure 7 from the '692 patent, which lays this process out in a flow chart.

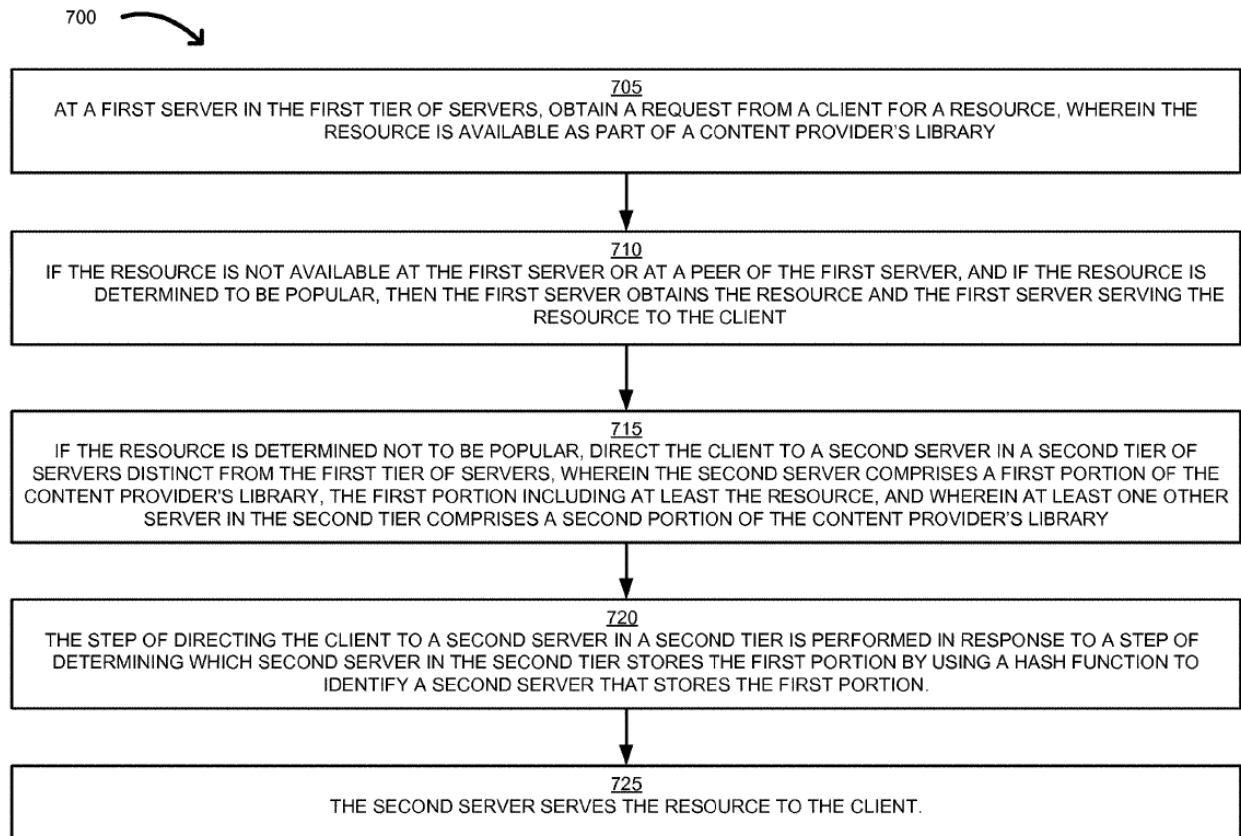


FIG. 7

67. In other embodiments, a resource may designate whether to be served based on popularity or not, and the network can utilize a “popularity service” (e.g., a separate server that evaluates the requested resource’s popularity) to identify where the request will be served from. *Id.*, 5:43-6:24.

B. Prosecution History

68. I have reviewed the ’692 prosecution history and provide the following summary.

69. On September 13, 2010, the Applicant filed an original set of claims that presented three independent claims and nineteen dependent ones. Ex-1002, 27-31

70. The Examiner, however, rejected each of the independent claims as anticipated or obvious. Specifically, the Examiner rejected then-independent claim 12—which aligns with issued independent claim 10—as being anticipated by Outten (U.S. Patent No. 7,024,466). Ex-1002, 245. The Examiner then rejected then-independent claims 1 and 17—which align with issued independent claims 1 and 15, respectively—as being obvious in view of Seed (U.S. Pub. No. 2006/0015574) and Outten. *Id.*, 246.

71. For then-pending independent claims 1 and 17, the Examiner cited Seed as teaching nearly all of the elements of the independent claims, but relied upon Outten to teach “if the resource is determined not to be popular,” directing a client to a second server in a second tier of servers, where the second-tier servers stored (distinct) portions of the content provider’s library. *Id.*, 247. The Examiner also relied upon a third reference, Gupta (U.S. Patent No. 6,212,565) to teach a handful of specific elements, such as partitioning portions of a content provider’s library alphabetically (*id.*, 248), and using a hash as part of the partitioning scheme (*id.*, 249).

72. In response to the Examiner's rejection, the Applicant amended the claims. Specifically, the Applicant clarified in independent claims 1 and 17 that "the second tier is any intermediate tier between the first tier and an origin server," but did not amend independent claim 12 at all. *Id.*, 290-294. Instead, the Applicant argued that Outten does not teach partitioning *distinct* portions of a content library across a tier of servers, and also does not direct, e.g., edge servers to intermediate servers, but rather intermediate servers to edge servers. *Id.*, 295-298.

73. The Examiner, however, maintained the rejection and issued a final rejection. *Id.*, 2112-27.

74. The Applicant then appealed the Examiner's rejection to the U.S. Patent Trial and Appeal Board ("Board"), (*id.*, 2156), largely repeating the arguments it had made to the Examiner regarding Outten's deficiencies (as well as no motivation to combine Seed and Outten) (*id.*, 2180-2191).

75. The Board affirmed the examiner in-part and reversed the examiner in-part.

76. First, the Board noted that, under the broadest reasonable interpretation of the claims (which I understand takes a broader view of the claims than the claim-construction standard applied here and as I summarize above), claims 1 and 17 recite methods that include two conditionals. For claim 1, the two conditionals are: (1) steps to perform if the resource is determined to be popular (e.g., step 1[c] here), and

(2) steps to perform if the resource is determined not to be popular (e.g., steps 1[d] here). *Id.*, 2329-2332. For claim 17, the two conditionals are: (1) if the resource is available at the first server (e.g., step 15[a] here), and (2) steps to perform if the resource is not available at the first server (e.g., steps 15[b]-[c] here). *Id.* In other words, the broadest reasonable interpretation of these claims “encompass[] at least two methods.” *Id.*

77. The Board then sided with the Applicant that Outten did not teach distributing *distinct* portions of a content provider’s library on different servers and thus vacated the anticipation rejection of independent claim 12. *Id.*, 2333-2334.

78. For independent claim 1, however, the Board affirmed the Examiner’s rejection. It noted that Seed discloses most of the claim’s limitations, but unlike independent claim 12, independent claim 1 did not require distributing *distinct* portions of a content provider’s library on different servers, just portions, which Outten taught. *Id.*, 2336-2339. But the Board vacated the obviousness rejection of independent claim 17 because, like independent claim 12, it did require *distinct* portions. *Id.*, 2339-2342. However, the Board nonetheless entered a new anticipation rejection for independent claim 17 based on Seed for teaching one of the two methods encompassed by the claim when the resource is available at the first server. *Id.*, 2344-2345.

79. After the Board remanded the file, the Examiner proposed an Examiner's Amendment that: (1) if the resource is determined not to be popular (in limitation 1[d][ii] here), expressly requiring distribution of *distinct* portions of the content provider's library across different servers; and (2) incorporating dependent claim 18—which required determining which second-tier server to direct a request using a hash function—into independent claim 17 (which became limitation 15[d] here). *Id.*, 2363-2368. The claims were then allowed to issue. *Id.*, 2392.

VI. PRIOR ART

A. Seed (U.S. Pub. No. 2006/0015574)

80. Seed is directed to a network to deliver website objects using a hierarchy of edge, parent, and origin servers. Seed, [0020], Fig. 2.

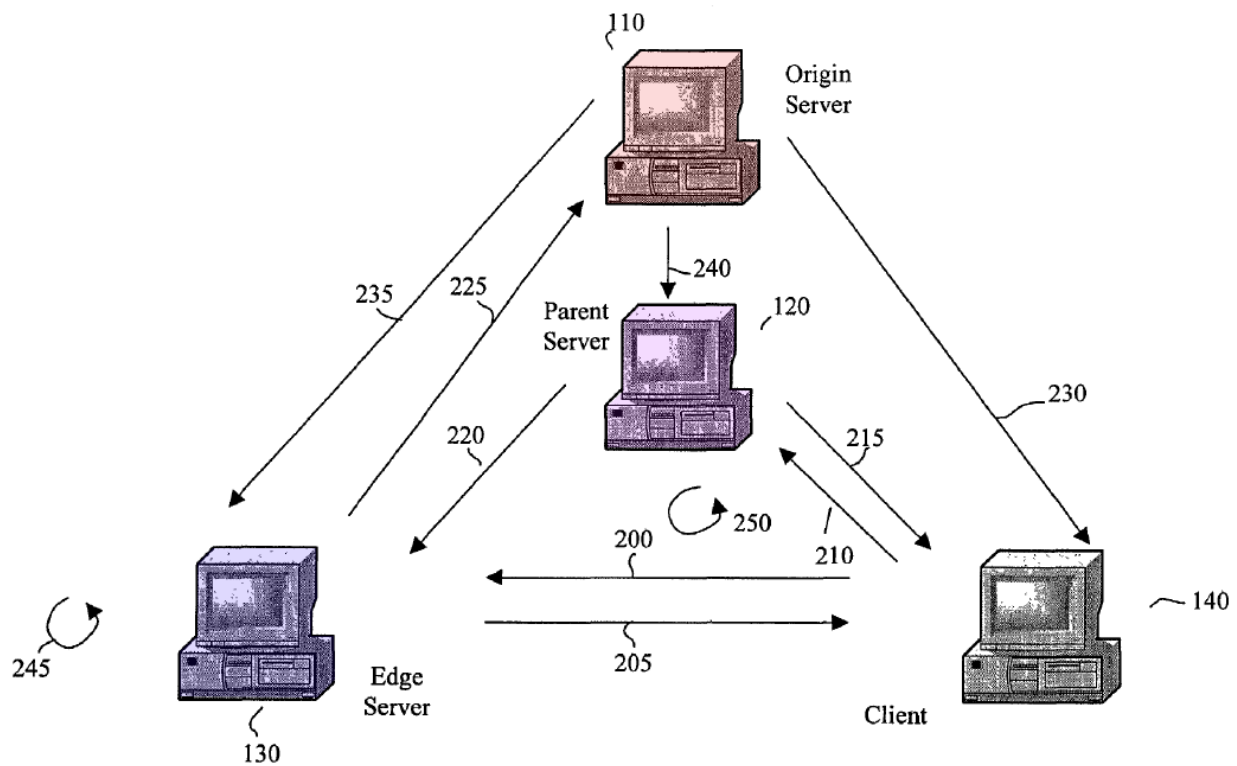


Fig. 2

81. Seed explains that edge servers are “populate[d]...with the most popular objects,” (Seed, [0016]), but “if a requested object is not available at the edge server a parent server that has the requested object will serve the requested object to the clients,” (*id.*, [0022]). To carry this out, Seed discloses that “[i]f the selected edge server does not have the requested object, a check is initiated...for the edge server to determine whether the requested object is popular and if so, to replicate the popular requested object to the edge server.” *Id.*, [0018], [0026]. Otherwise, the end user (client) is redirected to a parent server to obtain the object,

and the edge server does not replicate it (e.g., because the requested object is not popular). *Id.*, [0018], [0027].

82. Seed also contemplates that different parent servers will store different portions of a content provider's library, because it explains that parent servers may use a "recursive[]" redirection technique to search other parents for requested objects. Seed, [0029].

B. Swildens (U.S. Pub. No. 2004/0221019)

83. Swildens is directed to a network to deliver website files using a hierarchy of edge, origin, and staging servers. Swildens, [0067]-[0070], Fig. 2.

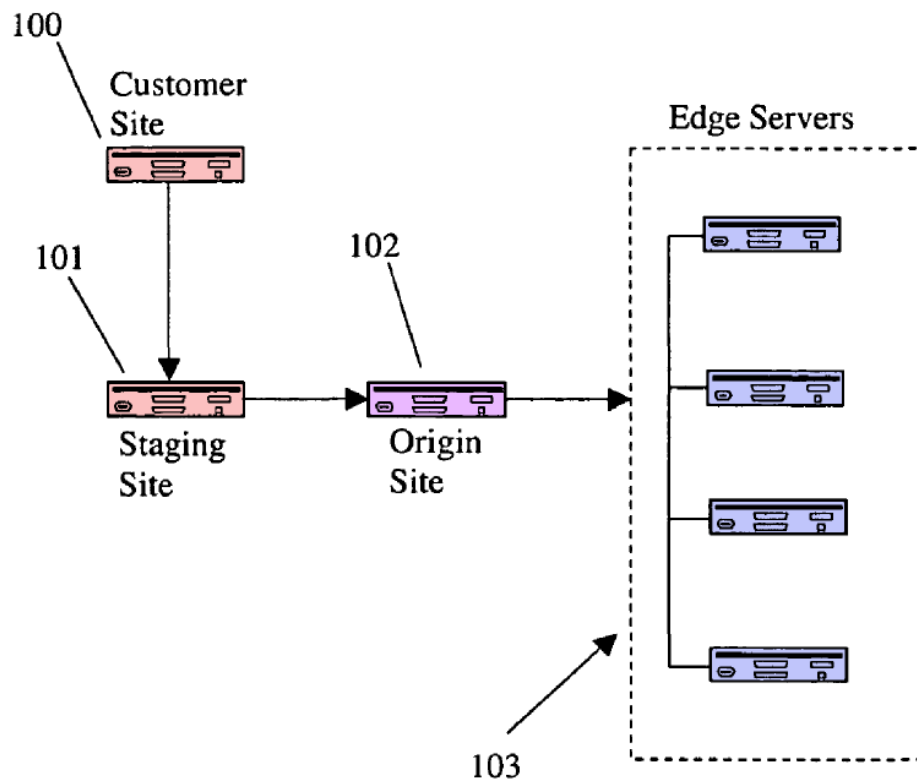


Fig. 1

84. Swildens discloses that “[a]n edge server maintains a dynamic number of popular files” and the origin server maintains “unpopular” files. *Id.*, [0055]. If a user requests a popular file, the edge server fulfills the request, but if the user requests an unpopular file, the user is redirected to the origin server to fulfill the request. *Id.*, [0068]-[0069]. Swildens, however, recognizes that “[a]n individual origin server does not have to be able [to] hold all of the content on its local storage device,” and describes using a “hash function” to partition distinct portions of the

unpopular files across multiple origin servers, and using the hash function to locate which origin server is storing the requested file. *Id.*, [0060]-[0066].

VII. LEVEL OF ORDINARY SKILL

85. I understand that when both interpreting the claims and assessing the prior art, I must do so from the perspective of a person of ordinary skill in the art (“POSITA”) as of the relevant priority date. My understanding is that the earliest possible priority date of the ’692 patent is April 4, 2008.

86. In my opinion, in the 2008 timeframe, a POSITA would have had at least a bachelor’s degree in computer science, electrical engineering, or a related field, and at least two years of work or research experience in the field of content-delivery management or networks. However, work experience can substitute for formal education and vice versa.³

³ I also understand that PO has proposed a definition of a POSITA in separate proceedings that, in addition to the experience I identify, educational experience may also include “computer engineering,” and work experience may include “2-3 years of work experience with content delivery, networking, or the equivalent.” Ex-1014, 6-7. I do not consider this definition to be materially different than my own, and I was at least a POSITA under either definition. Regardless, whichever definition is used, my opinions would remain the same.

87. I was at least a POSITA as of the earliest possible priority date of the '692 patent, and still am.

VIII. CLAIM CONSTRUCTION

88. As I noted above, I understand that unless claim terms are provided an express construction, the terms must be given their plain and ordinary meaning.

89. I understand that Petitioner is not advancing any constructions at this time, and I do not consider any claim constructions necessary for my purposes (to compare the plain and ordinary meaning of the claims to the prior art to evaluate whether the prior art would have rendered the claimed invention obvious). I have therefore applied the plain and ordinary meaning of each term in the claims throughout this declaration.

90. However, I also understand that during an appeal of the examination of the '692 patent, the Board expressly found that independent claims 1 and 15 recite two mutually exclusive methods, and only one of them needs to be performed. §V.B (Prosecution History). I do not take any position on the propriety of this finding, but I have assessed the prior art assuming all method steps must be performed. However, if only some method steps need to be performed, then my opinions and analysis only reinforce that the prior art taught the methods of those claims.

91. I also understand that in a suit between PO and a third party, the parties made several arguments related to claim terms in the '692 patent. First, PO argued

claim 1's conditionals mean that all steps need not be performed, which appears consistent with the Board's view, but the third-party argued that all steps must be performed, except they may be performed separately. Ex-1014, 12-14; Ex-1015 at 9-14. I do not have a position on this dispute, and I understand the Court has not resolved it yet, but regardless, the prior art taught both views, as I explain below.

92. Second, I understand the third-party argued that claiming servers with "distinct portions" of a content library (claims 1/15) is indefinite because the specification clarifies that "distinct portions" may "overlap only minimally," but the outer bound of "minimally" is unclear. Ex-1015, 7-9. PO argued the plain meaning permits "some overlap." Ex-1014, 9-11. I do not have a position on this dispute, and I understand the Court has not resolved it yet, but regardless, the prior art suggests no overlap between portions, which I understand there to be no dispute constitutes "distinct portions."

93. Finally, I understand that the third-party argued "origin server" in claims 1 and 15 refers to "a content provider's/customer's server," whereas PO advanced a plain-meaning construction. Ex-1015, 14. I do not have a position on this dispute, and I understand the Court has not resolved it yet, but regardless, the prior art taught the third party's specific construction as well.

IX. GROUNDS

A. Ground 1: Seed in view of Swildens (Claims 1, 2, 4-15, 17, 18)

1. Claim 1

- a. **1[pre]: A method of content delivery in a content delivery network comprising at least a first tier of servers, the method comprising:**

94. In my opinion, if the preamble is limiting, Seed discloses or suggests it.

95. Specifically, Seed discloses a method of content delivery in a content delivery network comprising a level of edge servers (*first tier*).⁴

96. Seed discloses “a method...for managed object replication and delivery over a network.” Seed, [0017]. Seed also describes the invention in the context of improving a “typical content delivery network (CDN)” (*id.*, [0010]), and therefore a POSITA would have understood the method is for content delivery in a content delivery network.

97. Seed also discloses the network has “parent server sites 120,” “edge server sites 130,” and “origin server sites 110,” which are arranged in levels. *Id.*, [0010], [0020]-[0022]; *see also id.*, [0018], [0025]-[0031] (describing relationships for content delivery at each level). Seed explains that each server level may contain

⁴ I use italics in parentheses to help map the prior art teachings to the claims.

“one” or a “cluster” of servers. Seed, [0020]. Seed depicts these server “levels” in Figure 2, and I have provided an annotated copy of that figure below:

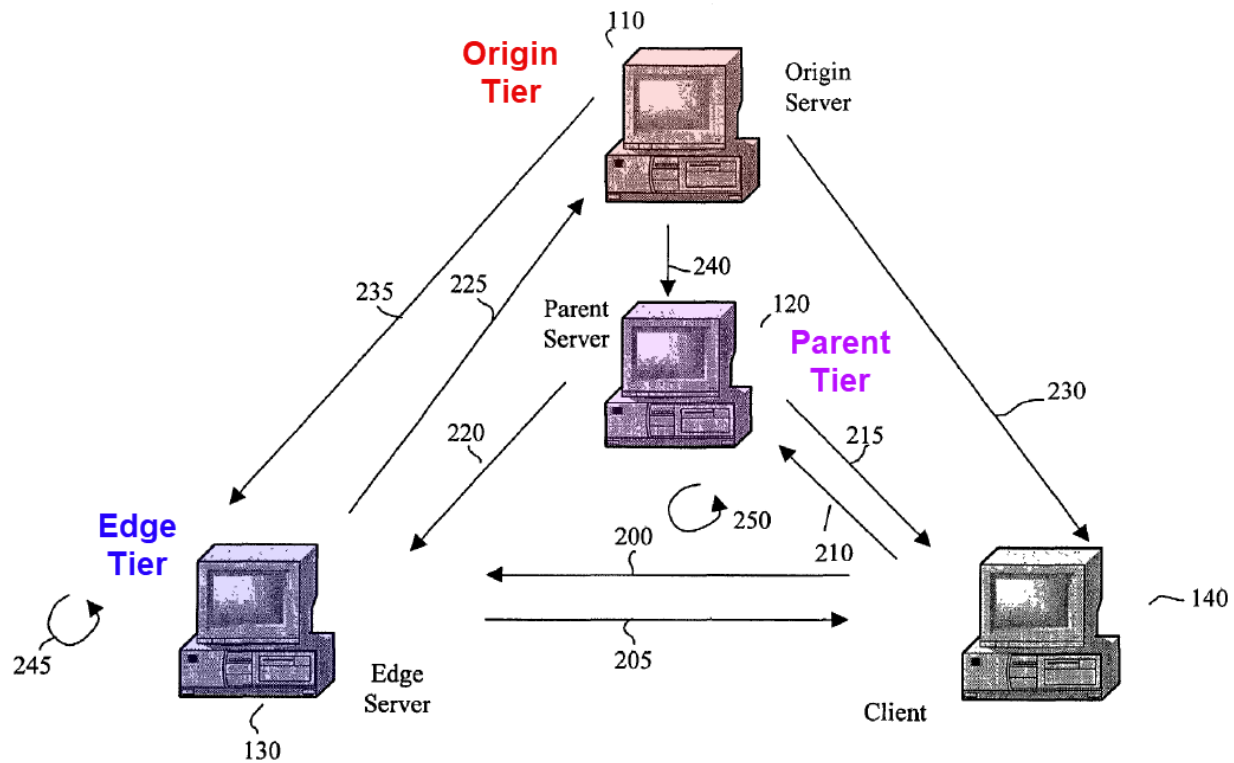


Fig. 2

98. In my opinion, a POSITA would have understood Seed’s parent, edge, and origin server levels to be “tiers” as that term is used in the ’692 patent. Specifically, the ’692 patent explains that “Tier 1 servers are also referred to as edge servers.... The Tier 2 servers...are also referred to as parent servers.” Ex-1001, 2:58-62. Moreover, Seed distinguishes between redirecting requests or transmitting content between “peers” (e.g., servers on the same level) or different levels (e.g., from edge to parent and/or origin) when a requested object is not available at the

server and/or level that is presently processing the request. *See, e.g.,* Seed, [0017], [0024]-[0029], [0062], Fig. 2.

- b. 1[a]: at a first server in the first tier of servers, obtaining a request from a client for a resource, wherein the resource is available as part of a content provider's library;**

99. In my opinion, Seed discloses or suggests this limitation.

100. Specifically, Seed discloses an edge server (*first server*) in a level of edge servers (*first tier*) obtains a request from a client for an object (*resource*) available as part of a content provider's collection of objects (*library*).

101. First, Seed discloses that an edge server (*first server*) obtains a request from a client for an object (*resource*). Seed discloses that “clients 140 access the network to request one or more objects” (Seed, [0020]), and “a request for a streaming object will be served by a handling edge server if that handling edge server has a copy of that object,” (*id.*, [0018]; *see also id.*, [0024] (“the method of managed object replication and delivery directs (at 200, 300) a client, requesting one or more objects, to an edge server in the network”), Fig. 3(a)).

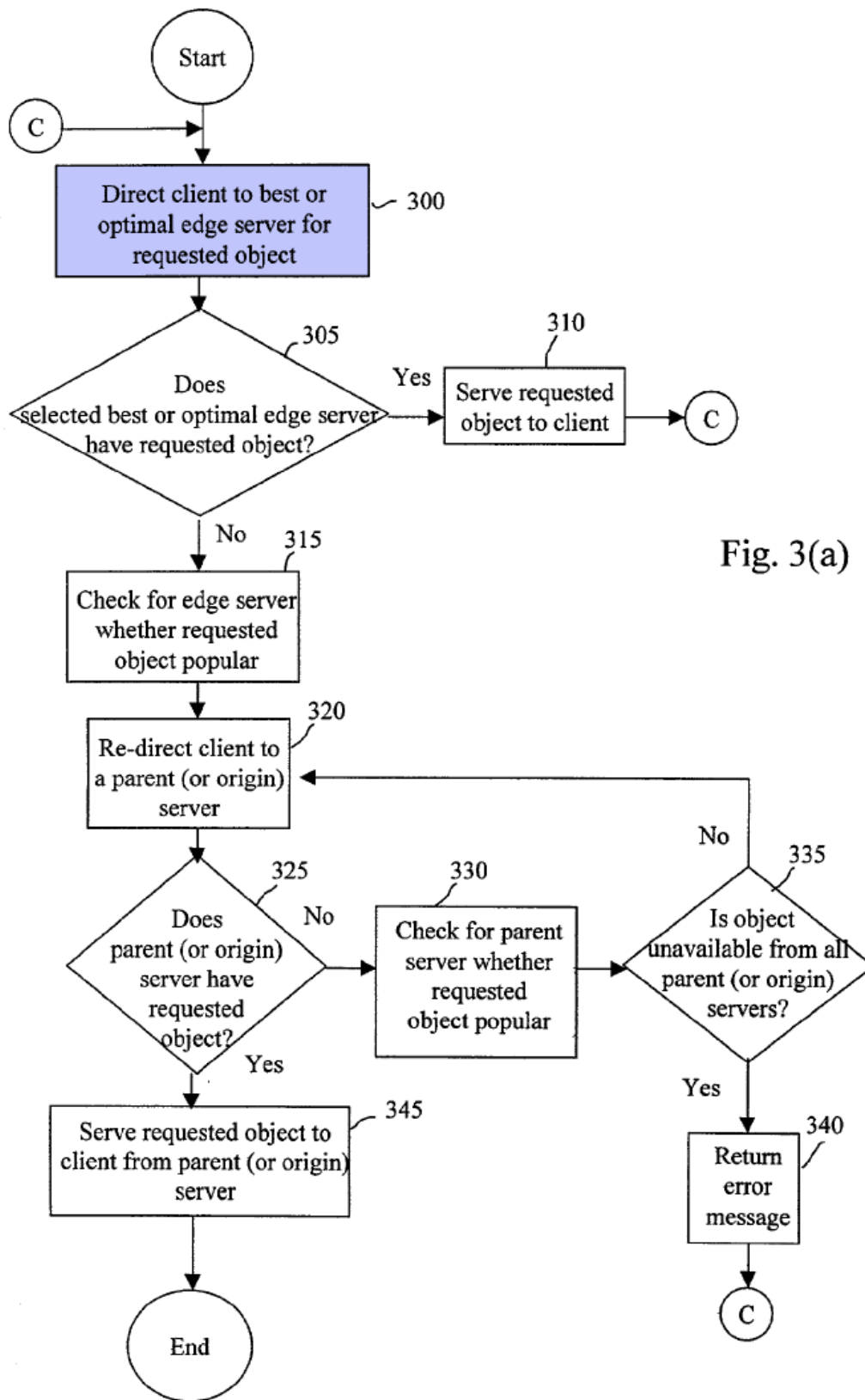


Fig. 3(a)

102. Second, Seed also describes what, in my opinion, would have been understood as objects (*resources*) being available as part of a content provider's library. Seed discloses that "[c]ontent provider companies...that subscribe to the CDN service" maintain an origin server "for storing and serving one or more objects" that are then distributed and served through the CDN. *Id.*, [0010], [0020]. Although Seed does not expressly mention a "library" as claimed, in my opinion, a POSITA would have understood that the content provider's collection of objects downloaded to the CDN servers would constitute such a "library," as content provider collections were often referred to using this term. *See, e.g.*, Ex-1009, 19 (when describing one exemplary CDN, noting it "stor[es] customer's content library"), 37 ("The server where the definitive version of the content resides is called [the] origin server. It is updated by the content provider."); Ex-1016, 2:20-21 ("The digital content items are stored in a content library...in a server"), 25:27-26:3, Fig. 2 (depicting a "Content Library" connected to a "CDN").

- c. **1[b]: if the resource is not available at the first server or at a peer of the first server, determining if the resource is popular;**

103. In my opinion, Seed discloses or suggests this limitation.

104. Specifically, Seed discloses that “[i]f the selected edge server does not have the requested object, a check is initiated (at 315) for the edge server to determine whether the requested object is popular.” Seed, [0026], Fig. 3(a).

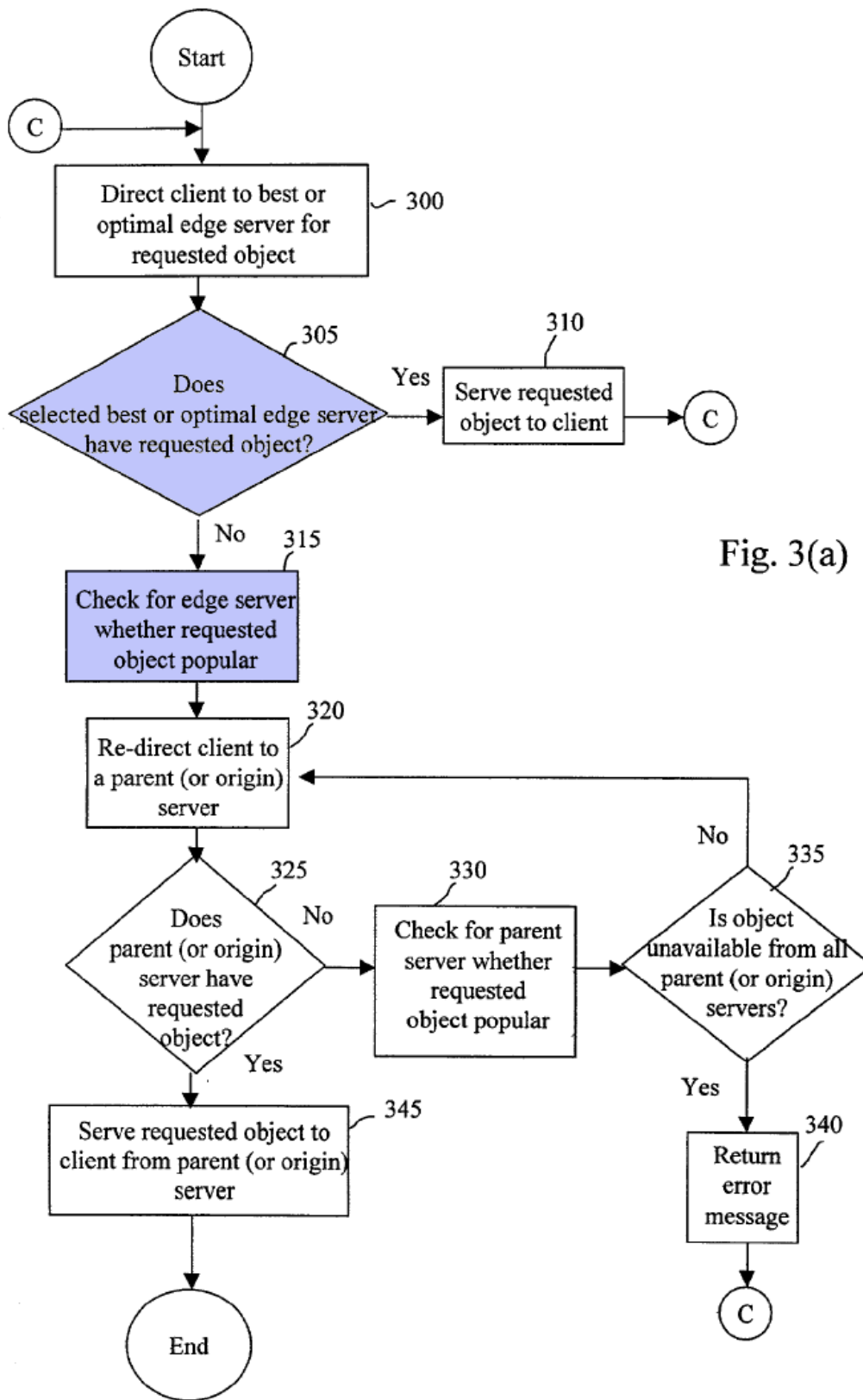


Fig. 3(a)

- d. **1[c]: if the resource is determined to be popular, then the first server obtaining the resource and the first server serving the resource to the client, otherwise,**

105. If both 1[c] and 1[d] must be performed (§VIII (Claim Construction)), in my opinion, Seed discloses or suggests this limitation.

106. Specifically, Seed discloses or renders obvious that if the object (*resource*) is popular, the edge server (*first server*) replicates (*obtains*) the object and then serves the object to the client, either in response to a subsequent request or by fetching and then serving the content from another server.

107. First, Seed discloses that if “the requested object is popular...to replicate the popular requested object to the edge server” (*if the resource is determined to be popular, then the first server obtaining the resource*). Seed, [0026].

108. Although Seed discloses replicating (*obtaining*) the object, Seed discloses a slightly different order of operations for replicating *and then serving* the content. When a popular object is missing from an edge server, Seed does not expressly describe ***first replicating*** the popular file to the edge server and ***then serving*** it to the client. Instead, Seed discloses ***first redirecting*** the client to another server that has the missing, popular object—ensuring the fastest delivery to the end user—and ***then replicating*** the object to the edge server. Seed, [0017]-[0018].

109. However, in my opinion, a POSITA would have understood that clients routinely request delivery of the same content (or what Seed calls “objects”) multiple

times, especially for objects that are deemed popular. For instance, each time a client accesses the same website or streams the same media, typically a new request is sent from the client to a server hosting the requested content. For this reason, although Seed discloses redirecting the client to another server when the edge server determines a requested popular object is missing and subsequently replicates the popular object in its cache, a POSITA would have understood that any subsequent request from the same or a different client for the same object would be served by the edge server that would now be hosting it after replicating it in its cache (*the first server serving the resource to the client*).

110. Although I do not understand the claim this way, if the claim requires that the edge server serve the object to the client after only *the first request* (and not subsequent requests for the same object), in my opinion, this would have also been obvious to a POSITA.

111. First, Seed discloses that “the checking of whether the requested object is popular and replicating the popular requested object to the edge server may be performed independently of one or more functions of the method of managed object replication and delivery.” Seed, [0027]. In my opinion, Seed recognizes that there is flexibility to the order of operations Seed discloses, and thus Seed’s disclosure is not limited to redirecting the first request for a missing, but popular object, and then separately replicating the popular object later.

112. Second, Seed recognizes that conventional CDN networks routinely configure edge servers to “fetch objects” from other servers “when a requested object is not available on a handling edge server.” Seed, [0013]; *see also, e.g.*, Ex-1009, 25 (“One can view the replica server as a caching proxy which fetches the content from the origin server on a local cache miss.”). Seed, however, prefers redirecting the first missed request for a popular object to another server, rather than fetching and serving the popular object itself to avoid “having [the object] go through the parent path” that “can be slow.” Seed, [0013]-[0014]. But even so, Seed still describes its network as being configured to fetch objects. *See, e.g.*, Seed, [0046] (noting “the cost of fetching” is considered as part of the replication policies of the network), [0059] (suggesting partitioning objects into smaller chunks so that “a first part of an object can be served from edge servers quickly, even if most of the object stream ***must be fetched*** from a parent server or origin server.” (emphasis added)).

113. In my opinion, because Seed discloses that its order of operations is flexible, and its servers are configured to fetch objects, a POSITA would have understood that Seed at the very least suggests that the edge servers can be configured to fetch popular objects and serve those objects directly to the client in response to a request (*the first server serving the resource to the client*).

114. But even if it is determined that Seed cannot be read this way, in my opinion, a POSITA would have been motivated to modify Seed to fetch-and-serve

in response to a request for a popular object that is not already being stored in the cache on the handling edge server. When a server receiving a request for an object did not have the requested object, there were only a handful of ways to fulfill the request. The most common ways for a server to facilitate delivery of the requested object included:

(1) Obtaining the object for itself and then serving it to the client (e.g., fetch-and-serve or stream-through). This is like when a store places an order for an out-of-stock item on behalf of the customer, and the store handles delivery once the product is received by the store.

(2) Sending or redirecting *the client* to another server that has the object. This is like a store informing a customer that another store has the requested item in stock, and the customer must go get it from the other store.

(3) Forwarding or redirecting *the request* to another server that has the object. This is like a store forwarding a customer's order for an out-of-stock item to another store that has the item in stock, and the other store handles delivery.

115. In my opinion, because there were only a finite number of ways (at a high level) for a server to facilitate the delivery of a requested but missing object, in my opinion, a POSITA would have been motivated to try any of those well-known ways to determine which worked best for the particular network and use case.

Choosing one option over another would have also been nothing but a simple substitution of one known element (e.g., fetch-and-serve) for another (e.g., redirecting-and-fetching) to achieve a predictable result (e.g., delivery of the requested object), or applying a known, suitable alternative technique (e.g., fetch-and-serve) to the one Seed discloses (e.g., redirecting-and-fetching) to solve a known problem in the art (e.g., serving content from the server handling a request for a missing object). In fact, those in the art routinely recognized that fetching, redirecting, and forwarding were known alternatives. *See, e.g.*, Ex-1017, 2:38-42 (“the server uses the virtual URL system to locate the best current location for the resource and either redirects the requesting browser to it...or else fetches and returns the resource”); Ex-1018, [0040] (“the secure edge server 508 can forward requests to the origin, fetch content from the origin to deliver to the end user, and/or deliver content from its own cache as required.”).

116. Regardless, a POSITA would have understood that each technique has its own tradeoffs, and although Seed expresses a preference for redirecting due to its speed, fetching would provide different benefits. For instance, the fetch-and-serve method may result in a slower response to the client, but it may also be cheaper than redirecting. As one example, in typical CDNs, “[t]he customer is charged for the bandwidth used to transfer files from the origin servers to the edge servers.” Swildens, [0043]. When using redirects to fulfill requests for popular objects, if a

handling edge server must redirect the request to, e.g., an origin server to fulfill because it is missing from the edge, the origin server needs to transfer the requested object to the client, leading to one cost associated with that transfer. Then, the handling edge server that originally received the request needs to replicate the popular object from the origin server, leading to a second cost for that transfer. In a fetch-and-serve scheme, however, only one request is made of the origin server by the handling edge server, and that same transfer from the origin is used to both replicate the object to the edge *and* serve the requested content from the edge.

117. In my opinion, a POSITA would have also had a reasonable expectation of success in making this modification because both fetching and redirecting were two of the most common ways to serve missing objects to clients, and Seed already envisions configuring the server having the capability to do both. Seed, [0046], [0059]; Ex-1017, 2:38-42. The only modification required would be to change some computer code to instruct edge servers how to handle a missed request for popular object: rather than direct the client to the location the server has identified so that the client can obtain the object for itself, to go fetch the object itself from the location it has identified (which the server is already programmed to do to obtain a copy for itself *after* redirecting the client) and then provide a copy to the client. In my opinion, the required changes to code and any further implementation details would have been well within the skills of a POSITA.

- e. **1[d][i]: if the resource is determined not to be popular, directing the client to a second server in a second tier of servers distinct from the first tier of servers,**

118. If both 1[c] and 1[d] must be performed (§VIII (Claim Construction)), in my opinion, Seed discloses or suggests this limitation.

119. Specifically, Seed discloses that if the resource is determined not to be popular, directing the client to a parent server (*second server*) among parent-level servers (*second tier*) distinct from the edge-level servers (*first tier*).

120. Seed discloses that edge servers are populated “with the most popular objects” and “the rest” are served from “parent servers.” Seed, [0016]. Seed also discloses that an edge server would not have the requested object for two reasons: (1) because the object is not sufficiently popular (and therefore does not need to be stored at the edge); or (2) the edge server has not yet made the necessary determination that the object is popular and replicated it (e.g., an unpopular object only recently became popular). Seed, [0016]-[0017], [0026], [0032]-[0034]. But when the object is determined not to be popular, Seed discloses that “the selected edge server directs (at 210, 320) the requesting client 140 to a parent server 120” among parent-level servers (§IX.A.1.b (1[pre])) without an additional instruction to replicate the object. Seed, [0028], Fig. 3(a).

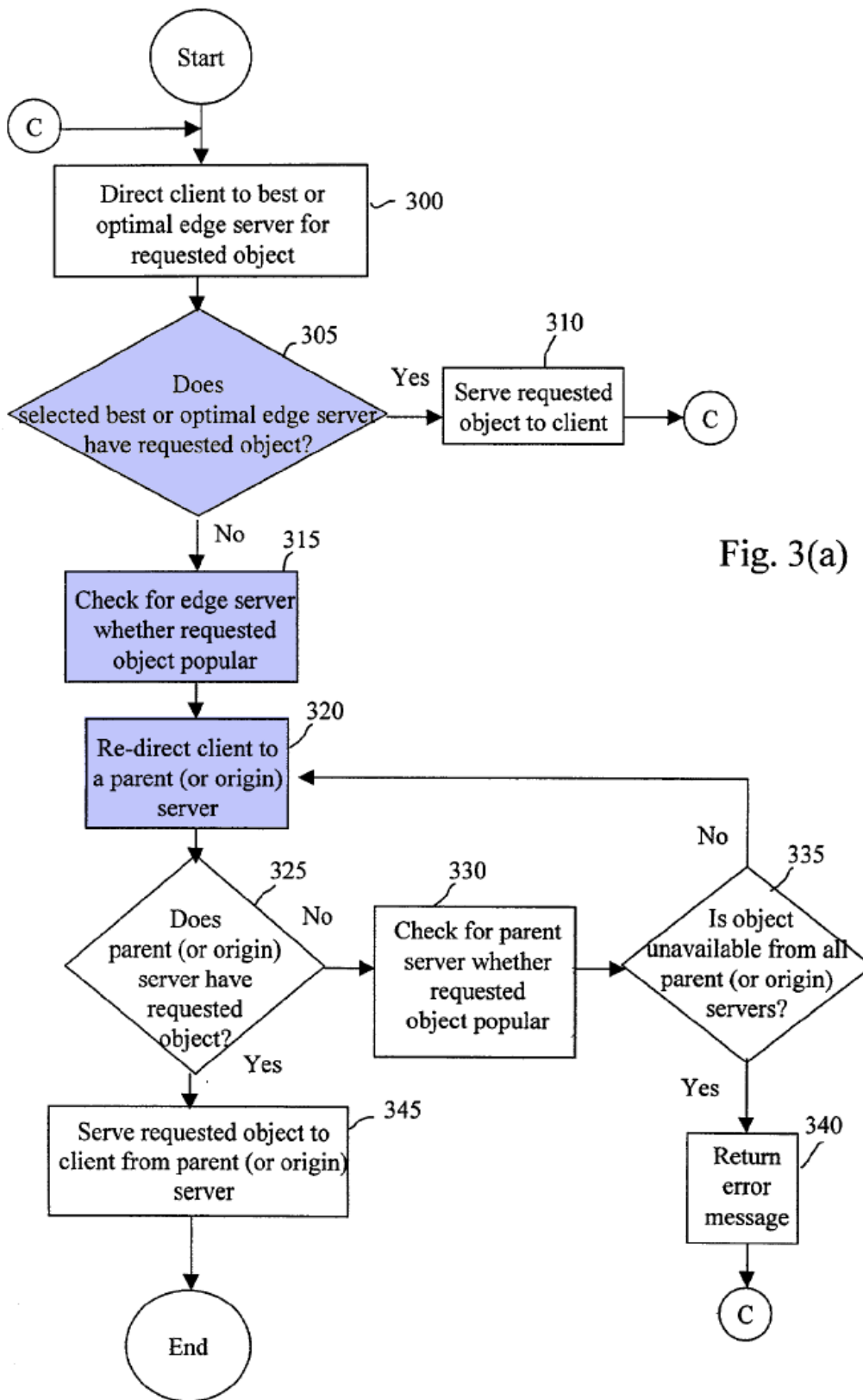


Fig. 3(a)

- f. **1[d][ii]: wherein the second server comprises a first portion of the content provider's library, the first portion comprising at least the resource, wherein at least one other server in the second tier comprises a second portion of the content provider's library, wherein the first portion of the content provider's library is distinct from the second portion, and**

121. If both 1[c] and 1[d] must be performed (§VIII (Claim Construction)), in my opinion, Seed alone or combined with Swildens teaches or suggests this limitation.

122. Specifically, Seed, alone or combined with Swildens, renders obvious a parent server (*second server*) that stores one (*first*) portion of the content provider's library that has the requested object (*resource*), and another parent server (*other server*) in the parent-level servers (*second tier*) stores a distinct (*second*) portion of the content provider's library.

123. Seed discloses that when the edge server redirects the client to a parent server, (§IX.A.1.e (1[d][i])), “if a parent server does not have the requested object, the parent server could itself use a redirection technique recursively (at 325, 335, 320) until a final parent server is reached that has the requested object.” Seed, [0029]. Seed also notes that *parent* servers also evaluate whether a missing, requested object is popular to determine whether they also need to replicate the object. Seed, [0028].

124. In my opinion, a POSITA would have understood these passages as a recognition that different parent-level servers have different sets of objects from the

content-provider's collection (*library*); some parent servers (*second server*) may have portions with the requested object (*a first portion of the content provider's library, the first portion comprising at least the resource*), and other parent servers (*one other server in the second tier*) have different portions that do not include the requested object (*wherein the first portion of the content provider's library is distinct from the second portion*). *Id.* Otherwise, if every parent server had the same set of objects, there would have been no need for Seed to have each parent server perform their own popularity determination to replicate objects, or perform recursive redirection to other parent servers in search of missing objects. However, Seed does not expressly specify the extent of any overlap between parent servers.

125. Even if Seed alone does not disclose this limitation, though, in my opinion, a POSITA would have been motivated to modify Seed to expressly partition distinct portions of a content provider's library across different parent servers based on Swildens. Swildens describes another CDN in which "edge server[s] maintain[] a dynamic number of popular files," while an intermediate set of "origin" servers store unpopular files. Swildens, [0043], [0055].

126. First, it is worth clarifying that Seed and Swildens use different terms to label their servers. Seed uses the terms "edge," "parent," and "origin" servers, but Swildens uses the terms "edge," "origin," "staging," and "customer" servers. *See, e.g.,* Seed, Fig. 2; Swildens, Fig. 1.

127. Each of these terms has a generally understood meaning that insinuates slightly different configurations. For instance, an “origin” server is considered a single source of truth for all objects in a content provider’s library. *See, e.g.*, Ex-1009, 5, 37 (“The server where the definitive version of the content resides is called the origin server. It is updated by the content provider.”).

128. However, as the ’692 patent recognizes, CDNs may also provide their own “surrogate” origin servers—e.g., a complete or partial clone of the content provider’s origin server—that are used to effectively insulate the content provider’s separate origin server from traffic. Ex-1019, 1:19-46 (“The content delivery infrastructure usually comprises a set of ‘surrogate’ origin servers that are located at strategic locations...for delivering content to requesting end users” distinct from “a content provider’s origin server”). In other words, “surrogate” origin servers are a set of intermediate servers provided by the CDN to replicate and stand-in for the content provider’s origin server.

129. Parent servers, on the other hand, are another set of intermediate servers generally configured more like edge servers. They cache at least some of a content provider’s library closer to the end user, but are typically not the first line of servers to handle incoming client requests like edge servers are. Instead, they are backups to a cluster of edge-level servers (i.e., they are parent-edge servers to support child-edge servers). For instance, if an edge server does not have the requested object, the

first place the edge server may look is its associated parent server that may store more of the content provider’s library, but is responsible for supporting many different edge servers.

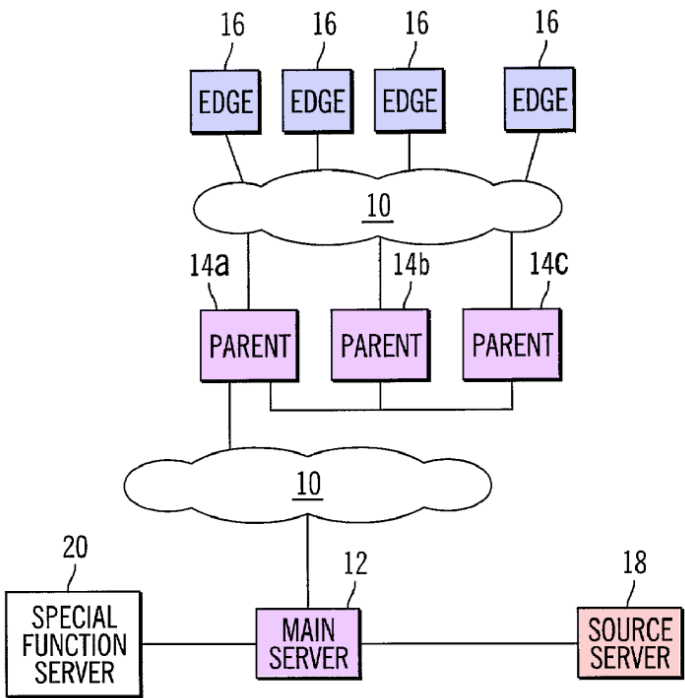


FIG. 2

Ex-1012, Fig. 2 (annotated).

130. To keep track of how the '692 patent and prior-art references reference these servers using a common set of terms, I have provided the table below:

Term	'692 patent	Seed	Swildens
<i>Customer-Origin</i>	Origin	Origin	Customer Server/ Staging Server
<i>(Intermediate) CDN-Origin</i>	CDN Origin	-	Origin
<i>(Intermediate) Parent</i>	Parent	Parent	-
<i>Edge</i>	Edge	Edge	Edge

131. With this understanding, Swildens recognizes that although intermediate (CDN-origin) “servers together store the complete copies of all the content for a particular customer,” (*id.*, [0050]), “[a]n individual [intermediate] server does not have to be able [to] hold all of the content,” (*id.*, [0060]). Instead, Swildens uses a “hash function”—which is an algorithm to “map[] the large number of content identifiers across the much smaller number of storage servers,” (*id.*, [0061])—to distribute distinct portions of a content provider’s library across different intermediate (CDN-origin) servers and identify on which server any specific content resides, (*id.*, [0060]).

132. A hash function works by applying a set of rules to sort one set of data often called “keys” (k) into another smaller set of data often called “buckets” or “slots” ($h(k)$). For instance, one of the simplest hash functions is a modulo, which uses basic division to sort any integer into a preset number of buckets by having the

remainder dictate which bucket the key is placed. To provide an example, if a hash function provides 10 buckets, the modulo would just be the unit-digit of the key. The key “51” would just be placed in bucket “1” because “51” can be divided by 10 five times, with “1” remaining. Similarly, the key “37” would be placed in bucket “7” because “37” can be divided by 10 three times, with “7” remaining.

133. However, hash functions can be far more complex and can serve different purposes. For instance, hashes can also be used for security. One common security-minded hash function is known as MD5, which is used to verify the authenticity of a file. An MD5 hash is a 128-bit value typically written as a 32-character hexadecimal, meaning it has 2^{128} buckets (or 34×10^{37} possible options). When a file is run through the MD5 hash function, it provides an output that should in theory be unique to only that file. Changing a single letter in a Word document and running them through an MD5 hash function would produce a completely different hashed output. To verify two documents are identical, then, their MD5 hashes can be compared to make sure no modifications have been made. *See* Seed, [0057] (describing MD5 hash to verify object replication). While MD5 is useful for security purposes, an MD5 hash function would not be good for distributing files to a limited number of servers, because it has more possible buckets than possible servers that could ever exist.

134. Swildens uses a hash function to distinctly distribute files across different servers (more like the modulo example I provide above) using, for instance, their content identifiers as the key. Swildens, [0060]-[0065]. And although Swildens recommends some level of redundancy between intermediate origin servers to improve performance (Swildens, [0058]), unless the hash function is specifically configured to allow overlap for redundancy, by running content identifiers through a hash function, a POSITA would have understood that would typically result in each server having wholly distinct, non-overlapping portions of the library. §VIII (Claim Construction). That is because running a content identifier (key) through a hash function would typically produce only one output, which a POSITA would have understood would be the server to which it is assigned (e.g., a one-to-one key-to-slot mapping).

135. In my opinion, a POSITA would have been motivated to distribute distinct portions of a content provider's library across Seed's intermediate parent-level servers using a hash function, consistent with Swildens. Swildens recognizes the tradeoffs involved: that "[s]ervers should typically have at least the combined storage and delivery capacity to store one copy of all of the content files and satisfy peak delivery load," and adding more intermediate servers "increase[s] the minimum amount of redundancy required and...improve[s] the performance of unpopular or rapidly popular content." Swildens, [0058]. However, "[t]he customer is charged for

the amount of storage required on the [intermediate] site 102 to store the customer's files" as well as "the bandwidth used to load the files onto the [intermediate] site 102 and edge servers 103." *Id.*, [0056]. In other words, it is another recognized tradeoff between speed, reliability, and cost. Storing copies of all content in a library on each intermediate server may be faster and more reliable because any intermediate server can fulfill any request directly, but requires more server memory (and thus higher costs). But storing distinct portions of a content library on each intermediate server—while ensuring each piece of content is stored on at least one server—may take longer to fulfill requests, but requires much less memory of each server. Thus, storing distinct portions of the content provider's library on each of Seed's parent servers would save memory costs while ensuring enough redundant copies of all (not popular) content exist in the network to fulfill any client requests.

136. Additionally, a POSITA would have appreciated that Swildens' hash function would be faster and more efficient in locating unpopular content distributed across Seed's parent servers than Seed's recursive redirection, also leading to faster fulfillment times. With Seed's recursive redirection, a parent server has no way to gauge whether another parent server is storing the requested object; it simply passes on the request until the request reaches a parent server that does. Seed, [0029]. For example, if five parent servers were used in Seed's system, the redirection would need to occur five times when the content is located in the fifth server. Swildens'

hash functions, however, identify the exact server that “should” be storing the requested file, and are incorporated into the uniform-resource locator (“URL”) to redirect the client to that exact server in the first instance. Swildens, [0137].

137. In my opinion, a POSITA would have had a reasonable expectation of success in combining Seed and Swildens because Seed already envisions distributing distinct portions of a content library across an intermediate (parent) tier of servers, for the reasons I described above. The only difference is that Swildens purposefully distributes distinct portions, and provides an efficient, easily implementable way to both distribute those distinct portions and then locate the requested content using hash functions. Combining Seed with Swildens therefore requires modifying how Seed assigns URLs to both distribute and then later locate content in the parent tier of servers by incorporating a hash into those URLs. In my opinion, implementing a hash function for this purpose would have been a straightforward endeavor for a POSITA, given its widespread use for this purpose. *See, e.g.*, Ex-1009, 58 (noting distributed hash tables are one of four common techniques to store and locate files in a CDN).

138. Although Swildens distributes distinct portions of unpopular content across its CDN-origin level servers rather than parent-level servers, a POSITA would have appreciated that Swildens teachings would apply to Seed’s parent servers despite the slightly different configurations their label suggests. Neither Seed

nor Swildens identify any unique parent- or origin-specific configurations, but rather instead both describe these servers as an intermediate set of servers between the edge and customer-origin servers to store and serve unpopular content not stored by the edge. In other words, a POSITA would have considered it largely irrelevant that Seed and Swildens called their servers different names.

g. 1[d][iii]: wherein the second tier is any intermediate tier between the first tier and an origin server that stores resources associated with the content provider's library, and

139. If both 1[c] and 1[d] must be performed (§VIII (Claim Construction)), in my opinion, Seed discloses or suggests this limitation.

140. Specifically, Seed discloses that parent-level servers (*second tier*) are intermediate to the edge-level servers (*first tier*) and origin server (e.g., customer's server). §IX.A.1.a (1[pre]).

141. Seed also discloses that “[c]ontent provider companies...that subscribe to the CDN service” maintain an origin server “for storing and serving one or more objects.” Seed, [0010], [0020]. In my opinion, given the well-known nature of origin servers, a POSITA would have understood that Seed's origin server stores objects associated with the content provider's collection of objects (*an origin server that stores resources associated with the content provider's library*).

h. 1[d][iv]: the second server serving the resource to the client.

142. If both 1[c] and 1[d] must be performed (§VIII (Claim Construction)), in my opinion, Seed discloses or suggests this limitation.

143. Seed discloses the parent server (*second server*) serves an unpopular object (*resource*) to the client. Seed, [0028] (“the client 140 is redirected to a parent server that has the requested object and is able to serve...the requested object to the client”), [0029], Fig. 3(a).

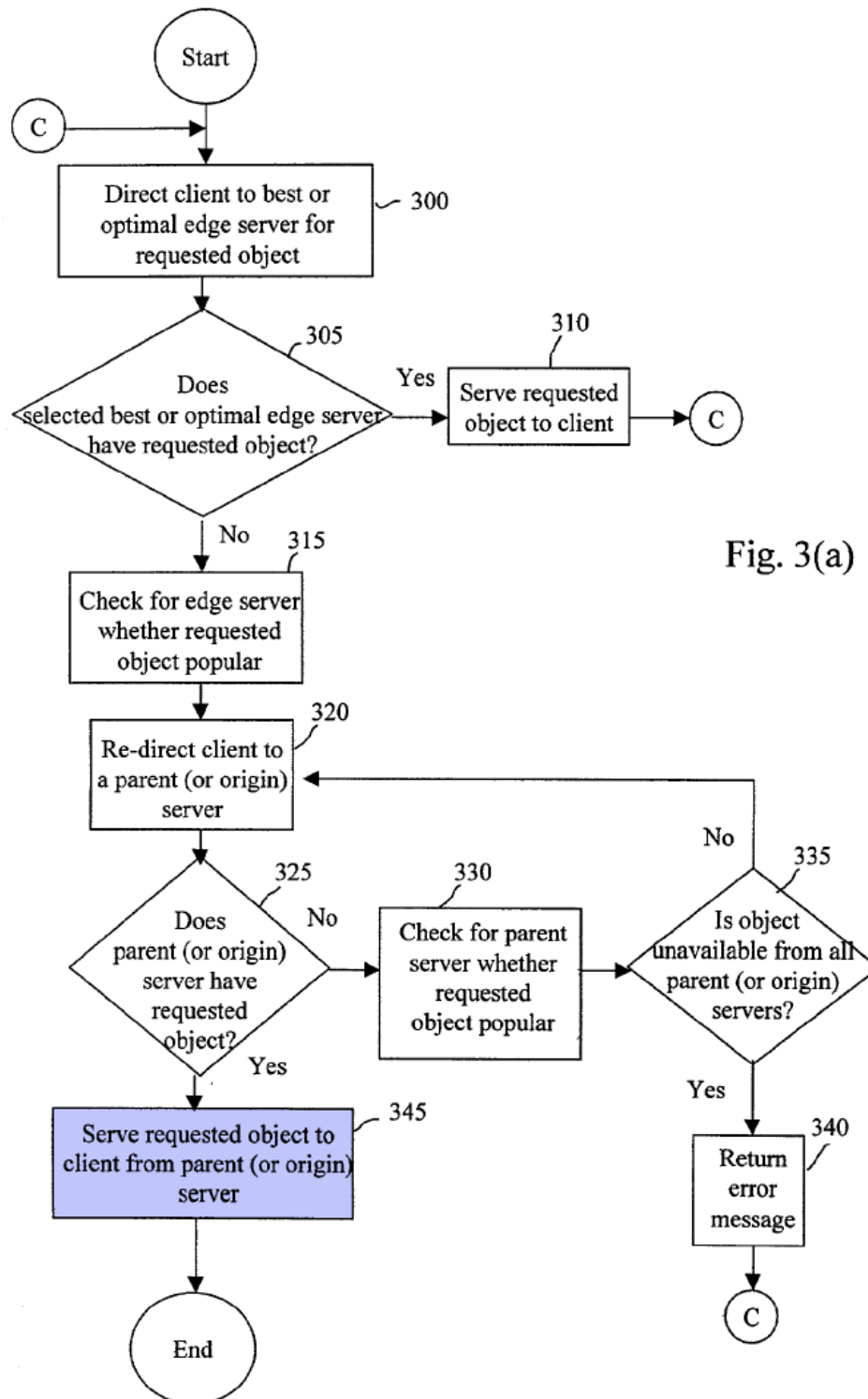


Fig. 3(a)

2. Claim 2: A method as recited in claim 1, wherein portions of the content provider's library are logically partitioned across servers in the second tier.

144. If both 1[c] and 1[d] must be performed (§VIII (Claim Construction)), in my opinion, Seed combined with Swildens teaches or suggests this additional limitation.

145. Seed contemplates storing different portions of a content provider's collection (and specifically unpopular objects) across parent-level servers (*second tier*) (Seed, [0029]; §IX.A.1.f (1[d][ii])), but Seed does not expressly describe how the collection is partitioned across parent servers.

146. Swildens, however, teaches partitioning a content provider's collection across intermediate (CDN-origin) servers (*second tier*) using a hash function, and specifically “comput[ing] a hash...from a content file identifier” and storing the file at a specific intermediate-level (CDN-origin) server based on the hash. Swildens, [0051]-[0052]. Swildens also teaches that “[a] good hash function is one that spreads the content storage requirements of the content identifiers across the actual storage capacities of each of [intermediate] server group.” *Id.*, [0061].

147. In my opinion, a POSITA would have understood that the hash function involves a logic-based, mathematical algorithm to convert, e.g., content identifiers (keys) into hashes for sorting into the available servers (buckets). In other words, Swildens teaches partitioning a content provider's library across multiple designated

servers where specific and distinct portions of the content are distributed to each of the intermediate-level servers (like Seed's parent level; *second tier*) using a hashing algorithm (*logic*) so that the system can calculate which server contains a specific portion of the content and fetch it.

148. For the reasons I explained previously, in my opinion, it would have been obvious to implement Swildens' hash function and partitioned libraries in Seed's CDN. §IX.A.1.f (1[d][ii]).

3. Claim 4: A method as recited in claim 1, wherein the content provider's library comprises a plurality of content-types, each resource in the library being associated with one of the plurality of content-types.

149. In my opinion, Seed discloses or suggests this additional limitation.

150. Seed discloses that “an object includes, without limitation, an audio file (such as, e.g., an MP3 (Motion Picture Experts Group-1 Layer 3) file and a RealNetworks, Inc. Real format file), a video file (such as an MPEG file), an image file (such as, e.g., a BMP (bitmap) file or JPEG (Joint Photographic Experts) file) and any other software or data file or object.” Seed, [0010]. This is consistent with how the '692 patent describes “content.” *See, e.g.*, Ex-1001, 12:29-31 (“Assume, for example, that a content provider's library comprises multiple content-types (e.g., movies, video games, music, software patches, etc.).”).

151. It is also my opinion, a POSITA would have understood that each of the above-identified file types are “content types” as claimed (e.g., .mp3, .mp4, .mpeg, .bmp, .jpeg), and that each object (*resource*) would be associated with one of the content types.

4. **Claim 5: A method as recited in claim 4 further comprising storing distinct portions of the content provider's library on two or more servers in the second tier, wherein the content-types on each of the two or more servers are proportionally distributed, the content-types comprising at least one of audio content, video content, gaming content, and software content.**

152. If both 1[c] and 1[d] must be performed (§VIII (Claim Construction)), in my opinion, Seed combined with Swildens teaches or suggests this additional limitation.

153. As I explained previously, Seed, alone or combined with Swildens, teaches storing distinct portions of a content provider's library across multiple parent-level servers (*second tier*). §§IX.A.1.e-IX.A.1.f (1[d][i]-[ii]).

154. As I also explained previously, Seed also discloses the library may include multiple content types, including audio content, video content, and software content. §IX.A.3 (cl. 4).

155. When combining Seed and Swildens to partition content libraries into distinct portions stored across multiple parent-level servers, in my opinion, a POSITA would have appreciated that the content types of objects would be

distributed on each of Seed's parent-level servers by some proportion (e.g., some parent servers would have some proportion of objects of a certain content type). For instance, some parent server content assignments may be hashed to store exclusively movies, and other parent servers may have their content assignments to be hashed to store exclusively music. Alternatively, some parent servers may distribute different movies across different parent servers, because they were known to be much larger files, and thus more resource-intensive to serve to clients. Thus, providing, e.g., size- or resource-based distributions would provide an easy way to balance server load. *See, e.g.*, Ex-1020, [0038] ("For example, a replica of a movie content having title XXX is stored on node N5; a replica of a movie content having title YYY is stored on node N3. In this way, contents are distributed and stored among multiple nodes Nn"); Ex-1021, 8:25-56 (describing heuristic distribution across servers).

5. **Claim 6: A method as recited in claim 1, wherein resources of the content provider's library are distributed in a substantially equal manner across servers in the second tier based on hash values of uniform resource locators (URLs) associated with two or more respective resources in the content provider's library.**

156. If both 1[c] and 1[d] must be performed (§VIII (Claim Construction)), in my opinion, Seed combined with Swildens teaches or suggests this additional limitation.

157. As I explained previously, Seed combined with Swildens teaches storing distinct portions of a content provider's library across multiple parent-level servers (*second tier*) using a hash function. §§IX.A.1.e-IX.A.1.f (1[d][i]-[ii]).

158. As part of this distribution, Swildens explains that “[a] good hash function is one that spreads the content storage requirements of the content identifiers across the actual storage capacities of each [intermediate] server group.” Swildens, [0061]. In other words, Swildens teaches distributing content libraries in a way that accounts for the storage capacity of each server. When doing so, in my opinion, a POSITA would have appreciated that when implementing Seed and Swildens' combined teachings in a network of parent servers that all have the same amount of storage, Swildens' hash function would produce a substantially equal distribution of content across the parent-level servers (*servers in the second tier*).

159. Swildens also explains that hash values are determined based on “content identifiers” or “published URLs” associated with the content (Swildens, [0061]-[0062], [0065]-[0066]), and “[r]edirected requests to [intermediate] servers have a hash included in the URL” that is used to locate which intermediate server is storing the requested object, (Swildens, [0060]). When combining Seed to use Swildens' hash function, therefore, the combination would distribute content across the parent-level servers (*servers in the second tier*) based on hash values either included in URLs, or of URLs themselves (*based on hash values of uniform*

resource locators (URLs)) associated with content via content identifiers or their published URLs (associated with two or more respective resources in the content provider's library).

6. **Claim 7: A method as recited in claim 1, wherein the step of directing the client to a second server in a second tier is performed in response to a step of using a hash to determine which second server in the second tier stores the first portion of the content provider's library.**

160. If both 1[c] and 1[d] must be performed (§VIII (Claim Construction)), in my opinion, Seed combined with Swildens teaches or suggests this additional limitation.

161. Specifically, although Seed discloses using recursive searching and redirection to direct the client to a parent server in the parent-level servers (*second tier*) that has the requested object, for the reasons I explained previously, it would have been obvious to use a hash function to determine which parent server (*second server*) is storing the requested object in its portion of the library, as taught by Swildens. §§IX.A.1.e-IX.A.1.f (1[d][i]-[ii]).

7. **Claim 8: A method as recited in claim 1 wherein the step of determining whether the resource is popular comprises determining whether a current popularity value for the resource exceeds a popularity threshold.**

162. In my opinion, Seed discloses or suggests this additional limitation.

163. Seed discloses that “an object is replicated (at 520) into storage”—and specifically because that the object has been determined to be popular—“when the

popularity P of the object is greater (at 500) than the initial threshold.” Seed, [0044]; *see also id.*, [0035]-[0040] (providing algorithm for calculating popularity), [0041]-[0043] (recommending a dynamic rather than fixed threshold).

8. Claim 9: A method as recited in claim 8, wherein the first tier of servers comprises a first popularity threshold and the second tier of servers comprises a second popularity threshold, and wherein the first popularity threshold is different from the second popularity threshold.

164. In my opinion, Seed discloses or suggests this additional limitation.

165. Seed discloses using a dynamic threshold that depends on the storage capacity of each server (Seed, [0041]-[0044]), and notes that both edge and parent servers determine thresholds, (*id.*, [0034], [0044], Fig. 3(b) (noting “purge [less-popular] object(s) from ... parent server”), Fig. 5 (dynamic threshold determination)).

166. In my opinion, therefore, a POSITA would have understood that edge servers (among a group of edge-level servers) would have different popularity thresholds based on their own storage capacities (*the first tier of servers comprises a first popularity threshold*), and parent servers (among a group of parent-level servers) would have separate thresholds based on their own storage capacities (*the second tier of servers comprises a second popularity threshold, wherein the first popularity threshold is different from the second*).

9. Claim 10

a. 10[pre]: A content delivery framework comprising:

167. In my opinion, if the preamble is limiting, Seed discloses or suggests it.

168. Specifically, as I explained previously, Seed discloses or suggests a content-delivery framework, including a CDN and method for using one. *See* §IX.A.1.a (1[pre]).

b. 10[a]: a plurality of edge servers forming a first tier of servers;

169. In my opinion, Seed discloses or suggests this limitation.

170. Specifically, as I explained previously, Seed discloses or suggests a plurality of edge servers (*first tier*). *See* §IX.A.1.a (1[pre]).

c. 10[b]: a plurality of parent servers forming a second tier of servers, the parent servers being distinct from the edge servers; and

171. In my opinion, Seed discloses or suggests this limitation.

172. Specifically, as I explained previously, Seed discloses or suggests a plurality of parent servers (*second tier*). *See* §IX.A.1.a (1[pre]).

- d. **10[c]: at least one popularity service constructed and adapted to obtain a first set of information from at least some of the edge servers about requests for content, and to provide the edge servers with a second set of information about which tier of servers should handle those requests for content, the popularity service making its determinations based, at least in part, on a measure of popularity of requested content; and**

173. In my opinion, Seed combined with Swildens teaches or suggests this limitation.

174. Seed discloses that each edge and parent server makes its own popularity determinations in response to their individual thresholds. *See, e.g.,* Seed, [0026], [0029], [0041]-[0044]. However, Seed does not expressly disclose a separate “popularity service” that receives information from edge servers and then provides information back to edge servers.

175. Swildens teaches that rather than having each edge server make its own popularity determination, however, as an alternative, “an auto migration server [(popularity service)] collects all of the log files...from the edge servers [(constructed and adapted to obtain a first set of information from...the edge servers)]...and tabulates the miss count...for every piece of content” (*the popularity service making its determinations based...on a measure of popularity of the requested content*) to determine whether any content is sufficiently popular based on a threshold such that it should be auto-migrated to the edge (*provide the edge servers*

with a second set of information about which tier of servers should handle those requests for content). Swildens, [0126], [0128], [0130].

176. In my opinion, a POSITA would have been motivated to implement Swildens' "auto migration server" in Seed's network to predict the popularity of objects based on information (such as the total number of miss counts) provided by multiple edge servers, collectively, rather than having each edge server determine popularity separately. A POSITA would have appreciated, for instance, that having a centralized server make popularity determinations (in addition to allowing individual servers to also make that determination) would allow the centralized server to identify objects of rapidly increasing popularity across the network, and to adjust by instructing the edge servers to pre-emptively replicate the content, thus reducing the load on parent or origin servers in the run-up.

177. In my opinion, a POSITA would have had a reasonable expectation of success in making this modification because it merely involves configuring a server central to one or more clusters of edge servers (like Seed's parent server, or a separate "auto migration server") to compile log information from the edge servers that they would each already have (such as miss counts) to make a popularity determination that would apply to a plurality of edge servers.

- e. **10[d]: wherein the requested content is available as part of a content provider's library, and wherein distinct portions of the content provider's library are logically partitioned across parent servers in the second tier.**

178. In my opinion, Seed discloses or suggests this limitation.

179. Specifically, as I explained previously, Seed discloses or suggests that the requested content is available as part of a content provider's library (§IX.A.1.b (1[a])), and Seed combined with Swildens teaches or suggests storing distinct portions of the content provider's library that are partitioned using a hash function (*logically*) across parent-level servers (*second tier*). §§IX.A.1.f (1[d][ii]), IX.A.2 (cl. 2).

- 10. **Claim 11: A content delivery framework as recited in claim 10, wherein at least one of the parent servers in the second tier comprises a first logically partitioned portion of the content provider's library, the first logically partitioned portion comprising at least the requested content.**

180. In my opinion, Seed combined with Swildens teaches or suggests this limitation.

181. Specifically, as I explained previously, Seed combined with Swildens teaches or suggests storing distinct portions of the content provider's library that are partitioned using a hash function (*logically*), including a first portion with the requested content, on one parent server in the parent-level servers (*second tier*). §§IX.A.1.f (1[d][ii]), IX.A.2 (cl. 2).

- 11. Claim 12: A content delivery framework as recited in claim 11, wherein at least one other parent server in the second tier comprises a second logically partitioned portion of the content provider's library, the second logically partitioned portion being distinct from the first logically partitioned portion.**

182. In my opinion, Seed combined with Swildens teaches or suggests this limitation.

183. Specifically, as I explained previously, Seed combined with Swildens teaches or suggests storing distinct portions of the content provider's library that are partitioned using a hash function (*logically*), including a second portion on another parent server among the parent-level servers (*second tier*) distinct from the first portion. §§IX.A.1.f (1[d][ii]), IX.A.2 (cl. 2).

- 12. Claim 13: A content delivery framework as recited in claim 10, wherein the second set of information further includes identification of a parent server in the second tier that stores a logically partitioned portion of the content provider's library comprising the requested content.**

184. In my opinion, Seed combined with Swildens teaches or suggests this limitation.

185. Specifically, as I explained previously, Seed combined with Swildens teaches or suggests using a hash function (*logic*) to partition and then identify which of a plurality of parent servers (*second tier*) is storing a portion of a content provider's collection (*library*). §§IX.A.1.f (1[d][ii]), IX.A.2 (cl. 2), IX.A.9.e (10[d]). Seed combined with Swildens also teaches using a central server (like Swildens'

auto-migration server) to make collective popularity determinations for a plurality of edge servers. §IX.A.9.d (10[c]).

186. When incorporating a hash function and central popularity-decision server into Seed's network, in my opinion, a POSITA would have appreciated that the central server would send information to the edge servers that maps the content identifiers (keys) to the parent server in a parent level of servers (buckets) that would store the portion of the content provider's collection (*library*) that stores the requested content. *See* Swildens, [0065].

13. Claim 14: A content delivery framework as recited in claim 10, wherein the second tier of parent servers is any intermediate tier between the first tier and an origin server that stores the content provider's library.

187. In my opinion, Seed discloses or suggests this limitation.

188. Specifically, as I explained previously, Seed discloses or suggests the parent-level servers are intermediate between the edge servers (*first tier*) and an origin server that stores the content provider's library. §IX.A.1.g (1[d][iii]).

14. Claim 15

a. 15[pre]: A method of content delivery in a content delivery network comprising a plurality of tiers of servers, including at least a first tier of servers and a second tier of servers, the method comprising:

189. In my opinion, if the preamble is limiting, Seed discloses or suggests it.

190. Specifically, as I explained previously, Seed discloses a method of content delivery in a CDN comprising edge-level (*first tier*), parent-level (*second tier*), and origin-level servers (collectively, *a plurality of tiers*). §IX.A.1.a (1[pre]).

- b. 15[a]: at a first server in one tier of the tiers of servers, obtaining a request from a client for a resource, wherein the resource is available as part of a content provider's library;**

191. In my opinion, Seed discloses or suggests this limitation.

192. First, the prior art can be mapped to this claim limitation in two different ways based on dependent claims 17 and 18, because although this limitation recites a “first server” in “one tier of the tiers of servers,” dependent claims 17 and 18 specify the “first server” may be two different types of servers in two different tiers. Specifically, for both dependent claims 17 and 18, the first-tier servers are edge servers and the second-tier servers are parent servers, but dependent claim 17 specifies the first server is an *edge* server, and dependent claim 18 specifies the first server is a *parent* server. In other words, this claim can be read as requiring:

Dependent claim 17	Dependent claim 18
“at [an edge server] in [a first tier of servers]....”	“at [a parent server] in [a second tier of servers]”

Although the prior art can be mapped to this claim limitation in two different ways based on dependent claims 17 and 18 (as well as other limitations, addressed below), I also understand that the prior art needs to only be mapped to the claims one way—

not both—for this limitation to be taught by the prior art. To avoid repeating the analysis for dependent claims 17 and 18, however, I address both here.

193. For dependent claim 17, as I explained previously, Seed discloses an edge server (*first server*) among edge-level servers (*one tier of the tiers of servers*) obtaining a request from a client for an object (*resource*) available as part of a content provider's library. §IX.A.1.b (1[a]).

194. For dependent claim 18, Seed discloses a parent server (*first server*) among parent-level servers (*one tier of the tiers of servers*) obtaining a request from a client for an object (*resource*)—after an edge server cannot locate a requested object and redirects client to make a request of the parent, (Seed, [0028])—available as part of a content provider's library, (§IX.A.1.b (1[a])).

- c. **15[b]: if the resource is not available at the first server or at a peer of the first server, then selectively redirecting the request from the client to a second tier server, the second tier server being in a tier distinct from the tier containing the first server, the redirecting being based, at least in part, on a current popularity value for the resource;**

195. If 15[b] must be performed (§VIII (Claim Construction)), in my opinion, Seed discloses or suggests this limitation.

196. Like 15[a], this limitation can also be read in at least two ways based on dependent claims 17 and 18. Specifically, this claim can be read as requiring:

Dependent claim 17	Dependent claim 18
“if the resource is not available at [the edge server]...then selectively redirecting the request...to a second tier server...distinct from the tier containing [the edge server]”	“if the resource is not available at [the parent server]...then selectively redirecting the request...to a second tier server...distinct from the tier containing [the parent server]”

197. For dependent claim 17, as I explained previously, Seed discloses that if the object (*resource*) is not available at the edge server (*first server*), the request is redirected to a parent-level server (*second-tier server*) distinct from the edge-level servers (*first tier*) containing the edge server (*first server*). §§IX.A.1.a (1[pre]), IX.A.1.e (1[d][i]). Seed also discloses the request is redirected when objects do not have a current popularity value that exceeds a popularity threshold. §§IX.A.1.d (1[c]) (noting edge servers store popular content), IX.A.1.e (1[d][i]) (noting edge servers redirect to parent servers for unpopular content), IX.A.7 (cl. 8) (using a popularity value for the popularity determination).

198. Dependent claim 18, however, appears to produce a possible contradiction in this limitation, because it requires that “if the resource is not available at a parent server,” which claim 18 specifies is in *the second tier*, “then selectively redirecting the request from the client *to a second tier server*”—in other words, another parent server—“*the second tier server being in a tier distinct from the tier containing the [the parent server in the second tier]*.” In other words, the

claim requires redirecting a request from one second-tier parent server to another second-tier parent server, but also that the two parent servers are in distinct tiers.

199. For purposes of analyzing this claim in view of the prior art, I have assumed one of three possible interpretations can apply.

200. **First**, this limitation could be read as “if the resource is not available at [a parent server in the second tier]...then selectively redirecting the request from the client to a second tier server.” In other words, the request is *redirected to a peer-parent server*, but that would be “in a tier distinct from the tier containing the first server” as also required by the claim. Under this interpretation, Seed discloses that “if a parent server does not have the requested object, the parent server could itself use a redirection technique recursively (at 325, 335, 320) until a final parent server is reached that has the requested object.” Seed, [0029]. In other words, Seed discloses redirecting a request from a parent server (*first server*) among parent-level servers (*second tier*) to at least another parent server also among parent-level servers (*second-tier server*).

201. However, as I mentioned previously, a POSITA would have been motivated to replace Seed’s recursive redirecting with Swildens’ targeted hash function. §IX.A.1.f(1[d][ii]). When doing so, in my opinion, a POSITA would have appreciated that one parent server may use a hash function to redirect to another parent server, because Swildens acknowledges that the hash function points only to

the server that “*should*” store the requested file, but that is prone to adjustment and error. Swildens, [0046]. When the first intermediate server *should*, but does not have the requested content, in my opinion, a POSITA would have appreciated a further redirect to another intermediate server that *should* also have the requested file would be required.

202. **Second**, this limitation could be read as “if the resource is not available at [a parent server in the second tier]...then selectively redirecting the request from the client to a second tier server, the second tier server being in a tier distinct from the tier containing the first server.” In other words, the “second tier” of parent servers has a **multi-tier substructure of parent servers**. Under this interpretation, Seed discloses that “[a]s will be apparent to those skilled in the art, more than one layer or level of parent servers...may be used.” Seed, [0022]. Seed therefore envisions multiple layers (*tiers*) of parent servers between the edge and origin tiers. When redirecting from one parent server (*first server in the second tier*) to another (*second-tier server*) (Seed, [0029]), in my opinion, a POSITA would have considered it obvious to redirect a request from one level of parent servers (*second tier*) to a distinct level of parent servers (a distinct *second tier*). As one example, one layer of parent servers closer to the edge may be configured to store objects that are less popular than those at the edge, but more popular than those stored by other lower-tier parent servers further from the edge.

203. **Third**, this limitation could be read as “if the resource is not available at [a parent server in the second tier]...then selectively redirecting the request from the client to a *second server*”—which claim 18 specifies is an “*origin server*”—with “*the second [origin] server* being in a tier distinct from the tier containing the [the parent server in the second tier]”. In other words, this interpretation would require rewriting “second tier server” to “second server.” Under this interpretation, Seed discloses that “where the parent servers [*first server in the second tier*] collectively are not populated with all of the objects and the network has access to the origin server of a requested object, the client may be redirected (at 225, 320) to the origin server (*second server*) if the requested object is not available on the parent servers,” Seed, [0031], at a distinct origin level (*tier*) from the parent servers (*second tier*), §IX.A.1.a (1[pre]).

204. For these reasons, in my opinion, Seed discloses this limitation under any possible interpretation. Additionally, under any interpretation, a POSITA would have understood that any redirecting would be “based, at least in part, on a current popularity value for the resource” because Seed discloses that the parent servers redirect requests only *after* the edge server has determined an object is not popular. Seed, [0026].

- d. **15[c]: wherein the second tier server comprises a first portion of the content provider's library, the first portion comprising at least the resource, wherein at least one other second tier server comprises a second portion of the content provider's library distinct from the first portion, and wherein the second tier of servers is any intermediate tier between the first tier and an origin server that stores the content provider's library, and**

205. If 15[c] must be performed (§VIII (Claim Construction)), in my opinion, Seed alone or combined with Swildens teaches or suggests this limitation.

206. Specifically, both dependent claims 17 and 18, specify the “first tier” is an edge tier of servers and the “second tier” is a parent tier of servers.

207. As I explained previously, Seed combined with Swildens discloses partitioning the content provider’s library across parent servers (*second-tier servers*), such that each parent server stores distinct portions of the content provider’s library, including one parent server with one portion that would have the requested object (*resource*), and a separate parent server with a distinct portion, with that the parent servers (*second tier*) being intermediate to the edge-level servers (*first tier*) and an origin server (e.g., customer’s server) that stores the content provider’s library. §§IX.A.1.f-IX.A.1.g (1[d][ii]-[iii]).

- e. **15[d]: wherein the step of selectively redirecting the request from the client to a second tier server comprises determining to which second tier server the request should be directed, the determining step using a hash function to identify a second tier server that stores the first portion of the content provider's library.**

208. If 15[d] must be performed (§VIII (Claim Construction)), in my opinion, Seed combined with Swildens teaches or suggests this limitation.

209. Specifically, both dependent claims 17 and 18, specify the “first tier” is an edge tier of servers and the “second tier” is a parent tier of servers.

210. As I explained previously, Seed combined with Swildens teaches redirecting a request from the client to a parent server (*second-tier server*) by determining to which parent server the request should be redirected using a hash function to identify the parent server that stores the portion of the content provider's library with the requested object. §§IX.A.1.e-IX.A.1.f (1[d][i]-[ii]; redirecting), IX.A.6 (cl. 7; hash function).

- 15. Claim 17: A method as recited in claim 15 wherein the first tier of servers comprises a plurality of edge servers and the second tier of servers comprises a plurality of parent servers, and wherein the first server is an edge server in the first tier and the second server is a parent server in the second tier.**

211. In my opinion, Seed discloses or suggests this additional limitation.

212. Specifically, as I explained previously, Seed combined with Swildens teaches or suggests the method of claim 15, in which the first-tier servers are edge

servers, the second-tier servers are parent servers, the first server is an edge server, and the second server is a parent server. §§IX.A.14.a-IX.A.14.e (15[pre]-[d]).

- 16. Claim 18: A method as recited in claim 15 wherein the first tier of servers comprises a plurality of edge servers and the second tier of servers comprises a plurality of parent servers, and wherein the plurality of tiers of servers further comprises an origin tier, and wherein the first server is a parent server in the second tier and the second server is a server in the origin tier.**

213. In my opinion, Seed discloses or suggests this additional limitation.

214. Specifically, as I explained previously, Seed combined with Swildens teaches or suggests the method of claim 15, in which the first-tier servers are edge servers, the second-tier servers are parent servers, the first server is a parent server, and the second server is either another parent server (in the same or a distinct level of parent servers) or an origin server. §§IX.A.14.a-IX.A.14.e (15[pre]-[d]).

B. Ground 2: Swildens in view of Seed (Claims 1, 2, 4-8, 10-15, 17, 18)

1. Claim 1

- a. 1[pre]: A method of content delivery in a content delivery network comprising at least a first tier of servers, the method comprising:**

215. In my opinion, if the preamble is limiting, Swildens discloses or suggests it.

216. Specifically, Swildens discloses a method of content delivery in a content-delivery network comprising at least a tier of edge servers (*first tier*).

217. Swildens discloses “[a] method...for the automatic migration of data via a distributed computer network [that] allows a customer to select content files that are to be transferred to a group of edge servers” for delivery to users. Swildens, Abstract, [0018]; *see also id.*, [0006] (“there is a clear need for a system that provides for the intelligent distribution of an information provider’s multimedia files across the Internet.”).

218. Even though Swildens does not use the phrase “content delivery network,” Swildens describes processes to deliver content to users through the network, which in my opinion would have indicated to a POSITA that Swildens was nonetheless directed to a content delivery network. *See, e.g., id.*, [0031]-[0039], [0046], [0135]-[0137], Fig. 5 (depicting steps for “Delivery of popular content”), Fig. 6 (depicting steps for “Delivery of unpopular content”). For instance, throughout many of the figures, Swildens provides example URLs exchanged between network elements that include the acronym “cdn.” *See, e.g.,* Swildens, Fig. 5 (depicting “Delivery of popular content” using URLs with “cdn”), Fig. 6. A POSITA would have understood that this is a telltale sign that the network element to which the URL is pointing is part of a “content delivery network” (“cdn”).

219. Swildens also discloses the network comprises tiers of “edge servers,” “origin servers,” a “staging server,” and a “customer’s server.” Swildens, [0042], [0052]-[0055], Figs. 1, 2, 5, 6. I have provided annotated versions of some of

Swildens' figures below that show the customer-facing terminal tiers in red, the intermediate "origin" tier in purple, and the client-facing terminal tiers in blue.

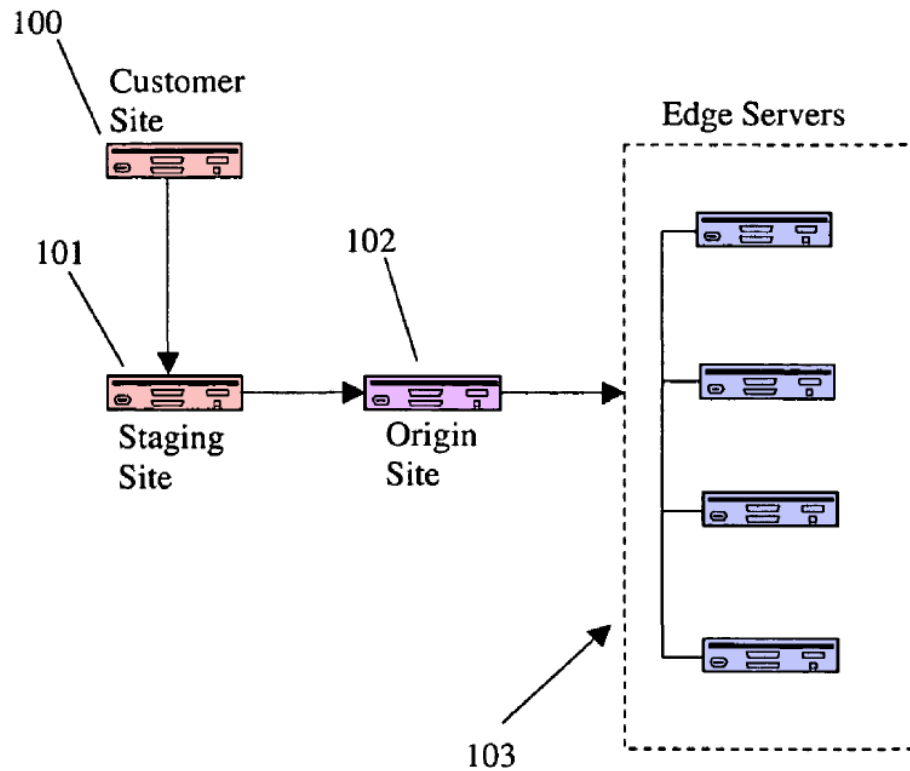
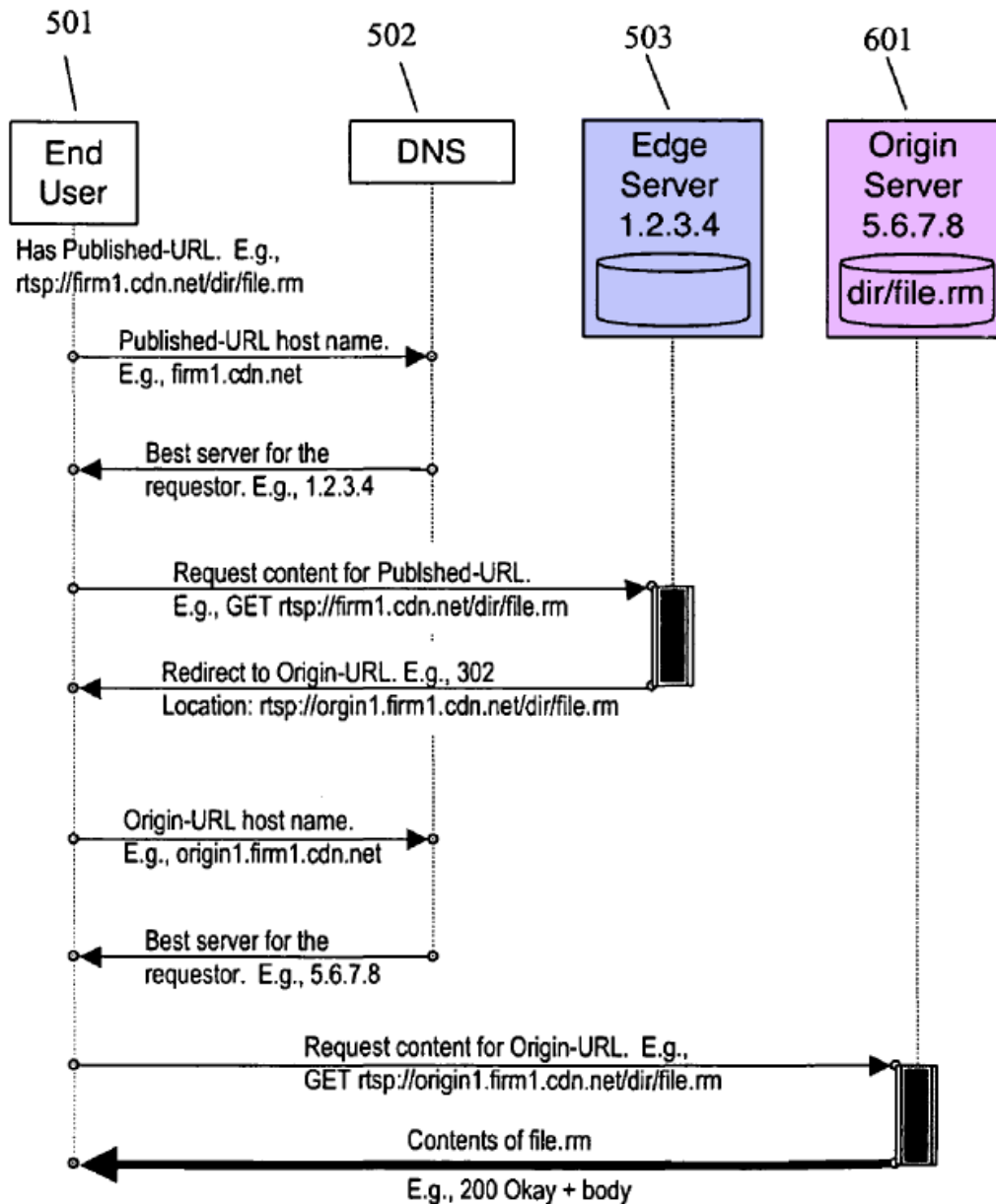


Fig. 1



Delivery of unpopular content

Fig. 6

- b. 1[a]: at a first server in the first tier of servers, obtaining a request from a client for a resource, wherein the resource is available as part of a content provider's library;

220. In my opinion, Swildens discloses or suggests this limitation.

221. Specifically, Swildens discloses an edge server (*first server*) among a group of edge servers (*first tier*) that obtain a request from a client for a content file (*resource*) available as part of a content provider's collection (*library*).

222. Swildens discloses that “[a]n edge server [(*first server*)] receives a user's request for content” (Swildens, [0046], [0068]), among a group of edge servers (*first tier*), (*see, e.g., id.*, [0009] (“FIG. 2 illustrates a block diagram of a user accessing popular content *from a group of edge servers*”), [0044]-[0045], Fig. 2).

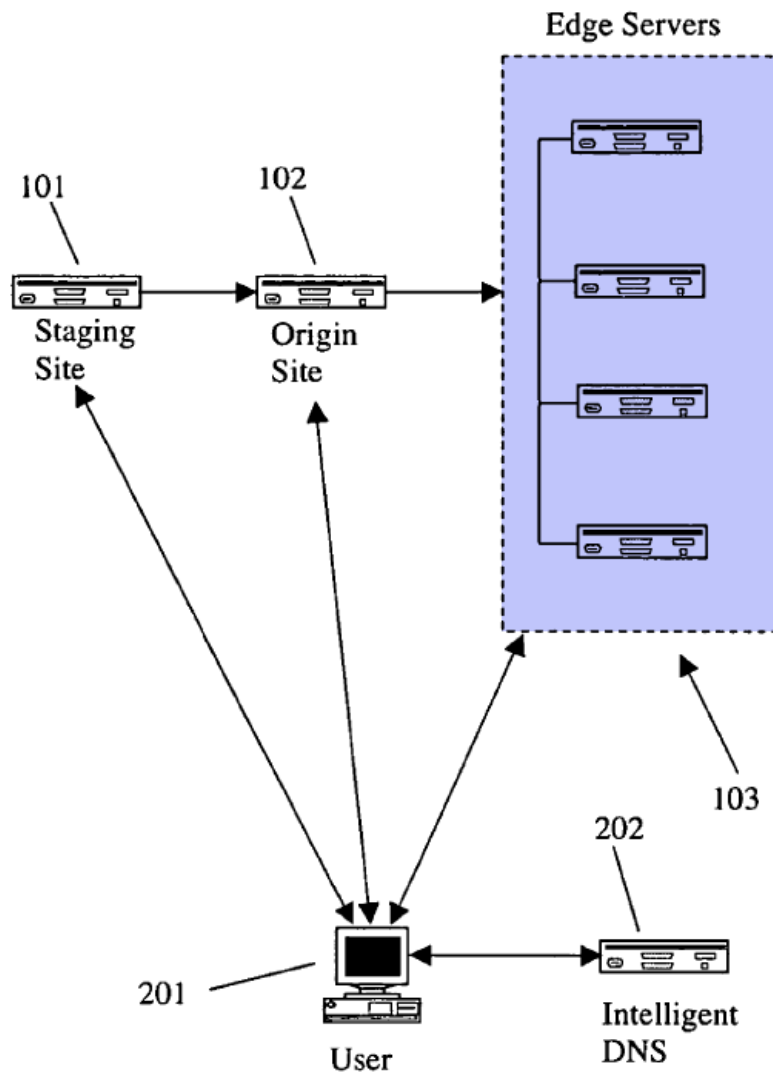


Fig. 2

223. Swildens also explains that “a customer...select[s] content files that are to be transferred to a group of edge servers” by uploading content files to the network through a staging server, and that origin servers “store all of a customer’s available content files.” Swildens, [0042]; *see also id.*, [0005] (“the number of files that are

actually accessed within *an information provider's collection* of multimedia files is low in comparison to the total number of files in the collection” (emphasis added)), [0030] (defining a “customer” as “[t]he provider of a subset of *an entire content collection*” (emphasis added)). In my opinion, a POSITA would have understood that the content file(s) requested by the clients comes from a collection (*library*) of content uploaded by a content provider/customer for distribution over the network.

c. 1[b]: if the resource is not available at the first server or at a peer of the first server, determining if the resource is popular;

224. In my opinion, Swildens, alone or combined with Seed, teaches or suggests this limitation.

225. Swildens discloses that “[a]n edge server maintains a dynamic number of popular files in its local memory for the customer.” Swildens, [0046], [0055]. When the edge server receives a request for a particular content file, “[i]f the edge server has the requested file, it then forwards or streams the file to the user 201 and records a hit for the file. If the edge server does not have the file, it records the miss in its running count and redirects the user 201 to the origin site 102.” *Id.*, [0068]; *see also id.*, [0046]. In other words, Swildens discloses performing an availability check at the first server for a requested content file, and records a hit or miss depending on whether it is available.

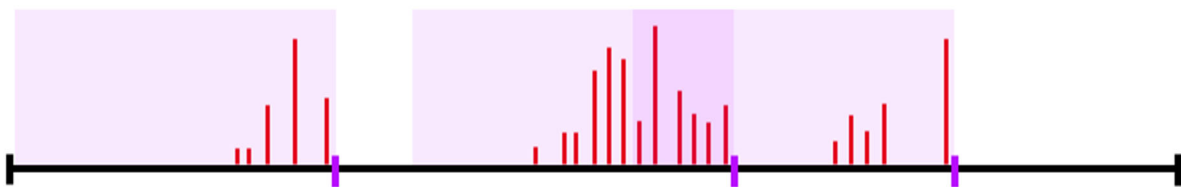
226. Swildens then discloses that “[t]he edge server checks the frequency of requests over a certain amount of time...to see if any new files have gained enough popularity to replace another file in the local storage.” *Id.*, [0068]. In my opinion, a POSITA would have understood that this is a separate popularity determination (*determining if the resource is popular*), which depends on the frequency of “missed” requests (*if the resource is not available*) for a file.

227. However, if this limitation requires using a “missed” request as a trigger to make a popularity determination, and not just that a popularity determination is made at some point after a “missed” request, in my opinion, Swildens would have been understood as at least suggesting this feature too. In some places, Swildens refers to checking the popularity of a missing file “over a certain amount of time (configurable by an administrator).” Swildens, [0068] (after a miss, evaluating “the frequency of requests over a certain amount of time”). At other times, Swildens discloses evaluating popularity “at some specified time interval” (using a centralized “auto migration server”). *Id.*, [0126].

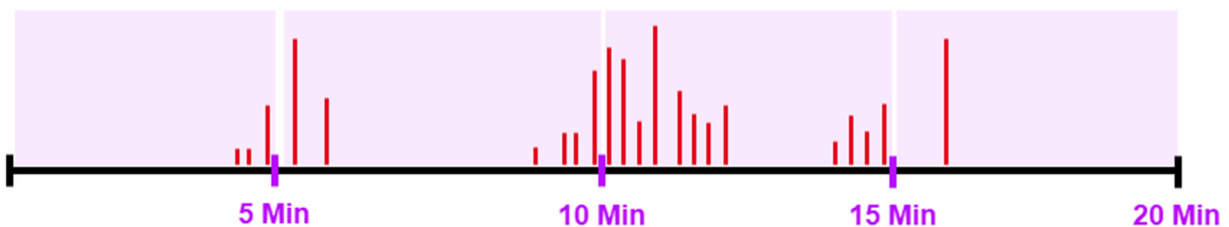
228. These are two different concepts. The former is specifying a ***window*** of time to make popularity determinations once a popularity determination is triggered, and the latter is specifying an ***interval*** to make popularity determinations. I have provided an exemplary timeline below showing the difference between these two concepts. The red bars exemplify the number of “missed” requests at a given time,

with bigger bars meaning more “missed” requests. The top figure shows popularity determinations being made due to some triggering event. It uses a window (represented by the purple box) of pre-defined time prior to the triggering event to see how many missed requests have occurred within the window to determine popularity. Windows can also be forward-looking in time. In this example, however, the server evaluates popularity three times, using three backwards-looking windows to evaluate popularity. The bottom figure shows popularity determinations being made periodically at predefined intervals (for instance, every 5 minutes), that simply looks at all miss counts that occurred within a given 5-minute interval.

Popularity Determination Window



Popularity Determination Interval



229. These two concepts can produce different results. For instance, in the top figure, assume adding all of the red bars together in the first (left-most) window includes a total of 20 misses. But using intervals, some of those same misses are included in one interval window, and some bars are included in another interval window, so the first interval may only count, for instance, 5 total misses.

230. Because Swildens appears to draw distinctions between popularity determination windows and popularity determination intervals, in my opinion, a POSITA would have understood that Swildens suggests, or at a minimum would have considered obvious, using the determination that a requested file is missing (e.g., incrementing the miss count) as a trigger to determine popularity over a predefined lookback window.

231. Regardless, even if Swildens cannot be read as disclosing using a “missed” request as a trigger to make a popularity determination, configuring Swildens to do so would have been obvious in view of Seed. Specifically, Seed teaches that “[i]f the selected edge server does not have the requested object, a check is initiated (at 315) for the edge server to determine whether the requested object is popular.” Seed, [0026], Fig. 3(a). In other words, Seed expressly teaches using a determination that a requested file is missing as a trigger to then determine whether the missing requested file is popular.

232. In my opinion, a POSITA would have been motivated to configure Swildens' edge servers to use the determination that a requested file is missing as a trigger to then determine the popularity of a missing file, consistent with Seed, to ensure edge servers update their caches immediately after determining files are missing. In my opinion, configuring Swildens this way would improve the quality of service for content deliveries, particularly for files that rapidly become popular by allowing servers to identify missing, popular files earlier, and reducing the number of redirects to other servers.

233. A POSITA would have understood that this could have been implemented several obvious ways consistent with both Swildens and Seed. As one option, the edge server would be configured to determine popularity over a backwards-looking window every time a file is determined to be missing. This, however, may lead to many repetitive popularity determinations and require more computing resources at the server. As an alternative option, the edge server would be configured such that, after it is determined that a file is missing, the server sets a timer for a forward-looking window, and if a predetermined number of requests for the same missing file come in during that window, that would indicate the requested file is in fact popular. There are, of course, other ways Swildens and Seed could be implemented together.

234. In my opinion, a POSITA would have had a reasonable expectation of success in making this combination because it requires only configuring the trigger used to make the popularity determination; whether a predetermined interval or some event like determining a file is missing from the server. Changing what triggers the popularity determination would have involved some changes to computer code at the edge servers (e.g., timing loops or conditional statements) that, in my opinion, were well within the skills of a POSITA to implement.

- d. 1[c]: if the resource is determined to be popular, then the first server obtaining the resource and the first server serving the resource to the client, otherwise,**

235. If both 1[c] and 1[d] must be performed (§VIII (Claim Construction)), in my opinion, Swildens alone or combined with Seed teaches or suggests this limitation.

236. Specifically, Swildens alone or combined with Seed teaches or suggests that if the file (*resource*) is popular, the edge server (*first server*) obtains the file and then serves the file to the client, either in response to a subsequent request or by fetching and then serving the file from another server.

237. First, Swildens discloses that “[w]hen a file has been requested from an edge server a sufficient number of times to become more popular than the lowest popular stored file, then the file is obtained from the origin site 102” (*if the resource*

is determined to be popular, then the first server obtaining the resource). Swildens, [0055], [0068].

238. Although Swildens discloses obtaining the file from the origin server, Swildens also notes the client should be redirected to a server that has the file in the interim (*id.*, [0068]-[0069]), suggesting the edge server obtains the file from the origin server later. In other words, like Seed (§IX.A.1.d (Ground 1, 1[c])), when a popular object is missing from an edge server, Swildens does not expressly describe ***first replicating*** the popular file to the edge server and ***then serving*** it to the client.

239. However, in my opinion, a POSITA would have understood that clients routinely request delivery of the same content multiple times. For instance, each time a client accesses the same website or streams the same media, typically a new request is sent from the client to a server hosting the requested content. For this reason, although Swildens discloses redirecting the client to another server when the edge server determines a requested popular object is missing, and then replicating the popular object to the edge server later, a POSITA would have understood that any subsequent request from the client after the same object would be served by the edge server that would now be hosting it after replication (*the first server serving the resource to the client*).

240. Although I do not understand the claim this way, if the claim requires that the edge server serve the object to the client after only ***the first request*** (and not

subsequent requests for the same object), this would have been obvious to a POSITA in view of other prior art such as Seed.

241. Seed recognizes that conventional CDN networks routinely configure edge servers to “fetch objects” from other servers “when a requested object is not available on a handling edge server.” Seed, [0013]; *see also, e.g.*, Ex-1009, 25 (“One can view the replica server as a caching proxy which fetches the content from the origin server on a local cache miss.”). Seed, however, prefers redirecting first missed requests for a popular object to another server, rather than fetching and serving the popular object itself to avoid “having [the object] go through the parent path” that “can be slow.” Seed, [0013]-[0014]. But even so, Seed still describes its network as being configured to fetch objects. *See, e.g.*, Seed, [0046] (noting “the cost of fetching” is considered as part of the replication policies of the network), [0059] (suggesting partitioning objects into smaller chunks so that “a first part of an object can be served from edge servers quickly, even if most of the object stream ***must be fetched*** from a parent server or origin server.” (emphasis added)).

242. A POSITA would have appreciated that when a server receiving a request for an object did not have the requested object, there were only a handful of ways to fulfil the request. The most common ways for a server to facilitate delivery of the requested object included::

(1) Obtaining the object for itself and then serving it to the client (e.g., fetch-and-serve or stream-through). This is like when a store places an order for an out-of-stock item on behalf of the customer, and the store handles delivery once the product is received by the store.

(2) Sending or redirecting *the client* to another server that has the object. This is like a store informing a customer that another store has the requested item in stock, and the customer must go get it from the other store.

(3) Forwarding or redirecting *the request* to another server that has the object. This is like a store forwarding a customer's order for an out-of-stock item to another store that has the item in stock, and the other store handles delivery.

243. In my opinion, because there were only a finite number of ways (at a high level) for a server to facilitate the delivery of a requested but missing object, in my opinion, a POSITA would have been motivated to try any of those well-known ways to determine which worked best for the particular network and use case. Choosing one option over another would have been nothing but a simple substitution of one known element (e.g., fetch-and-serve) for another (e.g., redirecting-and-fetching) to achieve a predictable result (e.g., delivery of the requested object), or applying a known, suitable alternative technique (e.g., fetch-and-serve) to the one Seed discloses (e.g., redirecting-and-fetching) to solve a known problem in the art

(e.g., serving content from the server handling a request for a missing object). In fact, those in the art routinely recognized that fetching, redirecting, and forwarding were known alternatives. *See, e.g.*, Ex-1017, 2:38-42 (“the server uses the virtual URL system to locate the best current location for the resource and either redirects the requesting browser to it...or else fetches and returns the resource”); Ex-1018, [0040] (“the secure edge server 508 can forward requests to the origin, fetch content from the origin to deliver to the end user, and/or deliver content from its own cache as required.”).

244. Regardless, a POSITA would have understood that each technique has its own tradeoffs, and although Swildens and Seed appear to use redirecting (with Seed attributing that decision due to its speed), fetching would provide different benefits. For instance, fetching may be slower, but it may also be cheaper than redirecting. As one example, in typical CDNs, “[t]he customer is charged for the bandwidth used to transfer files from the origin servers to the edge servers.” Swildens, [0043]. When using redirects to fulfill requests for popular objects, if a handling edge server must redirect the request to, e.g., an origin server to fulfill because it is missing from the edge, the origin server needs to transfer the requested object to the client, leading to one cost associated with that transfer. Then, the handling edge server that originally received the request needs to replicate the popular object from the origin server, leading to a second cost for that transfer. In a

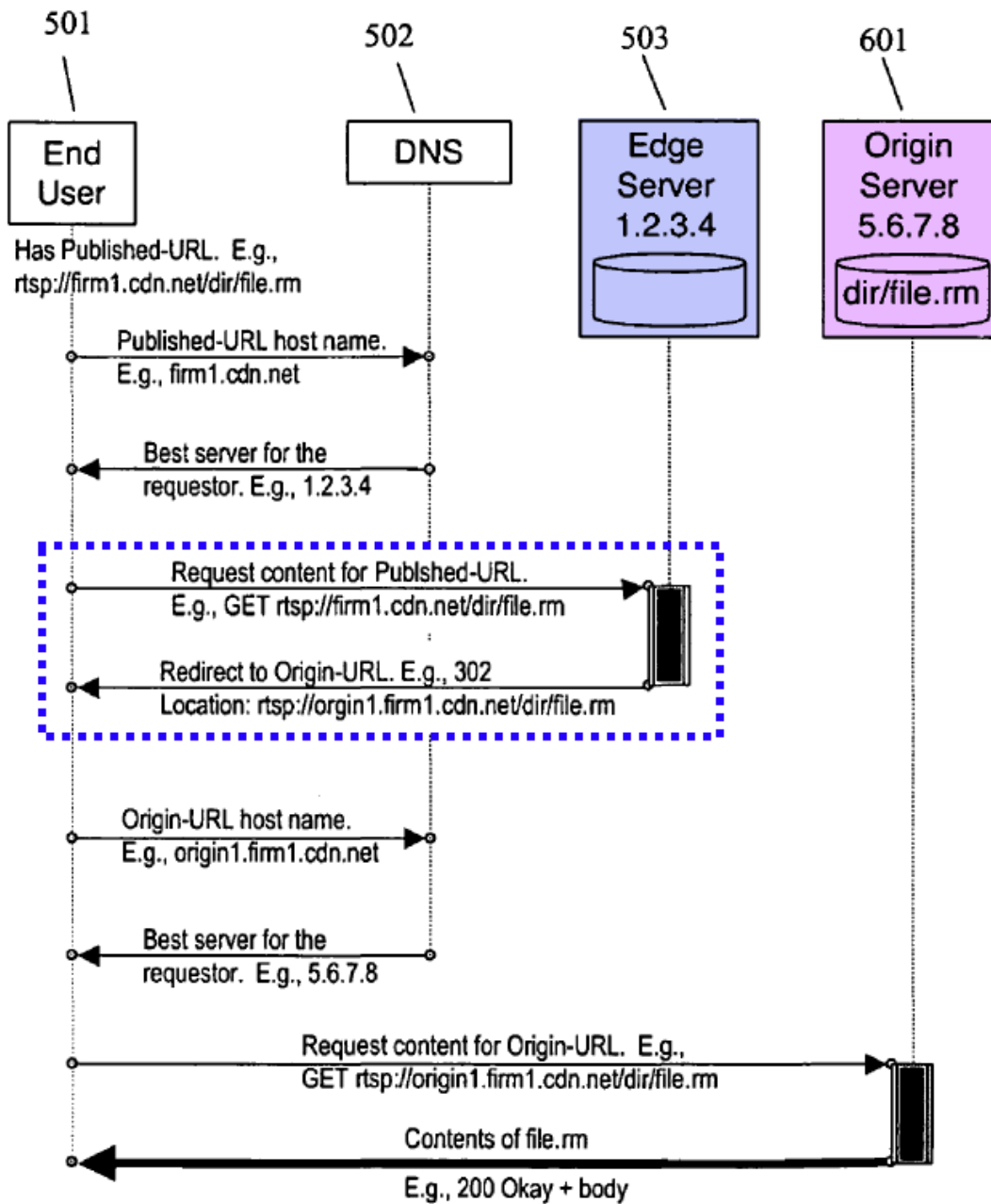
fetch-and-serve scheme, however, only one request is made of the origin server by the handling edge server, and that same transfer from the origin is used to both replicate the object to the edge *and* serve the requested content from the edge.

245. In my opinion, a POSITA would have also had a reasonable expectation of success in making this modification because both fetching and redirection were two of the most common ways to serve missing objects to clients. *See, e.g.,* Seed, [0046], [0059]; Ex-1017, 2:38-42. The only modification required would be to change some computer code to instruct edge servers how to handle a missed request for popular object: rather than direct the client to the location the server has identified so that the client can obtain the object for itself, to go fetch the object itself from the location it has identified (which the server is already programmed to do to obtain a copy for itself *after* redirecting the client) and then provide a copy to the client. In my opinion, the required changes to code and any further implementation details would have been well within the skills of a POSITA.

- e. **1[d][i]: if the resource is determined not to be popular, directing the client to a second server in a second tier of servers distinct from the first tier of servers,**

246. If both 1[c] and 1[d] must be performed (§VIII (Claim Construction)), in my opinion, Swildens alone or combined with Seed teaches or suggests this limitation.

247. First, Swildens discloses that an edge server “maintains a dynamic number of popular files in its memory.” Swildens, [0043]. Swildens discloses that when the edge server does not have the requested file—meaning the file either is not popular, or the edge server has not yet determined that the file is popular to obtain a copy—the server “redirects the user 201 to the origin site 102” among a group of origin servers. Swildens, [0068], Fig. 6; §IX.B.1.a (1[pre]).



Delivery of unpopular content
Fig. 6

248. In other words, Swildens discloses if the resource is determined not to be popular, directing the client to an intermediate (CDN-origin) server (*second*

server) in an intermediate (CDN-origin) group of servers (*second tier*) distinct from the edge servers (*first tier*).

249. Although Swildens alone meets this limitation, Swildens combined with other prior art also meets this limitation by mapping a different set of servers (e.g., “parent” servers) to the second tier of servers. Specifically, Swildens discloses the intermediate tier as being an “origin” tier, which a POSITA would have understood to refer to a CDN-origin tier. §IX.A.1.f (Ground 1, 1[d][ii], describing the different labels used by the ’692 patent, Swildens, and Seed, and how they would be understood to align). However, a POSITA would have understood that the exact structure of the network—including among other things how many tiers of servers to include, what types of servers to include in each tier, and the relationships between servers in a tier and between tiers—was a design choice based on the intended application of the network, balancing things such as performance and cost, as even Swildens recognizes. *See, e.g.*, IV (Technology Background, describing design considerations); Swildens, [0034], [0044]-[0045], [0077]-[0078], [0095]-[0101].

250. In my opinion, therefore, a POSITA would have also been motivated to incorporate an additional layer of servers—and specifically “parent” servers—between Swildens’ edge and origin servers when designing a network that emphasized performance over cost. Specifically, although Swildens only mentions edge and intermediate CDN-origin servers in its network, those in the art routinely

incorporated a third “parent” type between edge and CDN-origin servers, often strategically located in geographic regions near the edges to minimize load on the origin server(s), minimize latency, and maximize quality of server. Ex-1019, 1:12-64, 4:63-5:21, Fig. 3. Seed, for instance, expressly teaches another CDN that performed nearly identical functions as Swildens—storing popular content on edge servers, and unpopular content on intermediate servers (*compare, e.g.,* Swildens, [0055], [0068], *with* Seed, [0016], [0025]-[0028]), but stores the unpopular content on an intermediate tier of “parent” servers instead of CDN-origin servers like Swildens. *Compare, e.g.,* Swildens, [0055], [0068], *with* Seed, [0016], [0025]-[0028]. A POSITA would have recognized that incorporating an additional tier of parent servers would improve the performance of the CDN by providing caches of unpopular content closer to the edge, and allowing requests to be distributed across more servers.

251. Alternatively, however, if costs were still a concern, a POSITA would have understood that Swildens’ CDN-origin servers could be completely replaced by a set of parent servers consistent with Seed, because Swildens discloses its CDN-origin servers are configured to perform nearly the exact same functions as Seed’s parent servers (i.e., storing unpopular content further from the edge). In other words, the modification at most would only implicate some additional, implicit configuration differences between a CDN-origin server and parent server that neither

Swildens nor Seed expressly discuss or indicate any concern towards. In other words, this combination would involve a simple substitution of one known element (Swildens' intermediate-origin servers) with another (Seed's intermediate-parent servers) with no appreciable difference in the CDN's operation (storing and serving not popular content from the intermediate tier of servers).

252. In my opinion, a POSITA would have had a reasonable expectation of success in implementing an additional intermediate tier of "parent" servers (or replacing Swildens' CDN-origin servers with parent servers) consistent with Seed because implementing parent servers in a CDN was well known, and also because Swildens already envisions adjusting the number and arrangement of servers based on design considerations. Seed simply provides a framework to structure the tiers of servers to accomplish the same goals as Swildens of storing popular content close to the edge and unpopular content further from the edge.

253. Upon making this combination, a POSITA would have appreciated that Swildens combined with Seed teaches that if the resource is determined not to be popular, directing the client to an intermediate-parent server (*second server*) in an intermediate parent group of servers (*second tier*) distinct from the edge-level servers (*first tier*).

- f. **1[d][ii]: wherein the second server comprises a first portion of the content provider's library, the first portion comprising at least the resource, wherein at least one other server in the second tier comprises a second portion of the content provider's library, wherein the first portion of the content provider's library is distinct from the second portion, and**

254. If both 1[c] and 1[d] must be performed (§VIII (Claim Construction)), in my opinion, Swildens alone or combined with Seed teaches or suggests this limitation.

255. Swildens discloses that although its intermediate origin “servers together store the complete copies of all the content for a particular customer” (*id.*, [0050]), “[a]n individual origin server does not have to be able [to] hold all of the content” (*id.*, [0060]). Instead, Swildens discloses using a “hash function”—an algorithm to “map[] the large number of content identifiers across the much smaller number of storage servers” (*id.*, [0061])—to distribute distinct portions of a content provider’s library across a group of origin servers and identify on which server any given content resides (*id.*, [0060]). And although Swildens recommends some level of redundancy between intermediate origin servers to improve performance (Swildens, [0058]), unless the hash function is specifically configured to allow overlap for redundancy, by running content identifiers through a hash function, a POSITA would have understood that would typically result in each server having wholly distinct, non-overlapping portions of the library. §VIII (Claim Construction).

That is because running a content identifier (key) through a hash function would typically produce only one output, which a POSITA would have understood would be the server to which it is assigned (e.g., a one-to-one key-to-slot mapping).

256. For these reasons, in my opinion, Swildens discloses an intermediate CDN-origin server (*second server*) stores one portion of the content provider's library that would include the requested file (*resource*), but other intermediate CDN-origin servers (*other servers*) in the group of CDN-origin servers (*second tier*) would store other, distinct portions of the content provider's library.

257. However, when implementing an additional or alternative intermediate tier of parent servers consistent with Seed (§IX.B.1.e (1[d][i])), Swildens and Seed together teach an intermediate parent server (*second server*) stores one portion of the content provider's library that would include the requested file (*resource*), but other intermediate parent servers (*other servers*) in the group of parent servers (*second tier*) would store other, distinct portion of the content provider's library. §IX.B.1.e (1[d][i]), explaining that Seed's parent servers would be configured to operate like Swildens' CDN-origin servers, only closer to the edge).

- g. 1[d][iii]: wherein the second tier is any intermediate tier between the first tier and an origin server that stores resources associated with the content provider's library, and**

258. If both 1[c] and 1[d] must be performed (§VIII (Claim Construction)), in my opinion, Swildens alone or combined with Seed teaches or suggests this limitation.

259. Swildens discloses that the group of CDN-origin servers (*second tier*) are intermediate to the group of edge servers (*first tier*) and the customer/staging (customer-origin) server that stores files associated with the content provider's library. §§IX.B.1.a-IX.B.1.b (1[pre]-1[a]). As I explained previously, a POSITA would have understood that Swildens' customer/staging (customer-origin) server is another type of "origin server" consistent with the '692 patent. §IX.A.1.f (Ground 1, 1[d][ii], describing the different labels used by the '692 patent, Swildens, and Seed, and how they would be understood to align); *compare* Swildens, [0054] ("A customer selects multimedia files that are to be transferred to the edge servers," which are provided "from the customer site 100"), Fig. 1, *with* Ex-1001, 3:65-4:1 ("A content provider's/customer's server (or servers) are also referred to as origin servers," which "may be owned and/or operated by that content provider"), Figs. 3, 4; *see also* Swildens, [0056] ("The customer is charged for the amount of storage required on the origin site 102 to store the customer's files.... This allows the

customer to adjust the amount of storage anticipated for file demand,” meaning the customer does not own Swildens’ “origin” servers).

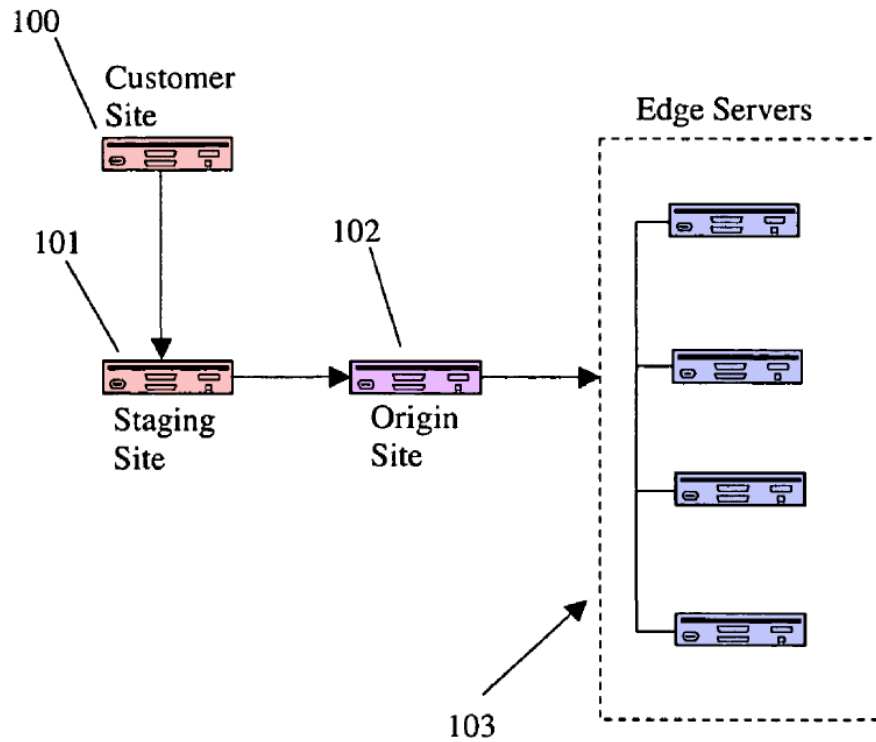
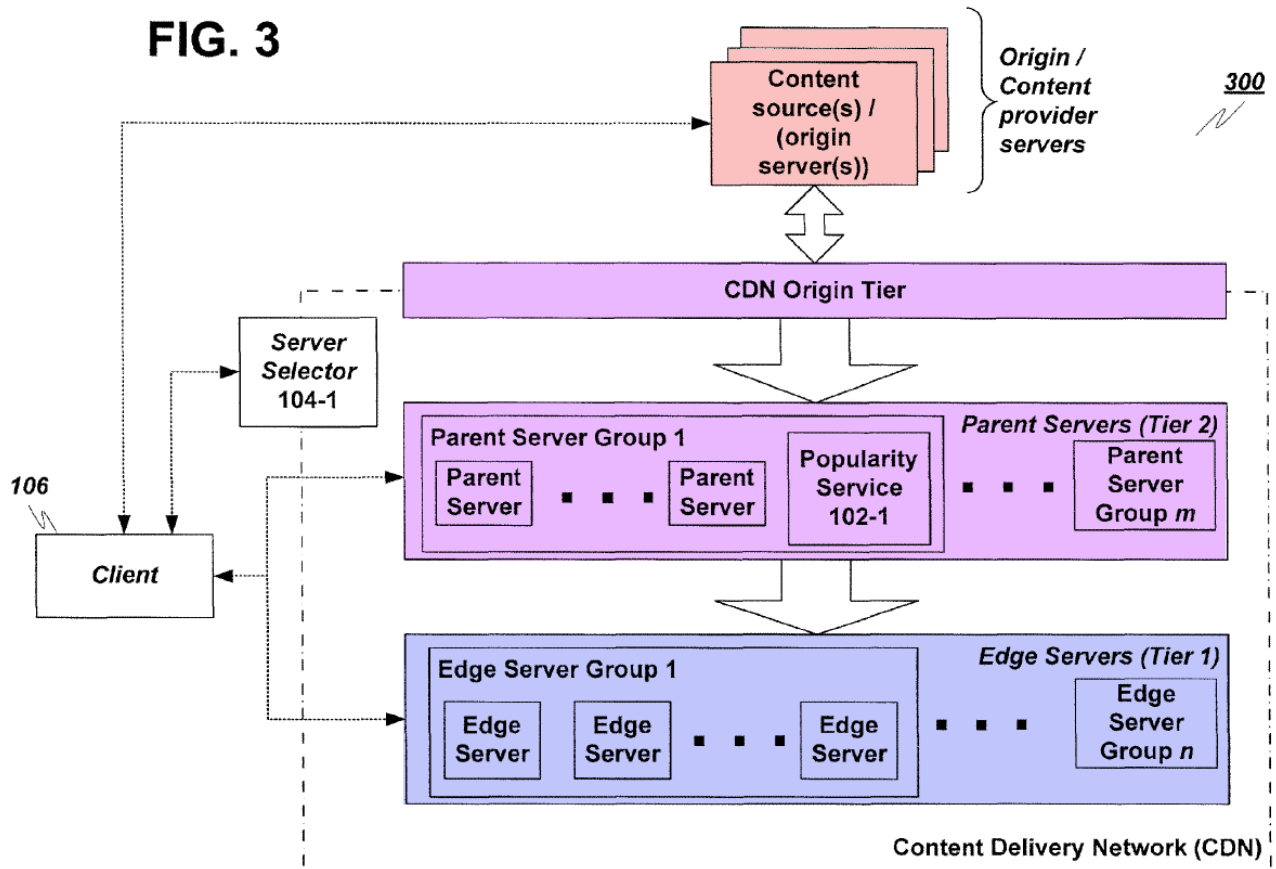


Fig. 1

FIG. 3



260. Regardless, even if Swildens' customer/staging (customer-origin) server cannot be considered an "origin" server, when implementing an additional or alternative intermediate tier of parent servers consistent with Seed (§IX.B.1.e (1[d][i])), Swildens and Seed together teach a parent group of servers (*second tier*) intermediate to the edge group of servers (*first tier*) and Swildens' origin servers that store files associated with the content provider's library.

h. 1[d][iv]: the second server serving the resource to the client.

261. If both 1[c] and 1[d] must be performed (§VIII (Claim Construction)), in my opinion, Swildens alone or combined with Seed discloses or suggests this limitation.

262. Swildens discloses the CDN-origin server (*second server*) serves unpopular or yet-to-be-replicated files to the client. Swildens, [0069] (“If the user is redirected to the origin site 102, the origin site 102 checks its local storage for the requested file. If the origin site 102 has the file, it sends or streams the file to the user 201.”), [0047] (same), [0102] (“The origin server can have the performance characteristics to deliver relatively unpopular content or content that has just become popular, but not yet replicated by the invention.”).

263. Alternatively, upon implementing an additional or alternative intermediate tier of parent servers consistent with Seed (§IX.B.1.e (1[d][i])), a POSITA would have been motivated and had a reasonable expectation of success in configuring the parent server (*second server*) to serve the unpopular or yet-to-be-replicated files to the client, consistent with Seed, for the reasons I explained previously. Seed, [0028]-[0029], Fig. 3(a); §IX.A.1.e (1[d][i]).

2. Claim 2: A method as recited in claim 1, wherein portions of the content provider's library are logically partitioned across servers in the second tier.

264. If both 1[c] and 1[d] must be performed (§VIII (Claim Construction)), in my opinion, Swildens alone or combined with Seed discloses or suggests this additional limitation.

265. Swildens discloses partitioning a content provider's collection across intermediate (CDN-origin) servers (*second tier*) using a hash function, and specifically "comput[ing] a hash...from a content file identifier" and storing the file at a specific intermediate-level (CDN-origin) server based on the hash. Swildens, [0051]-[0052]. Swildens also teaches that "[a] good hash function is one that spreads the content storage requirements of the content identifiers across the actual storage capacities of each of [intermediate] server group." *Id.*, [0061].

266. A POSITA would have understood that the hash function involves a logic-based, mathematical algorithm to convert, e.g., content identifiers (keys) into hashes for sorting into servers (buckets). In other words, Swildens discloses partitioning and distributing portions of a content provider's library across intermediate-level servers (*second tier*) using a hashing algorithm (*logic*).

267. Additionally, Swildens combined with Seed teaches partitioning a content provider's library across intermediate-parent servers (*second tier*) using

content-file identifiers and storage capacity of each intermediate server to partition (*logic*), for the reasons I have explained previously. §IX.B.1.e (1[d][i]).

3. Claim 4: A method as recited in claim 1, wherein the content provider's library comprises a plurality of content-types, each resource in the library being associated with one of the plurality of content-types.

268. In my opinion, Swildens discloses or suggests this additional limitation.

269. Swildens discloses the “multimedia files” distributed over the network include (among other things) “music and movies.” Swildens, [0003]; *see also id.*, [0129] (“popular content can vary from one edge location to another. For example, the top ten songs in the Atlanta area will be different from the top ten songs in London.”). In my opinion, a POSITA would have understood “music and movies” are “content types” as claimed, and that each file would be associated with one of the content types.

4. Claim 5: A method as recited in claim 4 further comprising storing distinct portions of the content provider's library on two or more servers in the second tier, wherein the content-types on each of the two or more servers are proportionally distributed, the content-types comprising at least one of audio content, video content, gaming content, and software content.

270. If both 1[c] and 1[d] must be performed (§VIII (Claim Construction)), in my opinion, Swildens alone or combined with Seed discloses or suggests this additional limitation.

271. As I explained previously, Swildens discloses storing distinct portions of a content provider's library across multiple different intermediate CDN-origin servers (*second tier*). §§IX.B.1.e-IX.B.1.g (1[d][i]-[iii]).

272. Additionally, Swildens combined with Seed teaches storing distinct portions of a content provider's library on multiple different intermediate parent servers (*second tier*). §§IX.B.1.e-IX.B.1.g (1[d][i]-[iii]).

273. Swildens also discloses the library may include multiple content types, including music (audio) and movies (video) content. §IX.B.3 (cl. 4).

274. In my opinion, based on these teachings, a POSITA would have appreciated that the file content types would be distributed across the intermediate servers—whether CDN-origin or parent—by some proportion. For instance, some CDN-origin/parent servers may store exclusively movies, and other CDN-origin/parent servers may store exclusively music. Alternatively, some CDN-origin/parent servers may distribute different movies across different CDN-origin/parent servers, because they were known to be much larger files, and thus more resource-intensive to serve to clients. Thus, providing, e.g., size- or resource-based distributions would provide an easy way to balance server load. *See, e.g.*, Ex-1020, [0038] (“For example, a replica of a movie content having title XXX is stored on node N5; a replica of a movie content having title YYY is stored on node N3. In

this way, contents are distributed and stored among multiple nodes N_n"); Ex-1021, 8:25-56 (describing heuristic distribution across servers).

5. **Claim 6: A method as recited in claim 1, wherein resources of the content provider's library are distributed in a substantially equal manner across servers in the second tier based on hash values of uniform resource locators (URLs) associated with two or more respective resources in the content provider's library.**

275. If both 1[c] and 1[d] must be performed (§VIII (Claim Construction)), in my opinion, Swildens alone or combined with Seed teaches or suggests this additional limitation.

276. As I explained previously, Swildens alone or combined with Seed discloses storing distinct portions of a content provider's library across multiple CDN-origin/parent level servers (*second tier*) using a hash function. §§IX.B.1.e-IX.B.1.f (1[d][i]-[ii]).

277. As part of the distribution, Swildens explains that "[a] good hash function is one that spreads the content storage requirements of the content identifiers across the actual storage capacities of each [intermediate] server group." Swildens, [0061]. In other words, Swildens discloses distributing content libraries in a way that accounts for the storage capacity of each server. When doing so, in my opinion, a POSITA would have appreciated that when implementing Swildens' and/or Seed's teachings in a network of CDN-origin/parent servers that all have the

same amount of storage, Swildens' hash function would produce a substantially equal distribution of content across the CDN-origin/parent level servers (*servers in the second tier*).

278. Swildens also explains that hash values are determined based on “content identifiers” or “published URLs” associated with the content (Swildens, [0061]-[0062], [0065]-[0066]), and “[r]edirected requests to [intermediate] servers have a hash included in the URL” that is used to locate which intermediate server is storing the requested object, (Swildens, [0060]). Swildens' hash function, whether applied on CDN-origin servers or parent servers consistent with Seed, would distribute content across the CDN-origin/parent level servers (*servers in the second tier*) based on hash values either included in URLs, or of URLs themselves (*based on hash values of uniform resource locators (URLs)*) associated with content via content identifiers or their published URLs (*associated with two or more respective resources in the content provider's library*).

6. **Claim 7: A method as recited in claim 1, wherein the step of directing the client to a second server in a second tier is performed in response to a step of using a hash to determine which second server in the second tier stores the first portion of the content provider's library.**

279. If both 1[c] and 1[d] must be performed (§VIII (Claim Construction)), in my opinion, Swildens alone or combined with Seed discloses or suggests this additional limitation.

280. For the reasons I have explained previously, Swildens discloses directing the client to an intermediate-origin server (*second server*) in an intermediate-origin tier (*second tier*) using a hash to determine which intermediate-origin server (*second*) in the tier (*second tier*) stores the portion of the library with the requested file (*resource*). §§IX.B.1.e-IX.B.1.g (1[d][i]-[iii]).

281. Additionally or alternatively, Swildens combined with Seed teaches directing the client to an intermediate-parent server (*second server*) in an intermediate-parent tier (*second tier*) using a hash to determine which intermediate-parent server (*second*) in the tier (*second tier*) stores the portion of the library with the requested file (*resource*). §§IX.B.1.e-IX.B.1.g (1[d][i]-[iii]).

7. Claim 8: A method as recited in claim 1 wherein the step of determining whether the resource is popular comprises determining whether a current popularity value for the resource exceeds a popularity threshold.

282. In my opinion, Swildens discloses or suggests this additional limitation.

283. Specifically, Swildens discloses determining whether the resource is popular includes determining whether a current rank (*popularity value*) for a file (*resource*) exceeds the lowest rank (*popularity threshold*) of the files stored at the edge. Alternatively, Swildens renders obvious determining whether the resource is popular includes determining whether a “miss count” (*popularity value*) for a file (*resource*) exceeds a miss-count threshold value (*popularity threshold*).

284. Swildens discloses that “when a file has been requested from an edge server a sufficient number of times to become more popular than the lowest popular stored file, the file is obtained from an origin site.” Swildens, [0043], [0055], [0068]. In my opinion, a POSITA would have understood that the request count or frequency of the “lowest popular stored file” serves as a “popularity threshold” as claimed for each edge server, because that is a threshold value that must be exceeded to replicate a missing file onto the edge server.

285. However, Swildens also discloses another embodiment of the network that uses a “miss count threshold value (e.g., ten misses in one hour)” to determine whether a requested (but missing) file is popular. Swildens, [0123]-[0133]. Swildens explains that using a “miss count threshold value” can reduce the number of files stored on edge servers by setting the threshold to an appropriate value. Swildens, [0131]-[0132].

286. Although Swildens discloses the miss-count threshold in a separate embodiment, in my opinion, a POSITA would have been motivated to apply a similar “miss count threshold value” to popularity determinations made at edge servers instead of Swildens’ rank-order threshold to reduce transmission costs to update the edge server. Swildens demonstrates the difference between these two options. Specifically, with a rank-order threshold, the number of files that may be stored at the edge server may be limited only by the available amount of memory.

As such, files that may not be considered truly “popular,” but are nonetheless more popular than other files may be hosted at the edge. Swildens provides an example in which an edge server stores nine files based on miss-count rank:

Name	MissCount
f1	234
f2	157
f3	50
f4	35
f5	20
f6	15
f7	4
f8	3
f9	1

Swildens, [0131].

287. In other words, if another file “f10” then had two miss counts, it may replace file “f9” at the edge even though two miss counts may not be considered “popular.”

288. In the alternative, a miss-count threshold could be set to a specific number, which would limit the edge server to storing only files that have been deemed sufficiently popular. Swildens, for instance, provides an example where the miss-count threshold is set to “10,” which means three files—f7, f8, and f9—would not be stored at the server like they might by under a rank-order threshold.

Name	MissCount
f1	234
f2	157
f3	50
f4	35
f5	20
f6	15

Swildens, [0132].

289. A POSITA would have therefore appreciated that by providing a miss-count threshold, edge servers would not be routinely updating their caches by swapping out marginally popular files (files that have a handful of errant misses here and there) simply due to rank, but instead would limit updates to files that hit a higher threshold-measure of popularity. This would reduce transmission costs because less file transfers from the intermediate server(s) would occur, providing users the ability to better control how many popular files would be stored at the edge, and providing a way to set an objective floor for popularity. Swildens, [0043] (“The customer is charged for the bandwidth used to transfer files from the origin servers to the edge servers.”).

8. Claim 10

a. 10[pre]: A content delivery framework comprising:

290. In my opinion, if the preamble is limiting, Swildens discloses or suggests it.

291. Specifically, as I explained previously, Swildens discloses or suggests a content-delivery framework, including a CDN and method for using one. *See* §IX.B.1.a (1[pre]).

b. 10[a]: a plurality of edge servers forming a first tier of servers;

292. In my opinion, Swildens discloses or suggests this limitation.

293. Specifically, as I explained previously, Swildens discloses or suggests a plurality of edge servers (*first tier*). *See* §IX.B.1.a (1[pre]).

c. 10[b]: a plurality of parent servers forming a second tier of servers, the parent servers being distinct from the edge servers; and

294. In my opinion, Swildens combined with Seed discloses or suggests this limitation.

295. Specifically, as I explained previously, Swildens combined with Seed discloses or suggests a plurality of parent servers (*second tier*). *See* §IX.B.1.e (1[d][i]).

d. 10[c]: at least one popularity service constructed and adapted to obtain a first set of information from at least some of the edge servers about requests for content, and to provide the edge servers with a second set of information about which tier of servers should handle those requests for content, the popularity service making its determinations based, at least in part, on a measure of popularity of requested content; and

296. In my opinion, Swildens discloses or suggests this limitation.

297. Swildens discloses that rather than having each edge server make its own popularity determination separately, in an alternative embodiment, “an auto migration server [(popularity service)] collects all of the log files...from the edge servers [(constructed and adapted to obtain a first set of information from...the edge servers)]...and tabulates the miss count...for every piece of content” (the popularity service making its determinations based...on a measure of popularity of the requested content) to determine whether any content is sufficiently popular based on a threshold such that it should be auto-migrated to the edge (provide the edge servers with a second set of information about which tier of servers should handle those requests for content). Swildens, [0126], [0128], [0130].

- e. **10[d]: wherein the requested content is available as part of a content provider's library, and wherein distinct portions of the content provider's library are logically partitioned across parent servers in the second tier.**

298. In my opinion, Swildens combined with Seed discloses or suggests this limitation.

299. Specifically, as I explained previously, Swildens discloses or suggests that the requested content is available as part of a content provider's library, (§IX.B.1.b (1[a])), and Swildens combined with Seed teaches or suggests storing distinct portions of the content provider's library that are partitioned using a hash

function (*logically*) across parent-level servers (*second tier*). §§IX.B.1.e-IX.B.1.f (1[d][i]-[ii]), IX.B.2 (cl. 2).

9. **Claim 11: A content delivery framework as recited in claim 10, wherein at least one of the parent servers in the second tier comprises a first logically partitioned portion of the content provider's library, the first logically partitioned portion comprising at least the requested content.**

300. In my opinion, Swildens combined with Seed teaches or suggests this limitation.

301. Specifically, as I explained previously, Swildens combined with Seed teaches or suggests storing distinct portions of the content provider's library that are partitioned using a hash function (*logically*), including a first portion with the requested content, on one parent server in the parent-level servers (*second tier*). §§IX.B.1.e-IX.B.1.f (1[d][i]-[ii]), IX.B.2 (cl. 2).

10. **Claim 12: A content delivery framework as recited in claim 11, wherein at least one other parent server in the second tier comprises a second logically partitioned portion of the content provider's library, the second logically partitioned portion being distinct from the first logically partitioned portion.**

302. In my opinion, Swildens combined with Seed teaches or suggests this limitation.

303. Specifically, as I explained previously, Swildens combined with Seed teaches or suggests storing distinct portions of the content provider's library that are partitioned using a hash function (*logically*), including a second portion on another

parent server among the parent-level servers (*second tier*) distinct from the first portion. §§IX.B.1.e-IX.B.1.f (1[d][i]-[ii]), IX.B.2 (cl. 2).

11. Claim 13: A content delivery framework as recited in claim 10, wherein the second set of information further includes identification of a parent server in the second tier that stores a logically partitioned portion of the content provider's library comprising the requested content.

304. In my opinion, Swildens combined with Seed teaches or suggests this limitation.

305. Specifically, as I explained previously, Swildens combined with Seed teaches or suggests incorporating a tier of parent servers into Swildens' CDN network, and using a hash function (*logic*) to partition and then identify which of a plurality of parent servers (*second tier*) is storing a portion of a content provider's collection (*library*). §§IX.B.1.e-IX.B.1.f (1[d][i]-[ii]), IX.B.2 (cl. 2), IX.B.8.e (10[d]). Swildens also discloses using a central server to make collective popularity determinations for a plurality of edge servers. §IX.B.8.d (10[c]).

306. When incorporating a parent tier of servers into Swildens' system consistent with Seed, in my opinion, a POSITA would have appreciated that the central server would send to edge servers information that maps the content identifiers to the parent server in a parent level of servers that would store the portion of the content provider's collection (*library*) that stores the requested content. *See* Swildens, [0065].

- 12. Claim 14: A content delivery framework as recited in claim 10, wherein the second tier of parent servers is any intermediate tier between the first tier and an origin server that stores the content provider's library.**

307. In my opinion, Swildens combined with Seed discloses or suggests this limitation.

308. Specifically, as I explained previously, Swildens combined with Seed teaches or suggests the parent-level servers are intermediate between the edge servers (*first tier*) and an origin server that stores the content provider's library. §§IX.B.1.e (1[d][i]), IX.B.1.g (1[d][iii]).

13. Claim 15

- a. 15[pre]: A method of content delivery in a content delivery network comprising a plurality of tiers of servers, including at least a first tier of servers and a second tier of servers, the method comprising:**

309. In my opinion, if the preamble is limiting, Swildens discloses or suggests it.

310. Specifically, as I explained previously, Swildens discloses a method of content delivery in a CDN comprising edge and origin levels (*a plurality of tiers*), each with a plurality of servers, including edge servers (*first tier*) and origin servers (*second tier*). §IX.B.1.a (1[pre]).

- b. 15[a]: at a first server in one tier of the tiers of servers, obtaining a request from a client for a resource, wherein the resource is available as part of a content provider's library;**

311. In my opinion, Swildens discloses or suggests this limitation.

312. First, as I explained previously, the prior art can be mapped to this claim limitation in two different ways based on dependent claims 17 and 18, because although this limitation recites a “first server” in “one tier of the tiers of servers,” dependent claims 17 and 18 specify the “first server” may be two different types of servers in two different tiers. Specifically, for both dependent claims 17 and 18, the first-tier servers are edge servers and the second-tier servers are parent servers, but dependent claim 17 specifies the first server is an *edge* server, and dependent claim 18 specifies the first server is a *parent* server. In other words, this claim can be read as requiring:

Dependent claim 17	Dependent claim 18
“at [an edge server] in [a first tier of servers]....”	“at [a parent server] in [a second tier of servers]”

Although the prior art can be mapped to this claim limitation in two different ways based on dependent claims 17 and 18 (as well as other limitations, addressed below), I also understand that the prior art needs to only be mapped to the claims one way—not both—for this limitation to be taught by the prior art. To avoid repeating the analysis for dependent claims 17 and 18, however, I address both here.

313. For dependent claim 17, as I explained previously, Swildens discloses an edge server (*first server*) among a group of edge servers (*first tier*) obtaining a request from a client for a file (*resource*) available as part of a content provider's library. §IX.B.1.b (1[a]).

314. For dependent claim 18, as I also explained previously, Swildens combined with Seed teaches a parent server (*first server*) among a group of parent servers (*second tier*) obtaining a request from a client for a file (*resource*) available as part of a content provider's library. Swildens, [0068] (redirecting request to intermediate-origin servers), §IX.B.1.e (1[d][i]) (implementing intermediate-parent servers between edge and origin servers configured like Swildens' origin servers).

- c. **15[b]: if the resource is not available at the first server or at a peer of the first server, then selectively redirecting the request from the client to a second tier server, the second tier server being in a tier distinct from the tier containing the first server, the redirecting being based, at least in part, on a current popularity value for the resource;**

315. If 15[b] must be performed (§VIII (Claim Construction)), in my opinion, Swildens alone or combined with Seed discloses or suggests this limitation.

316. Like 15[a], this limitation can also be read in at least two ways based on dependent claims 17 and 18. Specifically, this claim can be read as requiring:

Dependent claim 17	Dependent claim 18
“if the resource is not available at [the edge server]...then selectively redirecting the request...to a second tier server...distinct from the tier containing [the edge server]”	“if the resource is not available at [the parent server]...then selectively redirecting the request...to a second tier server...distinct from the tier containing [the parent server]”

317. First, as I explained previously, Swildens discloses that if the file (*resource*) is not available at the edge server (*first server*), the request from the client is redirected to an intermediate CDN-origin server (*second-tier server*) distinct from the group of edge servers (*first tier*) of the initial edge server (*first server*). §§IX.B.1.a (1[pre]), IX.B.1.e (1[d][i]).

318. For dependent claim 17, however, Swildens combined with Seed also teaches that if the file (*resource*) is not available at the edge server (*first server*), the request from the client is redirected to an intermediate-parent server (*second-tier server*) distinct from the group of edge servers (*first tier*) of the initial edge server (*first server*). §IX.B.1.e (1[d][i]).

319. As I also explained in Ground 1, however, dependent claim 18 appears to produce a possible contradiction in this limitation, because it requires that “if the resource is not available at a parent server,” which claim 18 specifies is in *the second tier*, “then selectively redirecting the request from the client *to a second tier server*”—in other words, another parent server—“*the second tier server being in a tier distinct from the tier containing the [the parent server in the second tier]*.” In

other words, the claim requires redirecting a request from one second-tier parent server to another second-tier parent server, but also that the two parent servers are in distinct tiers.

320. For purposes of analyzing this claim in view of the prior art, I have assumed one of three possible interpretations can apply.

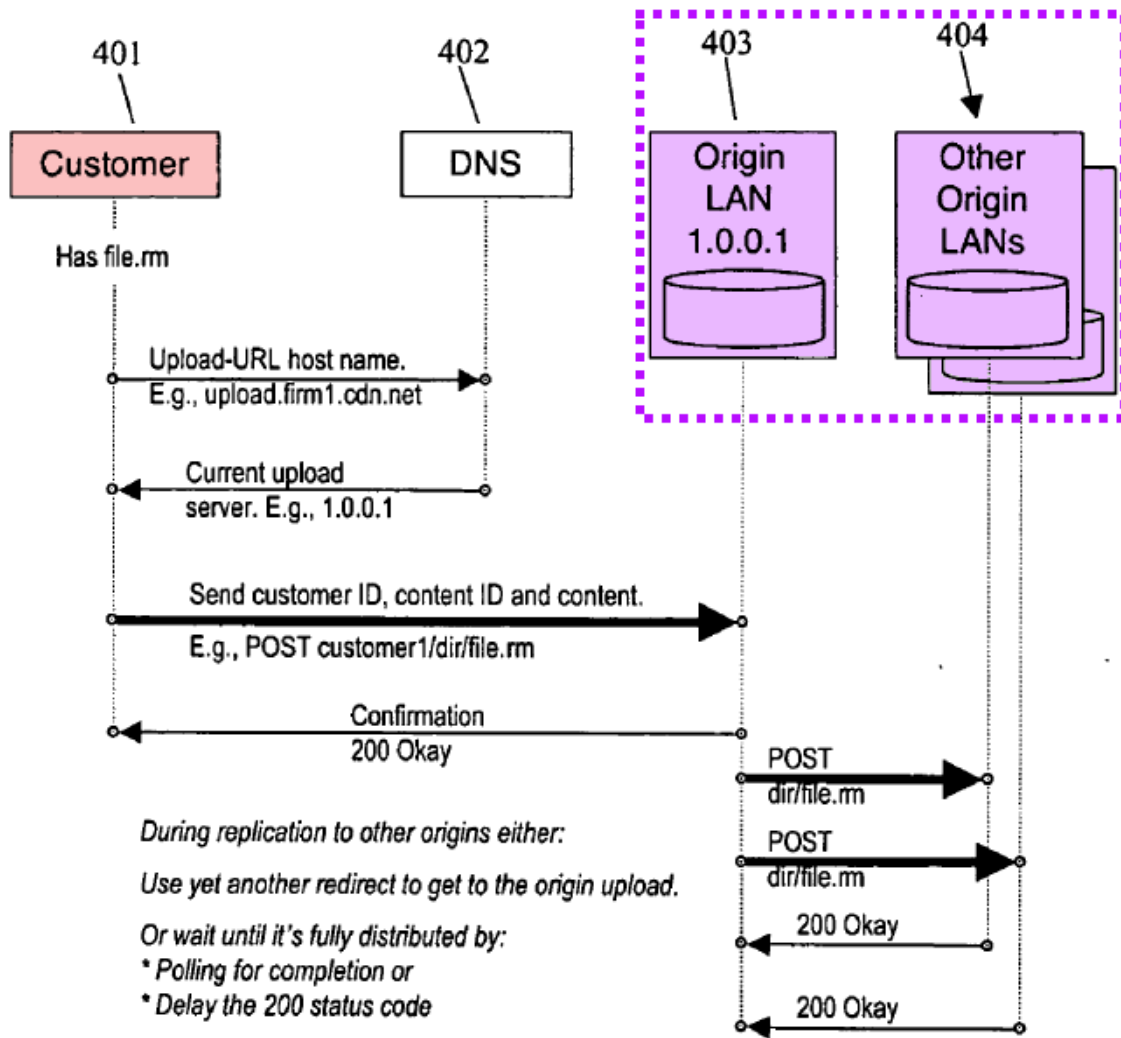
321. **First**, this limitation could be read as “if the resource is not available at [a parent server in the second tier]...then selectively redirecting the request from the client to a second tier server.” In other words, the request is *redirected to a peer-parent server*, but that would be “in a tier distinct from the tier containing the first server” as also required by the claim. Under this interpretation, Swildens discloses that “[a]n individual [CDN-]origin server does not have to be able [to] hold all of the content on its local storage device” and “is able to use to determine which other of the LAN-connected origin servers in its group should hold the content on its local storage device.” Swildens, [0060].

322. The CDN-origin server that a hash function points to, however, may not necessarily have the requested content because Swildens acknowledges that the hash function points only to the server that “*should*” store the requested file, but that is prone to adjustment and error. Swildens, [0046]. When the first intermediate server *should*, but does not have the requested content, in my opinion, a POSITA

would have appreciated a further redirect to another intermediate server that *should* also have the requested file would be required.

323. Moreover, when incorporating an additional or alternative tier of parent servers to perform the functions Swildens assigns to the CDN-origin servers, however (§IX.B.1.e (1[d][i])), Swildens combined with Seed suggests redirecting a request from a parent server (*first server*) among parent servers (*second tier*) to at least another parent server also among the parent servers (*second tier*). *See* Seed, [0029] (redirecting between parents).

324. **Second**, this limitation could be read as “if the resource is not available at [a parent server in the second tier]...then selectively redirecting the request from the client to a second tier server, the second tier server being in a tier distinct from the tier containing the first server.” In other words, the “second tier” of parent servers has **a multi-tier substructure of parent servers**. Under this interpretation, although Swildens depicts a single “level” (*tier*) of CDN-origin servers, Swildens also envisions multiple layers of CDN-origin servers, particularly when using a CDN-origin server as the “staging” server. *See, e.g.*, Swildens, Fig. 4, [0121].



Content upload and distribution to origins

Fig. 4

325. Similarly, Seed envisions providing multiple layers of parent servers. Seed, [0022]. When incorporating an additional tier of parent servers to perform the

functions Swildens assigns to the origin servers (§IX.B.1.e (1[d][i])), in my opinion, a POSITA would have found it obvious to redirect a request from one level of parent servers (*second tier*) to a distinct level of parent servers (a distinct *second tier*), consistent with Seed. As one example, one layer of parent servers closer to the edge may be configured to store objects that are less popular than those at the edge, but more popular than those stored by other lower-tier parent servers further from the edge.

326. **Third**, this limitation could be read as “if the resource is not available at [a parent server in the second tier]...then selectively redirecting the request from the client to a *second server*”—which claim 18 specifies is an “*origin server*”—with “*the second [origin] server* being in a tier distinct from the tier containing the [the parent server in the second tier]”. In other words, this interpretation would require rewriting “second tier server” to “second server.” Under this interpretation, Swildens discloses that “[i]f the origin site 102 does not have the file, then it redirects the user to the staging site 101,” which is a distinct staging-level (*tier*) from the origin servers. Swildens, [0069], §IX.B.1.a (1[pre]). When incorporating an additional or alternative tier of parent servers to perform the functions Swildens assigns to the CDN-origin servers, however (§IX.B.1.e (1[d][i])), a POSITA would have appreciated that “where the parent servers [(*first server in the second tier*)] collectively are not populated with all of the objects and the network has access to

the origin server of a requested object, the client may be redirected (at 225, 320) to the origin server (*second server*) if the requested object is not available on the parent servers,” consistent with Seed. Seed, [0031].

327. For these reasons, in my opinion, Swildens alone or combined with Seed teaches or suggests this limitation under any possible interpretation. Additionally, under any interpretation, a POSITA would have understood that any redirecting would be “based, at least in part, on a current popularity value for the resource” because Swildens discloses that the origin servers (or parent, in view of Seed) redirect requests only *after* the edge server has determined an object is not popular. Swildens, [0046].

- d. **15[c]: wherein the second tier server comprises a first portion of the content provider's library, the first portion comprising at least the resource, wherein at least one other second tier server comprises a second portion of the content provider's library distinct from the first portion, and wherein the second tier of servers is any intermediate tier between the first tier and an origin server that stores the content provider's library, and**

328. If 15[c] must be performed (§VIII (Claim Construction)), in my opinion, Swildens alone or combined with Seed teaches or suggests this limitation.

329. Specifically, both dependent claims 17 and 18, specify the “first tier” is an edge tier of servers and the “second tier” is a parent tier of servers.

330. As I explained previously, Swildens alone or combined with Seed discloses partitioning the content provider's library across an intermediate group of servers (CDN-origin or parent) (*second tier servers*), such that each intermediate server (CDN-origin or parent) stores distinct portions of the content provider's library, including one server with one portion that would have the requested object (*resource*), and a separate server with a distinct portion, with the servers (*second tier*) being intermediate to the edge-level servers (*first tier*) and a staging/origin (customer-CDN) server that stores the content provider's library. §§IX.B.1.f-IX.B.1.g (1[d][ii]-[iii]).

- e. **15[d]: wherein the step of selectively redirecting the request from the client to a second tier server comprises determining to which second tier server the request should be directed, the determining step using a hash function to identify a second tier server that stores the first portion of the content provider's library.**

331. If 15[d] must be performed (§VIII (Claim Construction)), in my opinion, Swildens alone or combined with Seed teaches or suggests this limitation.

332. Specifically, both dependent claims 17 and 18, specify the "first tier" is an edge tier of servers and the "second tier" is a parent tier of servers.

333. As I explained previously, Swildens alone or combined with Seed discloses redirecting a request from the client to an intermediate server (CDN-origin or parent) (*second-tier server*) by determining to which server (CDN-origin or

parent) the request should be redirected using a hash function to identify the server that stores the portion of the content provider's library with the requested file.

§§IX.B.1.e-IX.B.1.f (1[d][i]-[ii], redirecting), IX.B.6 (cl. 7, hash function).

- 14. Claim 17: A method as recited in claim 15 wherein the first tier of servers comprises a plurality of edge servers and the second tier of servers comprises a plurality of parent servers, and wherein the first server is an edge server in the first tier and the second server is a parent server in the second tier.**

334. In my opinion, Swildens combined with Seed discloses or suggests this additional limitation.

335. Specifically, as I explained previously, Swildens combined with Seed teaches or suggests the method of claim 15, in which the first-tier servers are edge servers, the second-tier servers are parent servers, the first server is an edge server, and the second server is a parent server. §§IX.B.13.a-IX.B.13.e (15[pre]-[d]).

- 15. Claim 18: A method as recited in claim 15 wherein the first tier of servers comprises a plurality of edge servers and the second tier of servers comprises a plurality of parent servers, and wherein the plurality of tiers of servers further comprises an origin tier, and wherein the first server is a parent server in the second tier and the second server is a server in the origin tier.**

336. In my opinion, Swildens combined with Seed discloses or suggests this additional limitation.

337. Specifically, as I explained previously, Swildens combined with Seeds teaches or suggests the method of claim 15, in which the first-tier servers are edge

servers, the second-tier servers are parent servers, the first server is a parent server, and the second server is either another parent server (in the same or a distinct level of parent servers) or an origin server. §§IX.B.13.a-IX.B.13.e (15[pre]-[d]).

X. SUMMARY AND ADDITIONAL REMARKS

338. It is my opinion that Grounds 1 and 2 each renders the Challenged Claims unpatentable as obvious, as I explained in the paragraphs preceding.

339. I currently hold the opinions expressed in this declaration based on the information available to me at this time. I may, however, form additional or new opinions should additional information or insights be made available, or upon further consideration.