

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

MICROSOFT CORPORATION,
Petitioner

v.

SANDPIPER CDN, LLC,
Patent Owner.

IPR2026-00180

U.S. Patent No. 10,701,173

**PETITION FOR *INTER PARTES* REVIEW
OF U.S. PATENT NO. 10,701,173**

Mail Stop PATENT BOARD
Patent Trial and Appeal Board
U.S. Patent & Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450

TABLE OF CONTENTS

I.	INTRODUCTION	1
II.	GROUND FOR STANDING.....	1
III.	IDENTIFICATION OF CHALLENGE	1
	A. Prior Art.....	1
	B. Grounds for Challenge	2
IV.	'173 PATENT OVERVIEW	3
	A. Technology Background	3
	B. The '173Pat.	3
	C. File History.....	7
V.	LEVEL OF ORDINARY SKILL	8
VI.	CLAIM CONSTRUCTION	8
VII.	DETAILED EXPLANATION OF GROUNDS.....	8
	A. Ground 1A: Obviousness over O'Rourke.....	8
	1. O'Rourke.....	8
	2. Claims 1, 8	13
	a. 1[pre].....	13
	b. 8[pre].....	14
	c. 1[a]/8[a]	15
	d. 1[b]/8[b].....	18
	e. 1[b][i]/8[b][i]	23
	f. 1[b][ii]/8[b][ii]	29
	g. 1[b][iii]/8[b][iii].....	29
	h. 1[b][iii][A]/8[b][iii][A].....	31
	i. 1[b][iii][B]/8[b][iii][B]	33
	3. Claims 2, 9	35

a.	2[pre].....	35
b.	9[pre].....	36
c.	2[a]/9[a]	36
4.	Claims 4, 11	38
5.	Claims 5, 12	38
6.	Claims 6, 13	39
B.	Ground 1B: Obviousness over O’Rourke and MS-MSSO	40
1.	MS-MSSO.....	40
2.	Motivation to Combine O’Rourke and MS-MSSO	41
3.	Claims 3, 10	43
C.	Ground 1C: Obviousness over O’Rourke and Blumofe	44
1.	Blumofe.....	44
2.	Motivation to Combine O’Rourke and Blumofe	47
3.	Claims 7, 14	51
D.	Ground 2A: Obviousness over Middleton	54
1.	Middleton	54
2.	Claims 1, 8	56
a.	1[pre].....	56
b.	8[pre].....	58
c.	1[a]/8[a]	59
d.	1[b]/8[b].....	64
e.	1[b][i]/8[b][i]	66
f.	1[b][ii]/8[b][ii]	69
g.	1[b][iii]/8[b][iii].....	71
h.	1[b][iii][A]/8[b][iii][A].....	73
i.	1[b][iii][B]/8[b][iii][B]	77
3.	Claim 2, 9.....	80
a.	2[pre].....	80
b.	9[pre].....	80

c.	2[a]/9[a]	80
4.	Claims 4, 11	82
5.	Claim 5, 12	82
6.	Claims 6, 13	84
E.	Ground 2B: Obviousness over Middleton and MS-MSSO.....	84
1.	Motivation to Combine Middleton and MS-MSSO	84
2.	Claims 3, 10	87
F.	Ground 2C: Obviousness over Middleton and Blumofe.....	89
1.	Motivation to Combine Middleton and Blumofe	89
2.	Claims 7, 14	94
VIII.	CONCLUSION.....	96
IX.	MANDATORY NOTICES	96
A.	Real Party in Interest	96
B.	Related Matters.....	96
C.	Notice of Counsel and Service Information.....	96
D.	Power of Attorney	97

EXHIBIT LIST

No.	Exhibit
1001	U.S. Patent No. 10,701,173 (“’173 Patent” or “’173Pat.”)
1002	File History of the ’173 Patent (“’173FH”)
1003	Declaration of Mark Lanning (“Lanning”)
1004	Curriculum Vitae of Mark Lanning
1005	U.S. Patent No. 7,912,921 (“O’Rourke”)
1006	U.S. Patent Publication No. 2011/0060812 (“Middleton”)
1007	U.S. Patent Publication No. 2012/0096106 (“Blumofe”)
1008	[MS-MSSO]: Media Streaming Server System Overview (Microsoft Corporation, 2009), available at https://web.archive.org/web/20100130235412/ http://download.microsoft.com/download/9/5/e/95ef66af-9026-4bb0-a41d-a4f81802d92c/Windows_communication_protocols.zip (“MS-MSSO”)
1009	Reserved
1010	Rajkumar Buyya, Lecture Notes in Electrical Engineering (Volume 9): Content Delivery Networks (Springer-Verlag Berlin Heidelberg, 2008) (“Buyya”)
1011	Abraham Silberschatz, Henry F. Korth, S. Sudarshan, <i>Database System Concepts - Sixth Edition</i> (McGraw-Hill, 2011) (“Silberschatz”)
1012	Kalapatapu Kalyani, <i>Internet Programming and Web Design</i> (Vikas Publishing House Pvt. Ltd., 2013) (“Kalyani”)
1013	John Dilley, et al., <i>Globally Distributed Content Delivery</i> , IEEE Internet Computing, September-October 2002, pp. 50-58 (“Dilley”)

No.	Exhibit
1014	Jianliang Xu, et al., <i>Caching and Prefetching for Web Content Distribution</i> , in Computing in Science & Engineering, IEEE CS and the AIP, July/August 2004, pp. 54-59 (“CiSE”)
1015	U.S. Patent No. 7,024,466 (“Outten”)
1016	Declaration of June Ann Munford (“Munford”)
1017	Akamai White Paper: Turbo-Charging Dynamic Web Sites with Akamai EdgeSuite (Akamai Technologies, Inc., 2001), available at https://www.andrew.cmu.edu/course/15-749/READINGS/optional/Akamai_EdgeSuite_Turbocharging_Websites.pdf (“Akamai EdgeSuite”)
1018	U.S. Patent No. 7,240,100 (“Wein”)
1019	David Gourley & Brian Totty, HTTP The Definitive Guide, O’Reilly (Sept. 2002) (“Gourley”)
1020	U.S. Patent Publication No. 2002/0065899 (“Smith”)
1021	U.S. Patent No. 8,180,720 (“Kovacs”)

LISTING OF CHALLENGED CLAIMS

Reference	Limitation
Claim 1	
1[pre]	A computer-implemented method, operable with a content delivery network (CDN), the method comprising:
1[a]	(A) in response to a request for content, determining, by a node in the CDN, if the content is cached locally at the node;
1[b]	(B) when it is determined by the node that said content is cached locally, then, after determining that the content is cached locally:
1[b][i]	(B)(1) determining, by the node, a current cache policy associated with the content, wherein determining the current cache policy comprises determining a time period for expiration of content in the cache; and then
1[b][ii]	(B)(2) determining, by the node and based on said current cache policy associated with the content, whether it is acceptable to serve the content that is cached locally;
1[b][iii]	(B)(3) based on said determining in (B)(2),
1[b][iii][A]	(i) when it is not acceptable to serve the content that is cached locally, obtaining, by the node, a new version of the content and then serving, by the node, the new version of the content, and
1[b][iii][B]	(ii) when it is acceptable to serve the content that is cached locally, serving, by the node, the content that is cached locally.
Claim 2	
2[pre]	The method as recited in claim 1 further comprising:
2[a]	(C) when it is determined that said content is not cached locally, then (C)(1) obtaining the content and then (C)(2) serving the content obtained in (C)(1).

Reference	Limitation
Claim 3	
3	The method as recited in claim 1 wherein the request comprises an HTTP request and wherein the content comprises one or more resources.
Claim 4	
4	The method as recited in claim 1 wherein the current cache policy is determined from a location distinct from the node.
Claim 5	
5	The method as recited in claim 4 wherein the location is a database.
Claim 6	
6	The method as recited in claim 1 wherein the cache policy relates to caching content within the CDN.
Claim 7	
7	The method as recited in claim 6 wherein the cache policy also relates to caching content outside the CDN.
Claim 8	
8[pre]	A computer program product having computer readable instructions stored on non-transitory computer readable media, the computer readable instructions including instructions for implementing a computer-implemented method, said method operable on a device comprising hardware including memory and at least one processor and running a service on said hardware, said method operable in a content delivery service (CDN), said method comprising:
8[a]	(A) in response to a request for content, determining, by a node of the CDN, if the content is cached locally at the node;

Reference	Limitation
8[b]	(B) when it is determined that said content is cached locally, then, after determining that the content is cached locally:
8[b][i]	(B)(1) determining, by the node, a current cache policy associated with the content, wherein determining the current cache policy comprises determining a time period for expiration of content in the cache; and then
8[b][ii]	(B)(2) determining, by the node and based on said current cache policy associated with the content, whether it is acceptable to serve the content that is cached locally;
8[b][iii]	(B)(3) based on said determining in (B)(2),
8[b][iii][A]	(i) when it is not acceptable to serve the content that is cached locally, obtaining, by the node, a new version of the content and then serving, by the node, the new version of the content, and
8[b][iii][B]	(ii) when it is acceptable to serve the content that is cached locally, serving, by the node, the content that is cached locally.
Claim 9	
9[pre]	The computer program product as recited in claim 8 wherein the method further comprises:
9[a]	(C) when it is determined that said content is not cached locally, then (C)(1) obtaining the content and then (C)(2) serving the content obtained in (C)(1).
Claim 10	
10	The computer program product as recited in claim 8 wherein the request comprises an HTTP request and wherein the content comprises one or more resources.

Reference	Limitation
Claim 11	
11	The computer program product as recited in claim 8 wherein the current cache policy is determined from a location distinct from the node.
Claim 12	
12	The computer program product as recited in claim 11 wherein the location is a database.
Claim 13	
13	The computer program product as recited in claim 8 wherein the cache policy relates to caching content within the CDN.
Claim 14	
14	The computer program product as recited in claim 13 wherein the cache policy also relates to caching content outside the CDN.

I. INTRODUCTION

U.S. Patent No. 10,701,173 (“’173 Patent” or “’173Pat.”; EX-1001) generally relates to content delivery networks (CDNs), which distribute content to clients on behalf of content providers.

The purported invention is directed to a cache server, which caches content (from origin servers) to distribute to clients, and its use of cache policies that determine how the cache server responds to client requests for content. It states that by determining the cache policies when the content is requested, the latest policies are used. But this idea was proposed by others well before the ’173Pat., and the patent inventors improperly claimed the idea as their own. For this reason, Microsoft Corporation (“Microsoft” or “Petitioner”) petitions for *inter partes* review (“IPR”) of claims 1-14 (“Challenged Claims”) of the ’173Pat.

II. GROUNDS FOR STANDING

Petitioner certifies that the ’173Pat. is available for IPR, and Petitioner is not barred or estopped from requesting IPR on the grounds herein.

III. IDENTIFICATION OF CHALLENGE

A. Prior Art

Petitioner respectfully requests cancellation of the Challenged Claims based on:

Prior Art	
O'Rourke (U.S. Patent No. 7,912,921), issued March 22, 2011; prior art under at least 35 U.S.C. §102(a) ¹ (EX-1005).	
Middleton (U.S. Patent Publication No. 2011/0060812), published March 10, 2011; prior art under at least 35 U.S.C. §102(a) (EX-1006).	
Blumofe (U.S. Patent Publication No. 2012/0096106), published April 19, 2012; prior art under at least 35 U.S.C. §102(a) (EX-1007).	
MS-MSSO; publicly available no later than January 30, 2010 ² ; prior art under at least 35 U.S.C. §102(a) (EX-1008).	

B. Grounds for Challenge³

Ground		Claims	Prior Art
1A	§103	1, 2, 4-6, 8, 9, 11-13	O'Rourke
1B		3, 10	O'Rourke, MS-MSSO
1C		7, 14	O'Rourke, Blumofe
2A		1, 2, 4-6, 8, 9 11-13	Middleton
2B		3, 10	Middleton, MS-MSSO
2C		7, 14	Middleton, Blumofe

¹ All references to 35 U.S.C. §§102, 103 are to the AIA statutory framework.

² EX-1016.

³ Petitioner reserves the right to raise §112 arguments in separate proceedings.

IV. '173 PATENT OVERVIEW

A. Technology Background

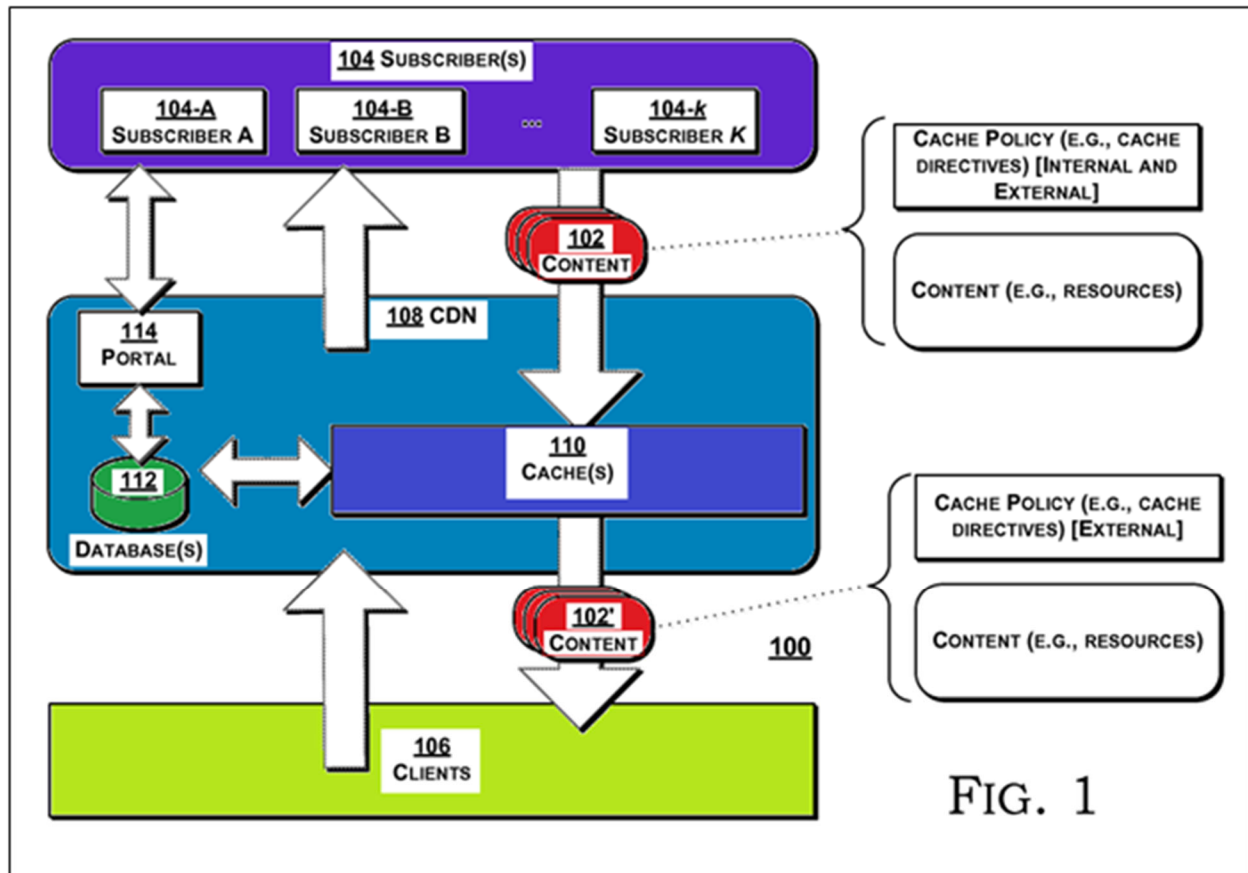
In the Internet's early days, content owners typically maintained their own servers to host content and transmit it to users. Lanning, ¶¶35-47. However, as the Internet grew, this presented problems in terms of scalability, reliability and performance. *Id.* To solve these problems, companies began building CDNs that provided additional layers of geographically distributed servers between the end user and the content owner's server. *Id.*, ¶¶46-56. The CDN's servers would "cache" (or store) copies of content and respond to requests on behalf of the content owner. *Id.*

B. The '173Pat.

The '173Pat. issued 6/30/2020, from Application No. 15/588,935, filed 5/8/2017. The '173Pat. is a continuation of U.S. Patent No. 9,648,127, filed 12/15/2014.

The patent is directed to systems/methods for distributing content using a CDN. EX-1001, 2:7-15. As illustrated in Figure 1, content 102 (red) associated with subscribers 104 (purple) is distributed to clients 106 (light green) via CDN 108 (light blue), which caches content 102 in one or more caches 110 (dark blue). *Id.*, 2:21-25, 3:1-10. Caches 110 are also referred to as "caching nodes"/"nodes." *Id.*, 4:53-61. Caching policies may specify how long cached content remains valid before it is deemed expired (also known as "time-to-live"/"TTL"). *Id.*, 2:3, 3:11-22. Cache(s)

110 can access data storage, such as database 112 (dark green), to retrieve caching policy information associated with content 102. *Id.*, 4:28-37.

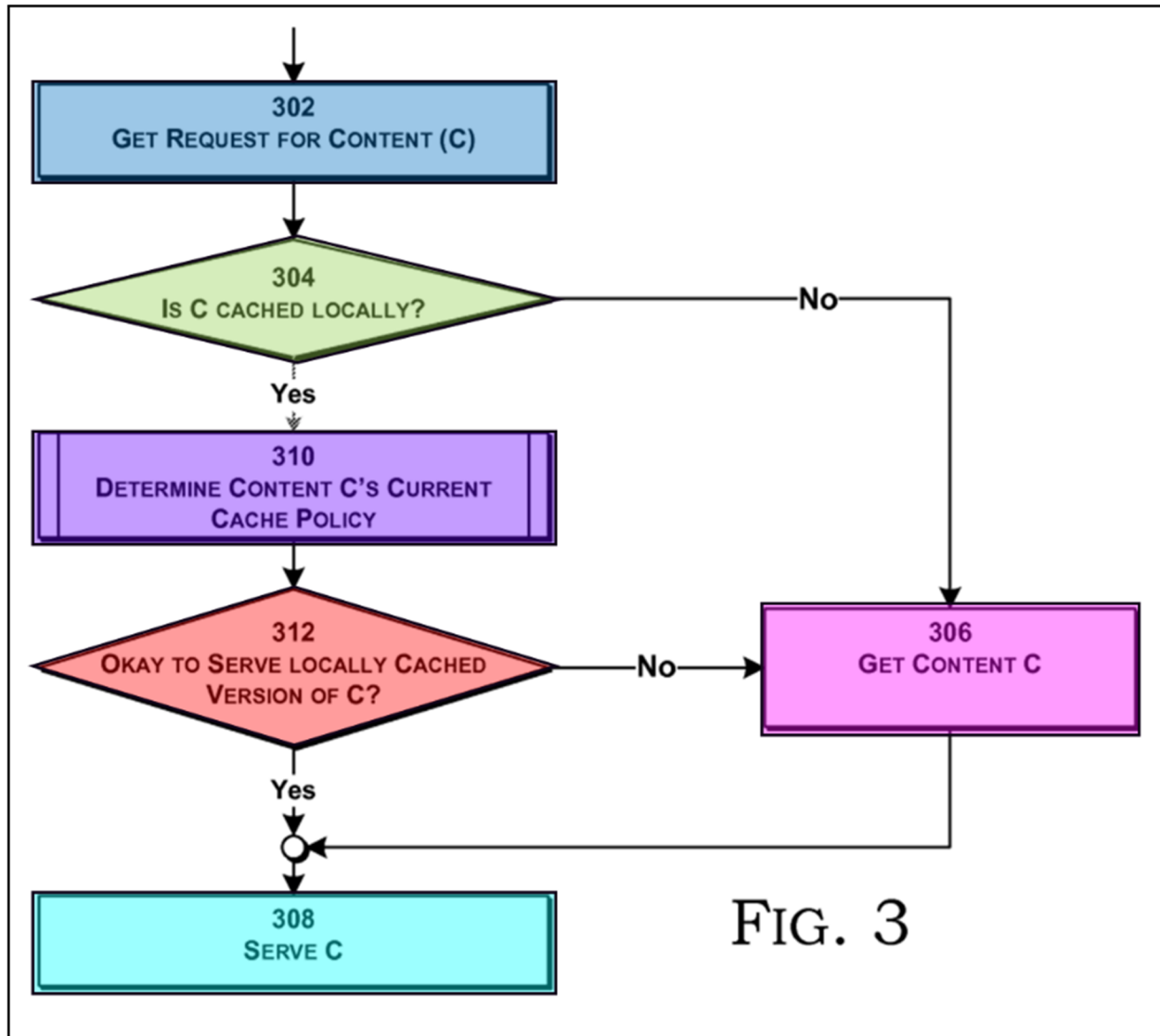


EX-1001, Fig. 1⁴

Figure 3 illustrates that when a cache (also referred to as a “caching node” or “node”) receives a content request 302 (blue), it checks for a locally-cached copy 304 (green). *Id.*, 5:20-30. If there is no local copy, the node retrieves the content from a fill-authoritative node in CDN 306 (pink) and serves the content 308 (turquoise). *Id.*, 5:28-34, Fig. 3. If there is a locally-cached copy of the content, the

⁴ Color annotations added to figures unless otherwise noted.

node obtains the current cache policy associated with the content 310 (purple) and determines whether the content can be served based on the cache policy 312 (coral). *Id.*, 5:39-45. For example, if the cache policy specifies that the content may be cached for one hour, the node compares the fill time of the locally-cached copy to the specified policy (and the current time) to determine whether the cached copy of the content can be served or if it needs to be refreshed or reloaded. *Id.*, 4:62-5:2; *see also id.*, 3:57-61, 5:55-63. If the content is valid based on the determination in 310 (purple), the node serves the locally-cached content in 308 (turquoise); if the content is invalid, the node obtains the content in 306 (pink) and serves in 308 (turquoise) the current version of the content. *Id.*, 5:45-54.



EX-1001, Fig. 3.

The '173Pat. purports to offer an advantage by determining the caching policy at the time content is requested, rather than when it was originally cached, a concept the patent refers to as “late-binding” of the caching policy. *Id.*, 3:62-4:3. However, as discussed below, caches that determined caching policies at request time were known prior to the '173Pat.'s effective date. Lanning, ¶¶64-71.

C. File History

During prosecution, Applicant asserted that the “late-binding policy” reflected the purported point of novelty, arguing the claims were patentable because in the prior art “the actions of [the node] are not dependent on the current cache policy.” ’173FH, 118-20; Lanning, ¶¶72-73.

O’Rourke was presented to the Examiner in multiple information disclosure statements. *See* ’173FH, 53, 304, 45, 272; Lanning, ¶74. Moreover, the Examiner was aware of (and considered) the Written Opinion of the International Searching Authority from the ’173Pat.’s PCT counterpart. *See* ’173FH, 277-284. In that Opinion, the International Searching Authority found all pending claims lacked novelty under PCT Article 33(2) as being anticipated by O’Rourke. *Id.* Despite being made aware of O’Rourke and its importance, the Examiner never applied it.

Middleton is assigned to Level 3 Communications, LLC—the same assignee as the ’173Pat. Middleton, Cover; ’173Pat., Cover. Middleton was filed 7/14/2010 and claimed priority to a provisional filed 9/10/2009. Middleton, Cover. It was published 3/10/2011, years before the ’173 priority application was filed. *Id.* Yet Level 3 Communications never disclosed Middleton to the PTO during the prosecution of the ’173 Patent. *See generally* ’173FH.

V. LEVEL OF ORDINARY SKILL

A person of ordinary skill in the art (“POSITA”) at the time of the invention 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. Work experience can substitute for formal education and vice versa. Lanning, ¶¶93-94.

VI. CLAIM CONSTRUCTION

Petitioner is unaware of any prior claim construction determination for the ’173Pat. No express constructions are necessary in this IPR. Petitioner has applied the claim terms’ ordinary meanings as understood by POSITAs. 37 C.F.R. §42.100(b).

VII. DETAILED EXPLANATION OF GROUNDS

A. Ground 1A: Obviousness over O’Rourke

1. O’Rourke

O’Rourke teaches a “cache server [that] includes a media serving engine that is capable of distributing media content” across a network. O’Rourke, Abstract. O’Rourke recognized that in media distribution systems of the time, cache servers had “substantially fixed architecture that [were] difficult to customize.” *Id.*, 1:19-21, 2:8-9. These cache servers would “perform various cache-related functions that are controlled by a set of cache policies that are determined by the manufacturer. These

existing cache servers ... minimize[d] the opportunities for a customer to modify the cache server to meet their specific needs.” *Id.*, 2:19-26.

To address this, O’Rourke teaches “a cache server that performs a variety of cache-related functions. The operation of the cache server is determined by a set of cache policies that are determined by the user of the cache server.” *Id.*, 3:26-29. O’Rourke explains that this “flexible architecture allows a common cache server to be configured in a variety of different ways depending on the set of policies applied to the cache server.” *Id.*, 2:32-35.

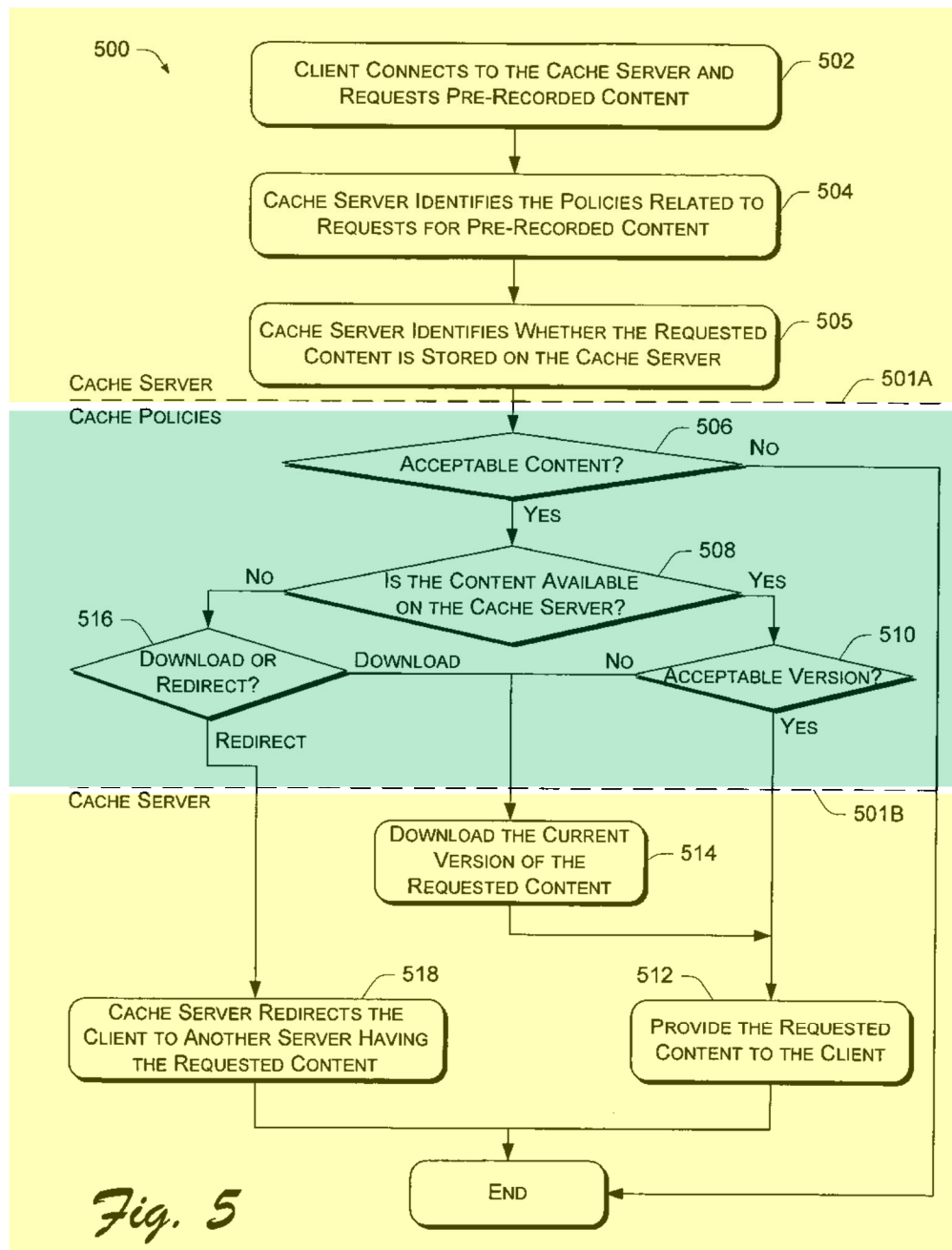
O’Rourke’s cache server includes a cache engine, “which performs the various functions necessary to cache media content. Example caching functions include downloading media content from an origin server, distributing media content to one or more clients, redirecting clients to a different server for particular media content, and deleting cached media content to release storage space for caching different media content.” *Id.*, 4:43-50, Fig. 4.

The cache server additionally includes a set of cache policies. *Id.*, 4:50-52, Fig. 4. The cache policies, which are typically defined based on an administrator’s knowledge of the origin servers, cache servers, clients, networks, and other network components, are used by the cache server’s cache engine in performing various caching functions. *Id.*, 4:50-52, 7:41-44. For example, the cache policies may define

whether the cache engine should download and cache content, whether the cache engine should delete cached content, etc. *Id.*, 4:52-56.

The cache policies (also referred to as policy plug-ins) can be added or removed at any time, “thereby providing flexible management of the cache policies. Thus, the operation of a cache server can be modified simply by changing the cache plug-in modules in the cache server.” *Id.*, 3:42-45, 5:2-8. In other words, the cache policies that dictate how the cache engine operates may change during operation of the cache server. Lanning, ¶¶76-78.

When the cache server receives a request for content, the functions performed by the cache server (e.g., the cache engine) are based on decisions made based on the cache policies. O’Rourke, 6:1-5; Lanning, ¶79. Figure 5 illustrates the functions performed by the cache server (the sections above line 501A and below line 501B, yellow), and the decisions made based on the cache policies (the section between lines 501A and 501B, green). O’Rourke, 6:5-10. As described in detail herein, the cache server’s actions (e.g., whether to serve cached content) depend on the changing cache policies. Lanning, ¶79.



O'Rourke, Fig. 5.

O'Rourke describes that its "procedure 500 for handling a client request to receive pre-recorded content" consists of functions performed by the cache server (the sections above line 501A and below line 501B, yellow) and the decisions made

based on the cache policies (the section between lines 501A and 501B, green). O'Rourke, 6:1-10. The first section of functions performed by the cache server (i.e., the section above line 501A) are those that identify information used in making the cache policy decisions (e.g., identifying the policies related to requests for pre-recorded content, identifying whether the requested content is stored on the cache server), and the second section of functions performed by the cache server (i.e., the section below line 501B) are those that perform operations based on the cache policy decisions (e.g., download the requested content, provided the requested content, etc.). Lanning, ¶¶79-80. Further, it would have been obvious to perform the information identifying functions (i.e., above line 501A) in a different order without altering the functionality of O'Rourke's procedure 500. *Id.* For example, it would have been obvious to, in response to a request for pre-recorded content, identify whether the requested content is stored on the cache server (505) and then identify the policies related to requests for pre-recorded content (504), because neither function depends on the other function. *Id.* O'Rourke suggests the same, explaining that "[a]lthough the description above uses language that is specific to structural features and/or methodological acts...the specific features and acts are disclosed as exemplary forms of implementing the invention." O'Rourke, 10:40-45. More generally, identifying whether content was in a cache (in response to a request for the content) before identifying cache policies associated with the request or content was known

to POSITAs. Lanning, ¶¶79-80; *see also* Wein, 20:26-21:8, Fig. 5; Akamai EdgeSuite, 6. In other words, it would have been obvious, in response to a request for content (502), to identify whether the requested content is stored in the cache (505) and then identify policies related to requests for pre-recorded content (504). Lanning, ¶¶79-80.

2. Claims 1, 8

a. 1[pre]

If limiting, O’Rourke discloses/suggests 1[pre]. Lanning, ¶¶98-100.

O’Rourke describes “*methods...that perform[] a variety of cache-related functions.*”⁵ O’Rourke, 3:26-27. A cache server, such as a Windows Media server, may perform the methods. *Id.*, 4:57-60, 6:1-5 (“The procedure 500 illustrates the functions performed by the cache server”). O’Rourke’s cache-related functions are thus *computer-implemented methods*. Lanning, ¶99.

O’Rourke’s cache-related functions include the cache server “distributing media content to one or more clients” over a communication network, such as a high-speed LAN. O’Rourke, 3:36-38, 4:45-50. O’Rourke’s cache-related functions are thus *operable with a content delivery network (CDN)*. Lanning, ¶100.

⁵ Italicized text is claim text unless otherwise noted.

b. 8[pre]

If limiting, O’Rourke discloses/suggests 8[pre]. Lanning, ¶¶101-103.

O’Rourke describes “a computer 642 that can be used in accordance with the invention” and can perform the functions of a cache server. O’Rourke, 8:45-49. The computer includes “one or more processors or processing units 644, a system memory 646,” and drives (e.g., hard disk drive 656). *Id.*, 8:49-52, 8:62-9:4. O’Rourke explains that “[t]he drives and their associated computer-readable media provide nonvolatile storage of computer readable instructions, data structures, program modules and other data for computer 642.” *Id.*, 9:4-7. The computer readable instructions and program modules on hard disk 656 are *[a] computer program product having computer readable instructions stored on non-transitory computer readable media, the computer readable instructions including instructions for implementing a computer-implemented method*. Lanning, ¶102.

Further, performing the program modules on computer 642, which may be a cache server, illustrates *said method operable on a device comprising hardware including memory and at least one processor and running a service on said hardware, said method operable in a content delivery service (CDN)*. Lanning, ¶103. For example, computer 642 may perform O’Rourke’s procedure 500 (*see* §VII.A.2.a). O’Rourke, 6:2-5 (“The procedure 500 illustrates the functions performed by the cache server”); Lanning, ¶103.

c. 1[a]/8[a]

O'Rourke discloses/suggests 1[a]/8[a]. Lanning, ¶¶104-108.

O'Rourke discloses “a procedure 500 for handling a client request to receive pre-recorded content,” performed by a cache server based on cache policies. O'Rourke, 6:1-5. The cache server “is able to redistribute...media content [from an origin server] to multiple local clients.” *Id.*, 3:46-55. The cache server (*node*) is thus in/of a *CDN*, and procedure 500 is performed *by a node [in]/[of] the CDN*. Lanning, ¶105.

Procedure 500 is “for handling a client request to receive pre-recorded content.” O'Rourke, 6:1-2. Figure 5 illustrates that the first block of procedure 500 is block 502 (red), in which the client connects to the cache server to request the pre-recorded content. *Id.*, 6:1-12. The blocks following block 502 are thus *in response to a request for content*. Lanning, ¶106.

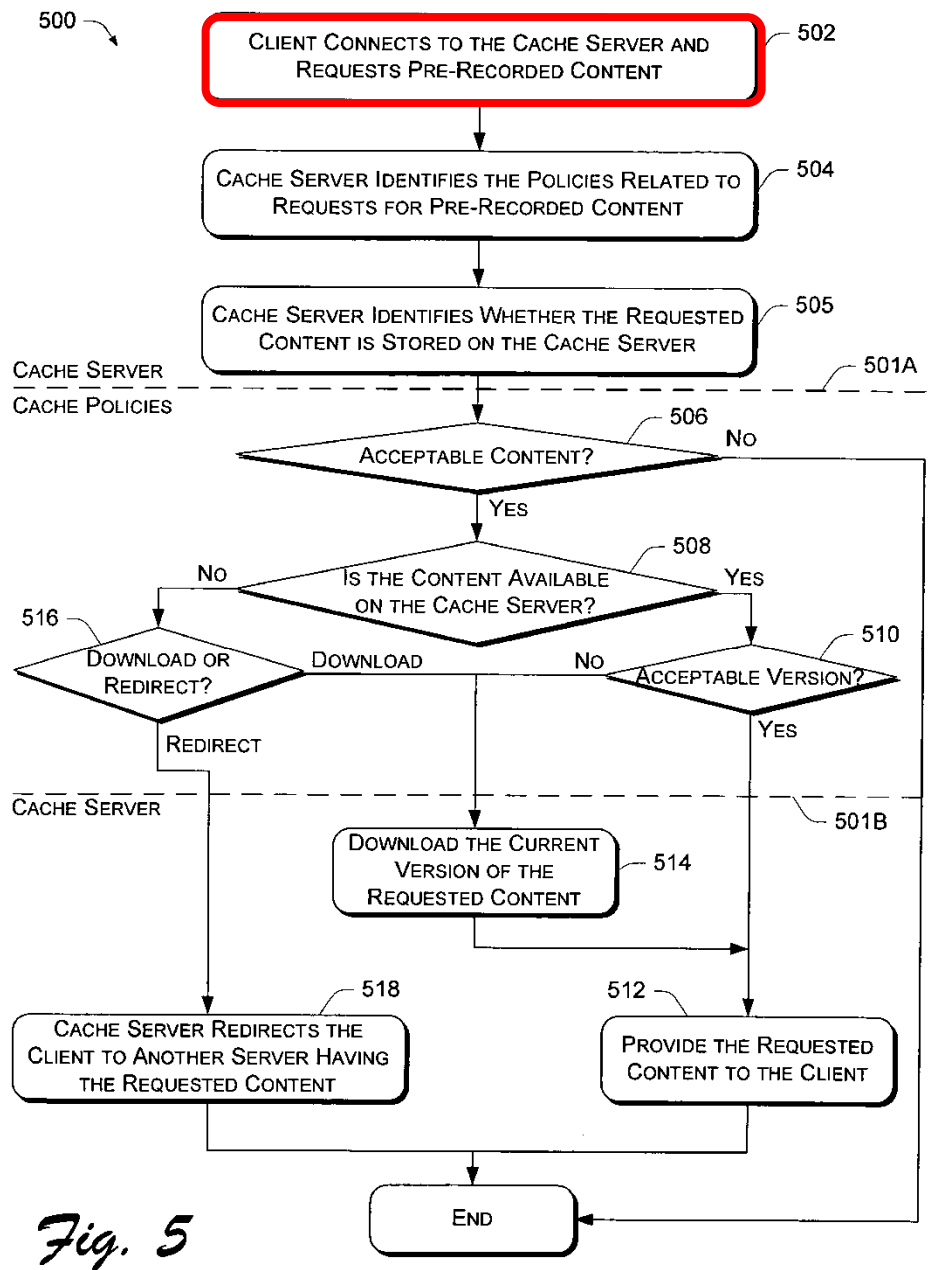


Fig. 5

O'Rourke, Fig. 5.

During procedure 500, “[t]he cache server then identifies whether the requested content is stored on the cache server” (*determining...if the content is cached locally at the node*). O'Rourke, 6:14-15; Lanning, ¶107. The determining of

whether the requested content is stored on the cache server is performed at block 505 (blue), after block 502 (red), and is thus *in response to* the client request for content. O'Rourke, 6:10-12, 6:14-15; Lanning, ¶107.

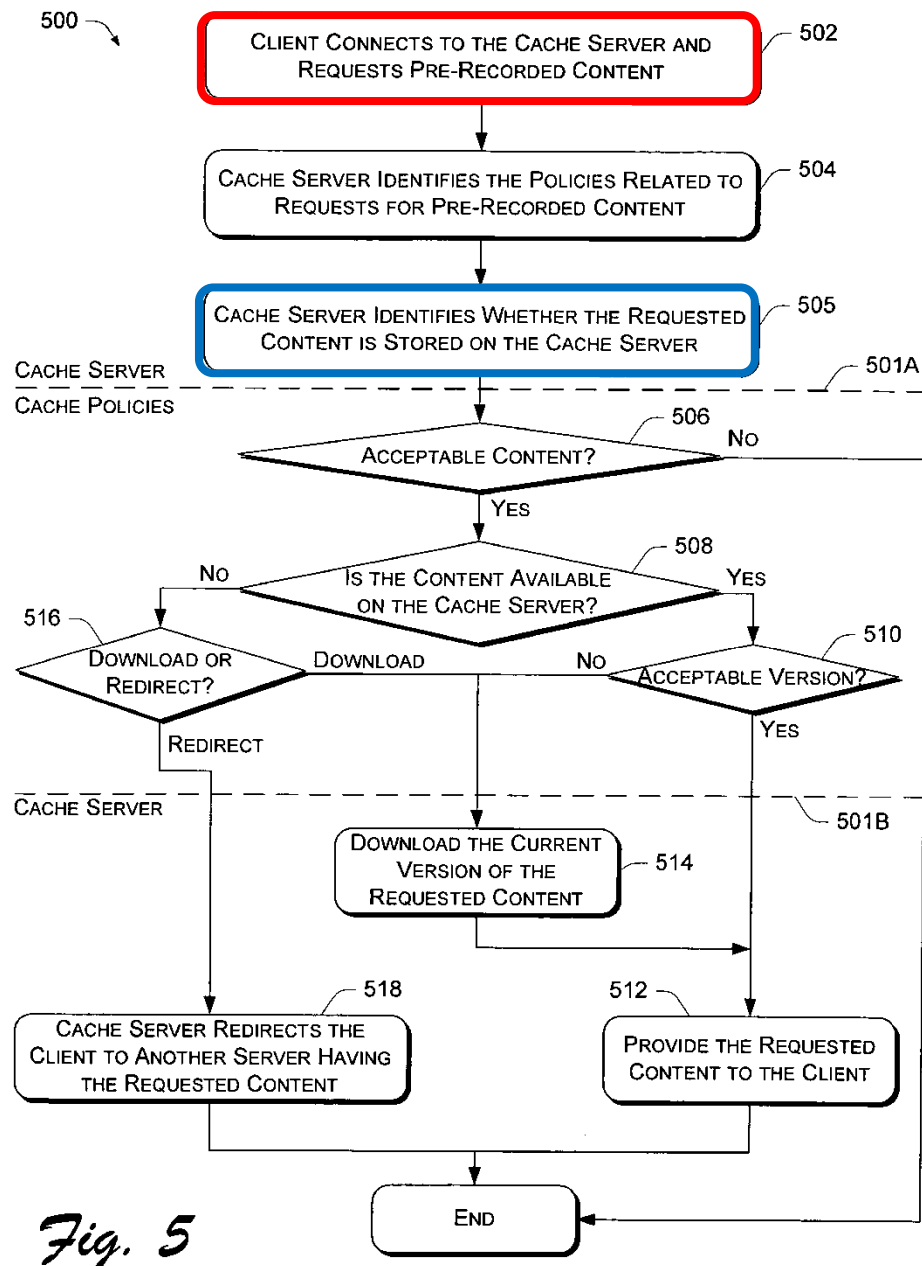


Fig. 5

O'Rourke, Fig. 5.

Additionally, as described in §VII.A.1, it would have been obvious in response to a request for pre-recorded content (502) for procedure 500 to identify whether the requested content is stored on the cache server (505) and then identify cache policies related to pre-recorded content (504). Lanning, ¶108. In other words, it would have been obvious after the cache server receives the request for pre-recorded content (502) (*in response to a request for content*) for the cache server to then identify whether the requested content is stored on the cache server (505) (*determining, by a node [in]/[of] the CDN, if the content is cached locally at the node*). *Id.* O’Rourke discloses/suggests 1[a]/8[a] for this additional reason. *Id.*

d. 1[b]/8[b]

O’Rourke discloses/suggests 1[b]/8[b]. Lanning, ¶¶109-113.

As described in §VII.A.2.cVII.A.2.c, O’Rourke describes “identif[ying] whether the requested content is stored on the cache server (505).” O’Rourke, 6:14-19. “For example, the cache server may examine a URL associated with the requested content to determine whether the content is available on the cache server.” *Id.* O’Rourke additionally describes how “the cache policies determine whether the requested content is available on the cache server (508).” *Id.*, 6:28-31. As described in detail herein, O’Rourke further describes different functions procedure 500 performs depending on the determination at block 508 (i.e., if the content is available on the cache server, determining whether it is an acceptable version; if the content

is not available on the cache server, determining whether to download it). *Id.*, 6:31-34, 6:55-58; Lanning, ¶110. In other words, O’Rourke describes identifying whether requested content is available on the cache server (505), and then responding to the client request differently depending on whether the content is available on the cache server (508). Lanning, ¶110. O’Rourke thus “determine[s] whether the cached version of the requested content is an acceptable version (510)” *when it is determined [...] that said content is cached locally. Id.*

O’Rourke additionally describes that “[i]n one embodiment, the cache server is a Windows Media Server and the set of cache policies are contained in one or more cache plug-in modules that are loaded in the Windows Media Server and called by the Windows Media Server...Alternatively, the cache plug-in modules can run as part of another process on the same computer system.” O’Rourke, 4:57-5:1.

Similarly, Figure 4 illustrates that the “[c]ache server 300 includes a cache engine 302 which performs the various functions necessary to cache media content,” and a “set of cache policies 304 are accessed and utilized by the cache engine 302 in performing its various caching functions.” *Id.*, 4:42-52. Because the cache policies may be loaded in and called by the cache server, the cache server (*node*) makes decisions based on the cache policies (further described herein). Lanning, ¶111. Thus, when “the cache policies determine whether the requested content is available on the

cache server,” that determination is made *by the node*. O’Rourke, 6:28-31; Lanning,
¶111.

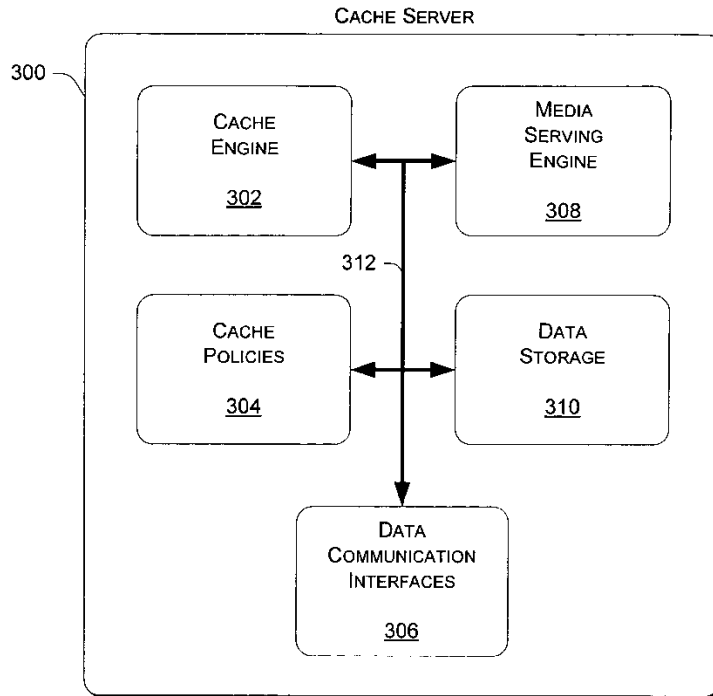
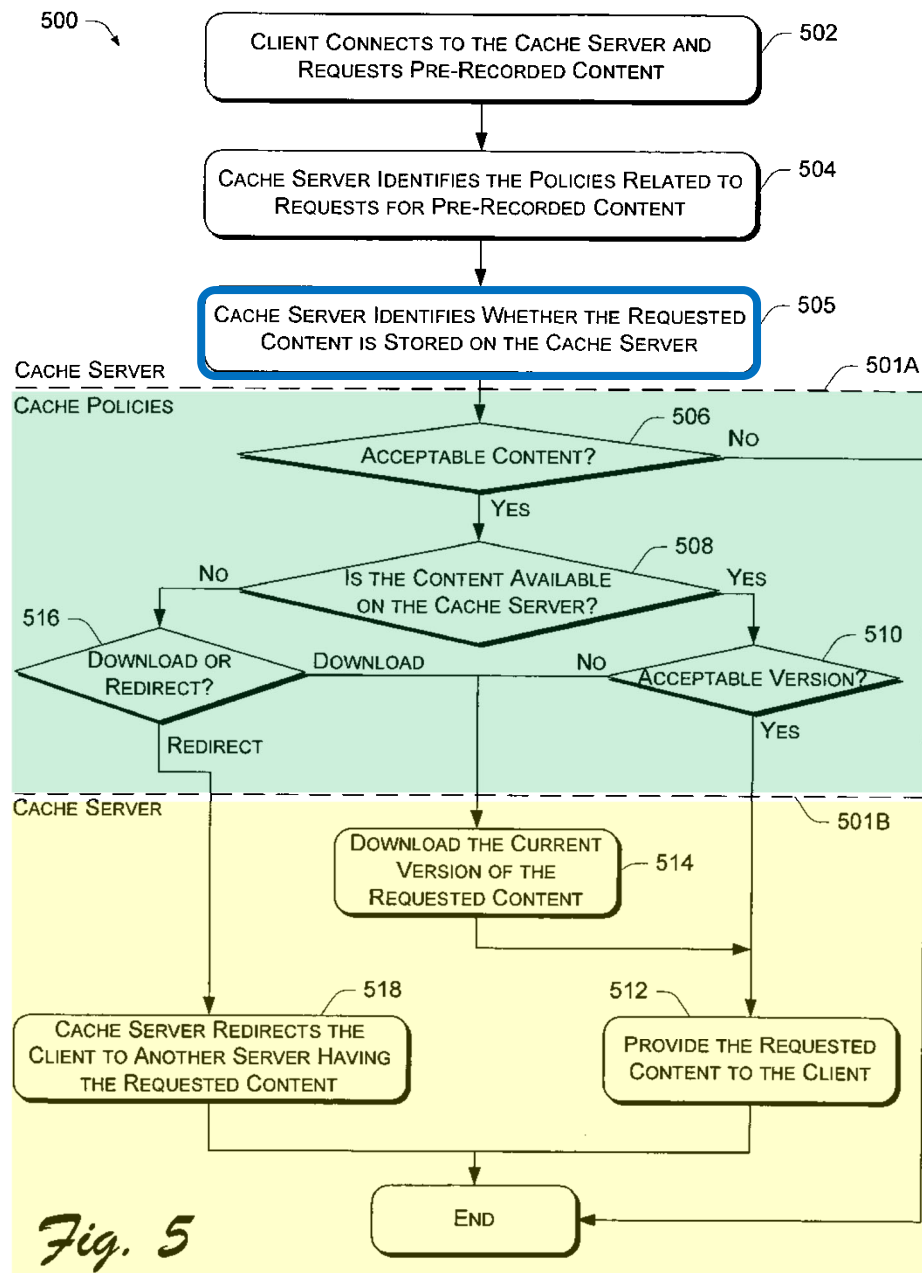


Fig. 4

O’Rourke, Fig. 4

Procedure 500 includes “functions performed by the cache server as well as the decisions made by the cache policies.” O’Rourke, 6:2-5. In Figure 5, the section below line 501B (yellow) represents functions performed by the cache server, and the section between lines 501A and 501B (green) represents decisions made by the cache policies (which, as described above, may also be performed by the cache

server). *Id.*, 6:5-10; Lanning, ¶112. Both highlighted sections of procedure 500 are performed after the cache server (*node*) “identifies whether the requested content is stored on the cache server” at block 505 (blue). O’Rourke, 6:14-15. Thus, the blocks following block 505 are performed *after determining that the content is cached locally*. Lanning, ¶112.



O'Rourke, Fig. 5.

Additionally, as described in §VII.A.1, it would have been obvious in response to a request for pre-recorded content (502) for procedure 500 to identify whether the requested content is stored on the cache server (505) and then identify

cache policies related to pre-recorded content (504). Lanning, ¶¶79-80, 113. Thus, in addition to the blocks below line 501A being performed *after determining that the content is cached locally* (as described above), it would have been obvious (as described further below) for the identification of cache policies related to pre-recorded content (504) to have also been performed *after determining that the content is cached locally* at block 505. *Id.*

e. 1[b][i]/8[b][i]

O'Rourke discloses/suggests 1[b][i]/8[b][i]. Lanning, ¶¶114-121.

O'Rourke teaches that the cache server may make cache policy decisions (*node*). §VII.A.2.d; Lanning, ¶115. Thus, cache policy decisions are *determin[ed]*, *by the node*. Lanning, ¶115.

After determining that the requested content is stored on the cache server, “the cache policies determine whether the cached version of the requested content is an acceptable version.” O'Rourke, 6:31-34; Lanning, ¶116. O'Rourke additionally describes that the cache policies may be “based on [an] administrator's knowledge and/or expectations with respect to, for example, the type of content requests received by the cache server.” O'Rourke, 7:48-51. For example, “[t]hese policies may also determine whether the cache server is required to have the latest version of the content or whether a ‘stale’ version of the content is sufficient in particular situations.” *Id.*, 7:64-67. Thus, the cache policies, such as the *cache policy* that

determines whether the cached version of the requested content is an acceptable version, are *associated with the content*. Lanning, ¶116.

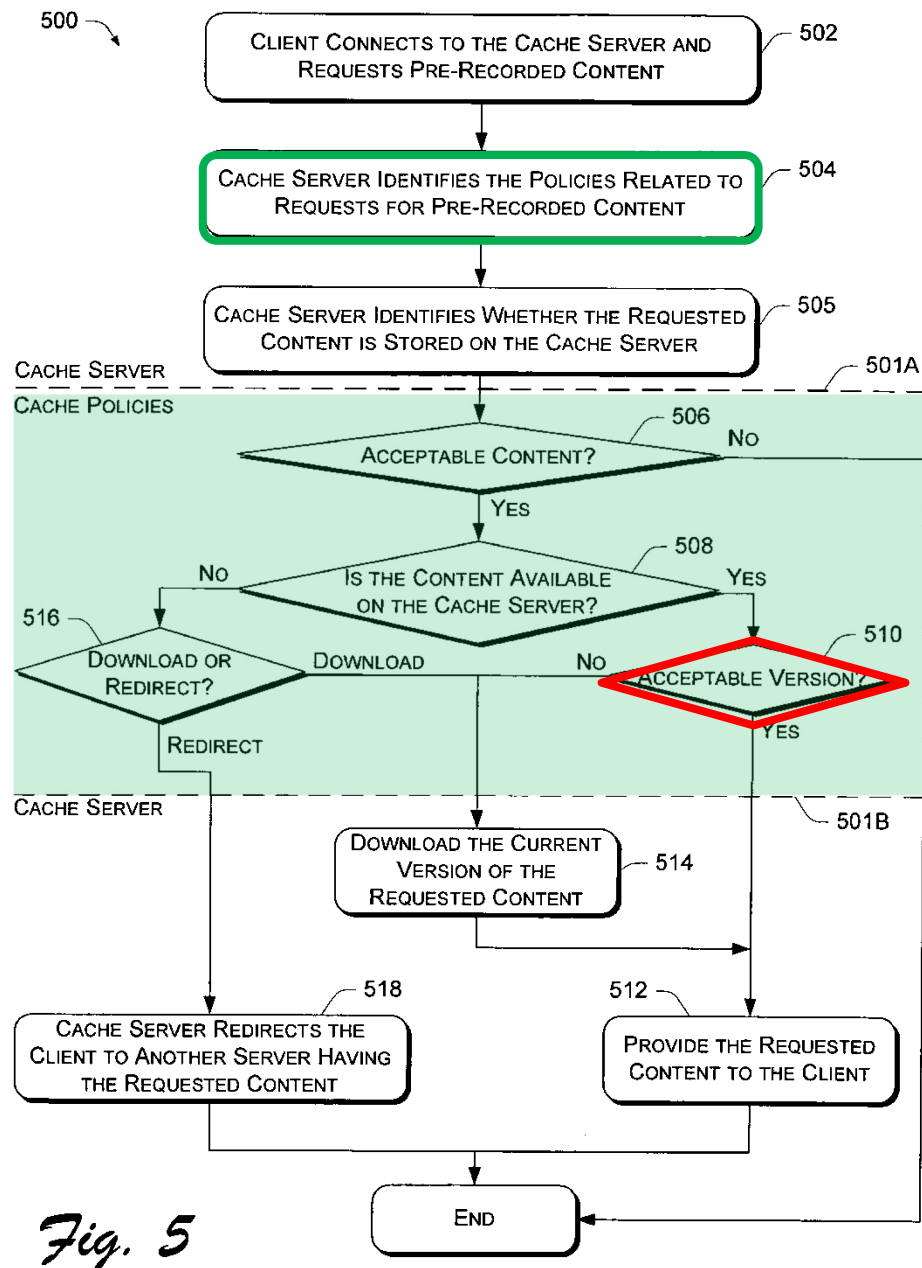
The cache policy can determine whether the content stored on the cache server is an “acceptable version” based on “whether the cached version is the same as the origin server’s version or is within thirty minutes of the origin server’s version.” O’Rourke, 6:34-37. Similarly, the cache policies may determine “whether a ‘stale’ version of the content is sufficient in particular situations.” *Id.*, 7:64-67. In other words, per the cache policy, stored content may not be an “acceptable version” if it is too stale (e.g., stored on the cache server for more than thirty minutes). Lanning, ¶117. And to determine whether content is too stale, the cache policy must be able to determine when the change from acceptably stale (e.g., within thirty minutes) to too stale (e.g., more than thirty minutes) occurs. *Id.* It would have been obvious to a POSITA that this change, from acceptably stale to too stale, is a *time period for expiration* (e.g., thirty minutes after the content was stored on the cache server). Thus, the *cache policy*, for determining whether the content stored on the cache server is an acceptable version, *comprises determining a time period for expiration of content in the cache*. *Id.*

For context, the ’173Pat. specification describes systems and methods for accomplishing the claimed step of “*determining...a current cache policy*.” Specifically, the ’173Pat. describes a system that “allows for specification, e.g.,

within a CDN node, of a cache expiration policy of content the CDN caches on behalf of its subscribers.” EX-1001, 3:42-44. “[A] CDN subscriber...use[s] a portal to set a caching policy for a particular kind of their content (e.g., for their PDF files or the like), e.g., so that those files are cached for a particular time period (e.g., a week, an hour, etc.).” *Id.*, 3:57-61; *see also id.*, 4:38-41 (CDN supports users “setting or overriding an internal caching policy for resources, e.g., via a self-service portal or the like (e.g., via an API).”). “[T]he CDN’s internal request processing process may determine the caching policy of requested content at request time rather than fill time,” a concept the specification refers to as “late-binding.” *Id.*, 3:64-67; *see also id.*, 4:30-34. This “late-binding” “assure[s] that the latest internal caching policy is used.” *Id.*, 3:64-4:3. The policy retrieved at request time is referred to as “the current cache policy.” *Id.*, 5:39-42; Lanning, ¶118. The purported “benefit” of such a system is “having relatively instant policy updates,” although the “downside of this approach is that the caching policy determination must be made for every request.” Ex-1001, 6:4-9. Thus, the ’173Pat. describes a system that allows subscribers to set or update cache policies while the CDN is in use: in such a system, the claimed step of “*determining...a current cache policy*” can be accomplished by determining the caching policy at request time to ensure that the latest policy is used. Lanning, ¶118.

O’Rourke operates similarly. O’Rourke describes how “the operation of [the] cache server can be modified simply by changing” the cache policies. O’Rourke,

5:1-8; Lanning, ¶119. That is, the cache policies (such as the *cache policy* for determining whether stored content is acceptable) may change during operation of the cache server. Lanning, ¶119. The operative policy at any point in time is “the currently active cache policy.” O’Rourke, 12:15-19 (claim 9); Lanning, ¶119. Moreover, after the cache server receives the request for content, it “identifies the policies related to requests for pre-recorded content (504).” O’Rourke, 6:12-14. In Figure 5, the identified cache policies (504, green) are then used to make the cache policy decisions (the section between lines 501A and 501B, shaded green), including the decision of whether the stored content is an acceptable version (510, red). Lanning, ¶119. In other words, O’Rourke teaches determining the cache policy of requested content at request time rather than fill time, i.e., the so-called “late-binding” of the ’173Pat. *Id.* The cache policies (e.g., identified at block 504) used to make the cache policy decisions (e.g., a determination of whether stored content is an acceptable version at block 510) are thus each *a current cache policy*, and determining the cache policy in response to a request is *determining...a current cache policy. Id.*



O'Rourke, Fig. 5.

O'Rourke's "procedure 500 for handling a client request to receive pre-recorded content" consists of functions performed by the cache server (i.e., the sections above line 501A and below line 501B) and the decisions made based on the

cache policies (i.e., the section between lines 501A and 501B). O’Rourke, 6:1-10. Before cache policy decisions are made, “the cache server identifies the policies related to requests for pre-recorded content (504)” and “identifies whether the requested content is stored on the cache server (505).” *Id.*, 6:12-15. Then, cache policy decisions are made based on cache policies of the content, such as whether the requested content is acceptable (506) and whether a cached version of the content is an acceptable version (510). *Id.*, 6:20-22, 6:31-34. In other words, procedure 500 transitions at line 501A from a prior identification of policies related to requests for pre-recorded content (504) to application of cache policies for the requested content (e.g., block 510). Lanning, ¶120. Thus, after line 501A (which occurs *after determining that the content is cached locally* as recited in 1[b]/8[b]) O’Rourke *determin[es]...a current cache policy* (to determine whether a cached version of the content is an acceptable version at block 510). *Id.*

If PO argues that cache policy decisions performed after line 501A do not disclose/suggest *determining...a current cache policy* that is performed *after determining that the content is cached locally* (as recited in 1[b]/8[b]), O’Rourke’s identification of cache policies (504) does. *Id.*, ¶121. As described in §VII.A.1, it would have been obvious for O’Rourke’s procedure 500 to identify whether the requested content is stored on the cache server (505) and then identify cache policies related to pre-recorded content (504). *Id.*, ¶¶79-80, 121. Thus, identifying the

policies related to requests for pre-recorded content (504) (*determining...a current cache policy*) is performed *after determining that the content is cached locally* as recited in 1[b]/8[b]. *Id.*, ¶121.

f. 1[b][ii]/8[b][ii]

O'Rourke discloses/suggests 1[b][ii]/8[b][ii]. Lanning, ¶¶122-123.

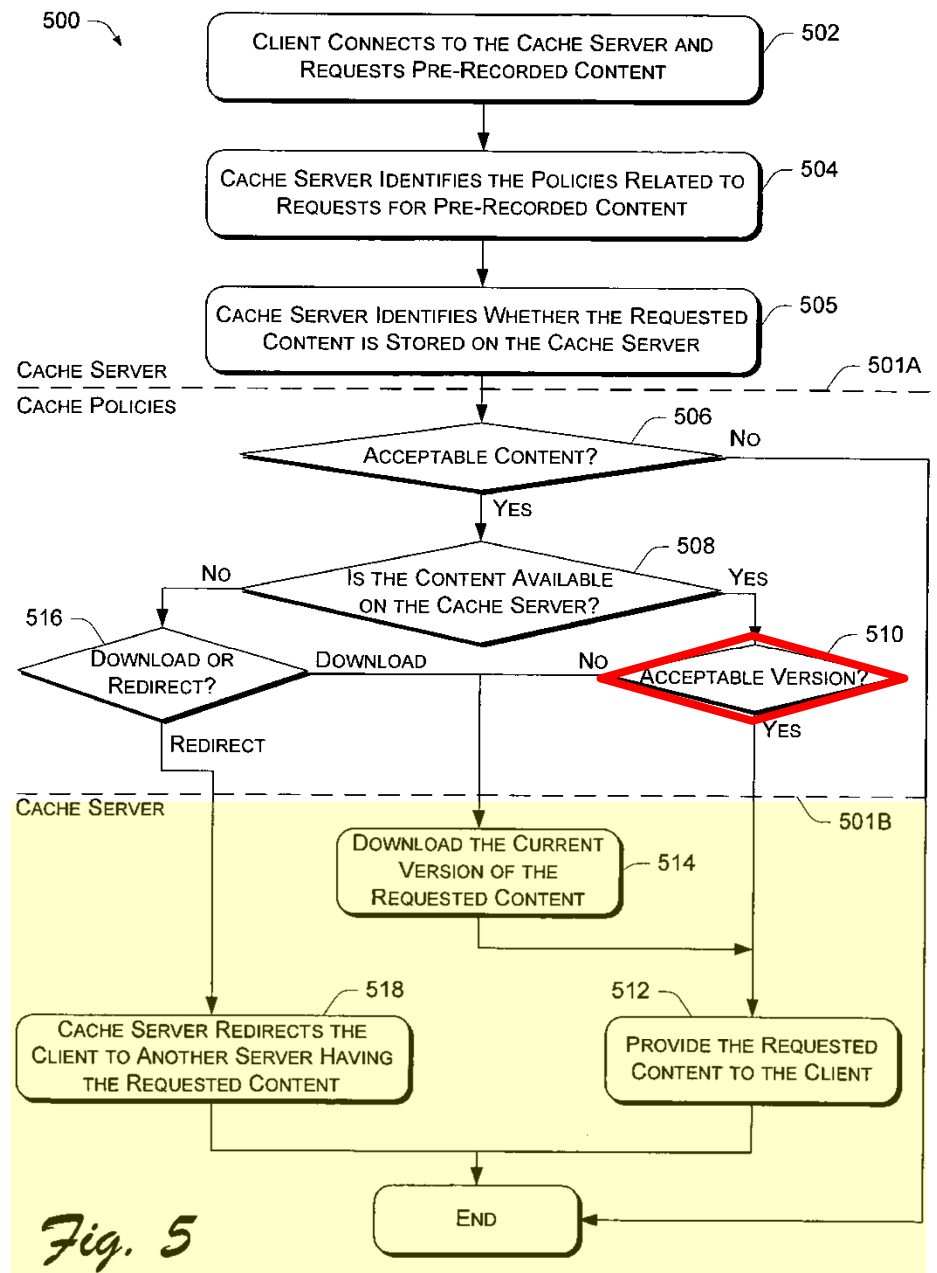
“If the requested content is available, then the cache policies determine whether the cached version of the requested content is an acceptable version.” O'Rourke, 6:31-34. And as described herein, if the cached version of the content is acceptable, it is provided to the client. *Id.*, 6:37-40. Thus, the cache policies determine *whether it is acceptable to serve the content that is cached locally*. Lanning, ¶123. And as described in §§VII.A.2.d-VII.A.2.e, each of the cache policies (including the cache policy that determines whether the content stored on the cache server is acceptable) is a *current cache policy associated with the content* and its decisions are *determin[ed], by the node*. *Id.*

g. 1[b][iii]/8[b][iii]

O'Rourke discloses/suggests 1[b][iii]/8[b][iii]. Lanning, ¶¶124-126.

(B)(2), as recited in 1[b][iii]/8[b][iii], refers to 1[b][ii]/8[b][ii]. Lanning, ¶125. O'Rourke's cache policy determination of whether content stored on the cache server is an acceptable version discloses 1[b][ii]/8[b][ii]. §VII.A.2.f; Lanning, ¶125.

In O'Rourke, the determination of whether stored content is an acceptable version is performed at block 510 (red), based on whether procedure 500 proceeds to blocks within the section below line 501B (yellow). O'Rourke, 6:5-10; Lanning, ¶126. Thus, the blocks of procedure 500 performed after block 510 are *based on said determining in (B)(2). Id.*



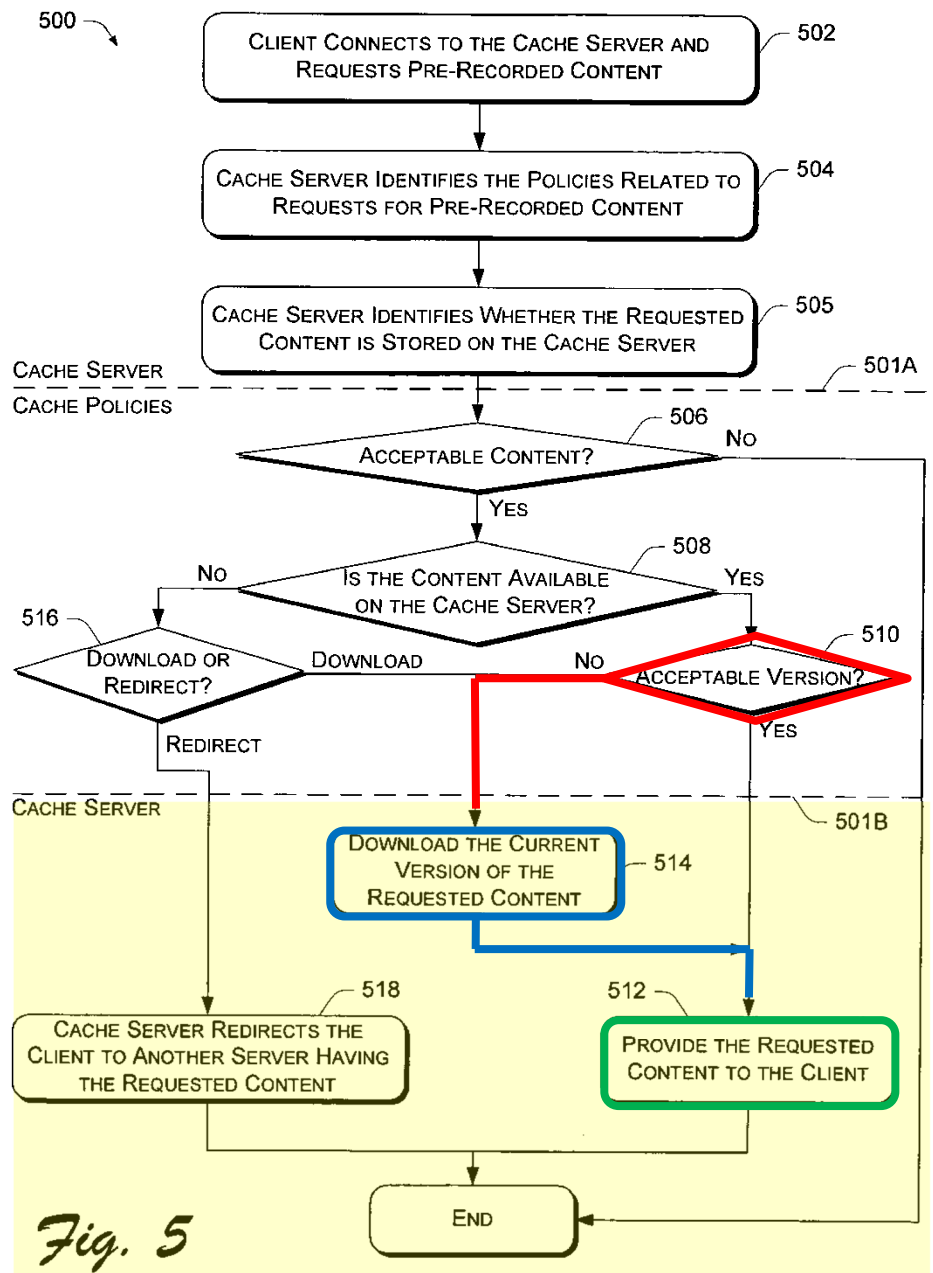
O'Rourke, Fig. 5.

h. 1[b][iii][A]/8[b][iii][A]

O'Rourke discloses/suggests 1[b][iii][A]/8[b][iii][A]. Lanning, ¶¶127-129.

“If the cached version of the requested content is not acceptable, then the procedure branches from block 510 to block 514, where the cache server downloads the current version of the requested content from the origin server” (*when it is not acceptable to serve the content that is cached locally, obtaining, by the node, a new version of the content*). O’Rourke, 6:40-43; Lanning, ¶128. After downloading a current version of the content from the origin server, “the cache server provides the current version to the client (512)” (*and then serving, by the node, the new version of the content*). O’Rourke, 6:51-54; Lanning, ¶128.

Figure 5 illustrates downloading of the current version of the requested content (514, blue) and providing the downloaded content to the client (512, green) when the cache policy determines the cached version is not acceptable (510, red). Lanning, ¶129.



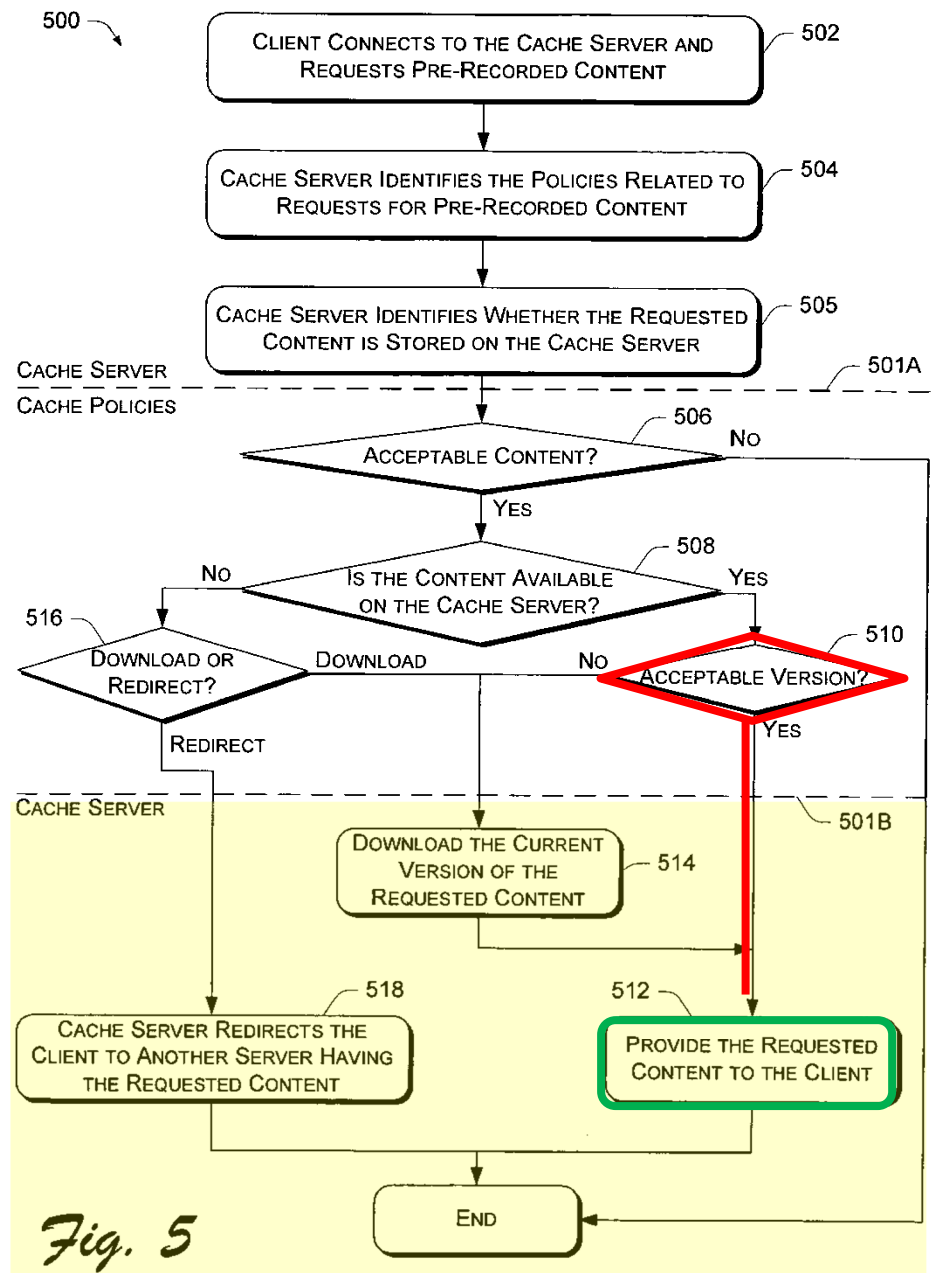
O'Rourke, Fig. 5.

i. 1[b][iii][B]/8[b][iii][B]

O'Rourke discloses/suggests 1[b][iii][B]/8[b][iii][B]. Lanning, ¶¶130-132.

“If the cached version is acceptable, the procedure continues [from block 510] to block 512, where the cache server provides the cached version of the requested content to the client” (*when it is acceptable to serve the content that is cached locally, serving, by the node, the content that is cached locally*). O’Rourke, 6:37-40; Lanning, ¶131.

Figure 5 illustrates providing the content to the client (512, green) when the cache policy determines the cached version is acceptable (510, red). Lanning, ¶132.



O'Rourke, Fig. 5.

3. Claims 2, 9

a. 2[pre]

If limiting, O'Rourke discloses/suggests 2[pre]. Lanning, ¶133.

O'Rourke's procedure 500 discloses/suggests *the method as recited in claim*

1. Lanning, ¶133; *see generally* §VII.A.2.

b. 9[pre]

If limiting, O'Rourke discloses/suggests 9[pre]. Lanning, ¶134.

O'Rourke's computer readable instructions and program modules are the *computer program product as recited in claim 8*, and O'Rourke's procedure 500 discloses/suggests what *the method further comprises*. Lanning, ¶134; §VII.A.2.

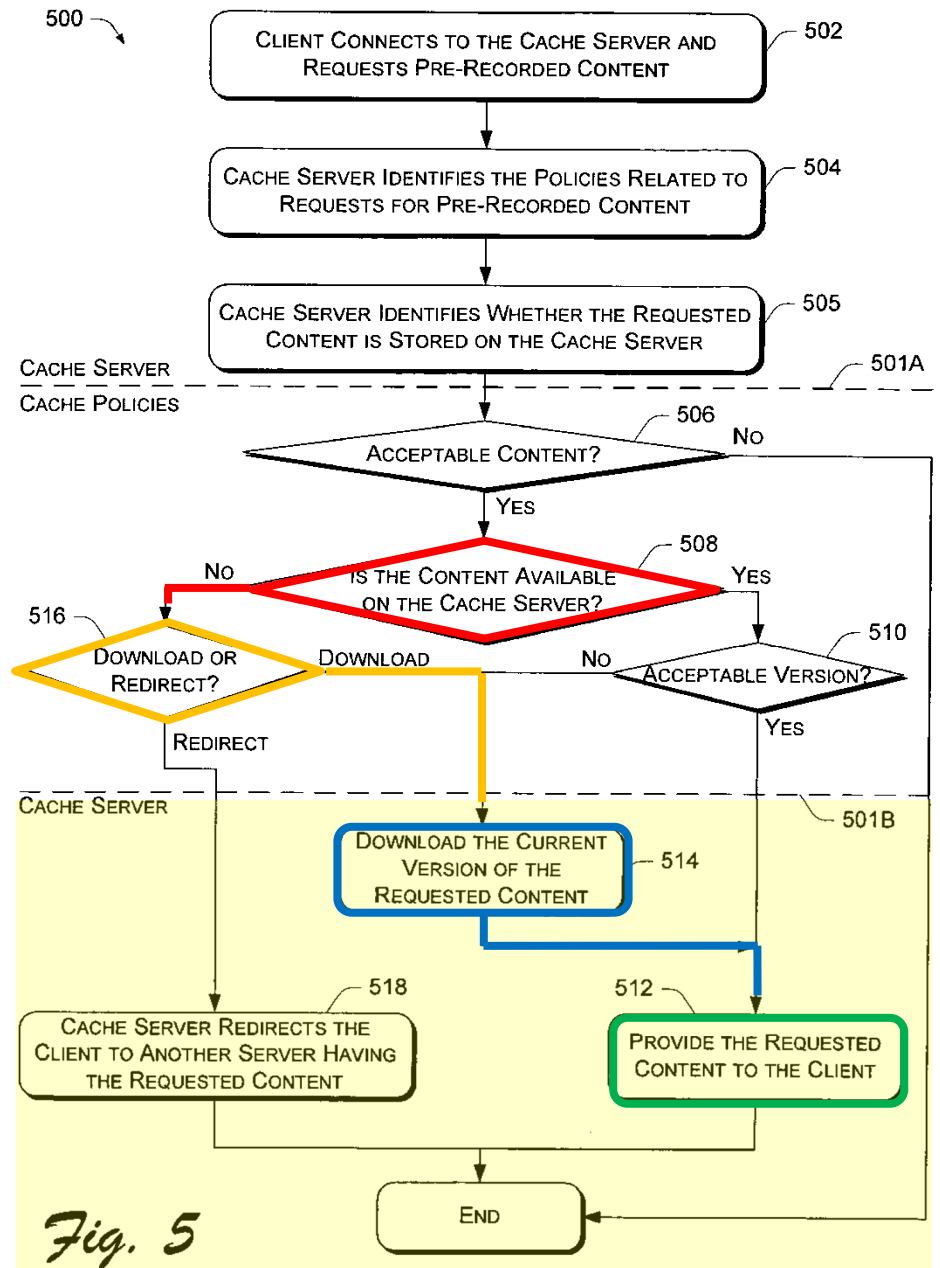
c. 2[a]/9[a]

O'Rourke discloses/suggests 2[a]/9[a]. Lanning, ¶¶135-137.

In O'Rourke, if it is “determine[d] that the requested content is not available on the cache server, then the cache policies determine whether to download the requested content or redirect the client to a different server” ((C) *when it is determined that said content is not cached locally*). O'Rourke, 6:55-58; Lanning, ¶136. Based on the cache policies, “the cache server downloads the current version of the requested content from the origin server” (*then (C)(1) obtaining the content*). O'Rourke, 6:58-62; Lanning, ¶136. After downloading the content from the origin server, the cache server *then (C)(2) serves the content obtained in (C)(1)*. O'Rourke, 6:51-54, Fig. 5; Lanning, ¶136.

Figure 5 illustrates downloading the current version of the requested content from the origin server (514, blue) and providing the content to the client (512, green)

when the requested content is not available on the cache server (508, red) and the cache policies determine to download the content (516, orange). Lanning, ¶137.



O'Rourke, Fig. 5.

4. Claims 4, 11

O'Rourke discloses/suggests claims 4 and 11's additional limitations. Lanning, ¶138-139.

O'Rourke describes that “cache policies are contained in one or more cache plug-in modules,” and that “the cache plug-in modules can run as part of another process...on a different computer system that supports the Windows Media Server.” O'Rourke, 4:57-60, 4:66-5:2. That is, the cache plug-in modules (containing the *current cache policy*) can be *determined from a location distinct from the node* (i.e., a different computer system than the cache server). Lanning, ¶139.

5. Claims 5, 12

O'Rourke discloses/suggests claim 5 and 12's additional limitations. Lanning, ¶¶140-141.

O'Rourke describes that “cache policies are contained in one or more cache plug-in modules,” and that “the cache plug-in modules can run as part of another process...on a different computer system that supports the Windows Media Server.” O'Rourke, 4:57-60, 4:66-5:2; §VII.A.4. Further, a POSITA would have understood that O'Rourke's storing of cache policies and/or cache plug-in modules on a different computer system (*location*) includes being stored on a database of the different computer system. Lanning, ¶141. For example, O'Rourke describes using data storage devices for storing data and other information used, such as cache

policies. *See, e.g.*, O'Rourke, 3:39-42, 5:17-21; Lanning, ¶141. A POSITA would have understood that when storing large amounts of data (e.g., cache policies) on storage devices, databases provided various advantages such as improved security, data integrity, and scalability. *See, e.g.*, Silberschatz, pp. 3-6 (describing the advantages of databases over other approaches to maintaining data, such as file-processing systems); Lanning, ¶141. And databases were commonly used to maintain information within CDNs. *See, e.g.*, Buyya, pp. 117-119 (describing different approaches to maintain and replicate databases at different locations of a CDN); Lanning, ¶141. Thus, it would have been obvious for the different computer system to utilize a *database*. Lanning, ¶141.

6. Claims 6, 13

O'Rourke discloses/suggests claims 6 and 13's additional limitations. Lanning, ¶¶142-143.

O'Rourke's cache server can be a Windows Media Server, and "[a]ny number of cache plug-in modules can be loaded into the Windows Media Server. The combination of all loaded cache plug-in modules defines the set of cache policies." O'Rourke, 4:57-63. "[T]he operation of a cache server can be modified simply by changing the cache plug-in modules in the cache server." *Id.*, 5:6-9. The cache policies control the operation of the cache server, such as what type of content to cache on the cache server, and how to respond to a cache miss. *Id.*, 7:52-53, 8:1-2;

Lanning, ¶143. Thus, the cache policies *relate to caching content within the CDN* (e.g., at the cache server). Lanning, ¶143.

B. Ground 1B: Obviousness over O’Rourke and MS-MSSO

1. MS-MSSO

MS-MSSO is a “Protocol Family System Document (PFSD) [that] is primarily intended to cover” Microsoft’s Media Streaming Server System Overview Protocol Family. MS-MSSO, 2, 6. It “provides an overview of the Windows Media protocols that are implemented in the Windows Server operating systems and that are used to interoperate or communicate with Windows client operating systems.” *Id.*, 2. That is, MS-MSSO “describes the intended functionality of the Media Streaming Server System and how it interacts with systems or applications that need to stream media” to or from a media server. *Id.* In other words, MS-MSSO describes Windows Media protocols used by Windows Media Servers. Lanning, ¶¶90-92.

MS-MSSO describes a number of protocols used by Windows Media Servers, including the Windows Media HTTP Streaming Protocol, which “depends on HTTP for the transfer of all protocol messages, including the transfer of the multimedia data. The multimedia data flows from the server to the client.” MS-MSSO, 44-47.

2. Motivation to Combine O'Rourke and MS-MSSO

A POSITA would have been motivated to modify the content delivery system of O'Rourke to utilize the HTTP Streaming Protocol taught by MS-MSSO. Lanning, ¶¶144-149.

O'Rourke identifies the problem that exiting cache servers were difficult to customize. §VII(A)(1); Lanning, ¶145. O'Rourke discloses a solution that allowed manufacturers to customize their cache servers. Lanning, ¶146.

O'Rourke's cache server may be a Windows Media Server. O'Rourke, 4:57-60. In this embodiment, "the set of cache policies are contained in one or more cache plug-in modules that are loaded in the Windows Media Server and called by the Windows Media Server," and these cache plug-in modules can "run in the same space as the Windows Media Server" or, alternatively, "can run as part of another process on the same computer system or on a different computer system that supports that Windows Media Server....Thus, the operation of a cache server can be modified simply by changing the cache plug-in modules in the cache server." *Id.*, 4:57-5:8.

A POSITA, aware of the benefits of O'Rourke to configure existing cache servers including the Windows Media Server, would have looked to MS-MSSO for further details about how the Windows Media Server operated and the supported protocols/functions. Lanning, ¶¶147-148. A POSITA would have further recognized

that the Windows Media HTTP Streaming Protocol, used by Windows Media Servers, provides additional advantages when, for example, “the multimedia data needs to pass through a proxy or a firewall.” MS-MSSO, 47; Lanning, ¶148. Accordingly, a POSITA, aware of the benefits both of allowing customizable caching policies and of optimizing data transmission in various network environments, including those involving proxies and firewalls, would have been motivated to further improve upon O’Rourke (cache servers with customizable caching policies, configured as a Windows Media Server) to utilize the Windows Media HTTP Streaming Protocol, as taught by MS-MSSO. Lanning, ¶148.

This would have involved combining prior art elements (MS-MSSO’s Windows Media HTTP Streaming Protocol, and O’Rourke’s CDN incorporating a Windows Media Server) according to known methods (deploying a Windows Media Server as a cache server as taught by O’Rourke, and utilizing the Windows Media HTTP Protocol as taught by MS-MSSO) to yield predictable results (improved caching and network performance). *Id.*

Furthermore, a POSITA would have had a reasonable expectation of success combining O’Rourke with MS-MSSO in this manner. Lanning, ¶149. O’Rourke already taught a cache server that could be implemented as a Windows Media Server. *Id.*; O’Rourke, 2:54-59, 5:10-15, 7:41-44. A POSITA would have recognized that the Windows Media Server taught in O’Rourke could be readily modified to utilize

the Windows Media HTTP Streaming Protocol. Lanning, ¶149. This combination would have been well within the capabilities of a POSITA because O’Rourke provides details regarding how to use a Windows Media Server as its cache server, and MS-MSSO describes using various protocols within such a Windows Media Server, including the Windows Media HTTP Streaming Protocol. *Id.* Additionally, the resulting system’s elements would perform functions performed prior to combination—O’Rourke’s CDN, utilizing a Windows Media Server, would allow users to customize caching policies, and utilizing the Windows Media HTTP Streaming Protocol would improve network performance. *Id.*

3. Claims 3, 10

O’Rourke-MS-MSSO teaches claims 3 and 10’s additional limitations. Lanning, ¶¶150-152.

O’Rourke’s cache server may be a Windows Media Server. O’Rourke, 4:57-60. A POSITA would have understood that Windows Media Servers utilize HTTP. Lanning, ¶151. For example, the Windows Media HTTP Streaming Protocol, used by Windows Media Servers, “depends on HTTP for the transfer of all protocol messages, including the transfer of the multimedia data.” MS-MSSO, 47; Lanning, ¶151. In particular, “[t]he Windows Media HTTP Streaming Protocol depends on HTTP 1.0.” MS-MSSO, 47. Using Windows Media HTTP Streaming Protocol was advantageous when, for example, “the multimedia data needs to pass through a

proxy or a firewall.” MS-MSSO, 47. It would therefore have been obvious in the O’Rourke-MS-MSSO system for the cache server configured as a Window Media Server (as taught by O’Rourke) to utilize the Windows Media HTTP Streaming Protocol (as taught by MS-MSSO). Lanning, ¶151. Thus, in the O’Rourke-MS-MSSO combination, requests to the cache server *comprises an HTTP request. Id.; see also* Buyya, pp. 18, 19.

Additionally, O’Rourke describes “procedure 500 for handling a client request to receive pre-recorded content.” O’Rourke, 6:1-5. The pre-recorded content can be “media content, such as movies, television programs, concerts, sporting events, and the like,” and the cache server is capable of “providing a higher quality real-time video image to the local clients.” *Id.*, 3:46-55. In other words, the requested content *comprises one or more resources* (e.g., audio and/or video data). Lanning, ¶152; *see also* MS-MSSO, 46 (“The Windows Media HTTP Streaming Protocol is used to transfer real-time multimedia data (that is, audio and video)”).

C. Ground 1C: Obviousness over O’Rourke and Blumofe

1. Blumofe

Blumofe teaches “extend[ing] [a CDN] into one or more mobile or wireline networks that cannot or do not otherwise support fully-managed CDN edge servers.” Blumofe, Abstract. Blumofe recognized that “[b]ecause of the way that fixed or wireless mobile networks are architected and operated, it is not always possible or

practical to deploy fully-managed CDN servers therein (i.e., close to end-users).” Blumofe, [0007]. For example, Blumofe observed that in 3G wireless mobile networks, the gateways used to connect wireless end-users to the Internet, at which CDN servers could be deployed, may be thousands of miles and hundreds of milliseconds away from the end-users. *Id.*

To address this, Blumofe describes a CDN Extender. *Id.*, [0025]; Lanning, ¶¶87-88. The Extender may be implemented in network environments that do not support CDN services (such as a wireless mobile network), and operated as a cache beyond the CDN’s edge. Blumofe, [0025], [0027]; Lanning, ¶¶88.

Figure 3 illustrates how the Extender 300 (blue) is implemented in a network infrastructure (green) outside the CDN (red).

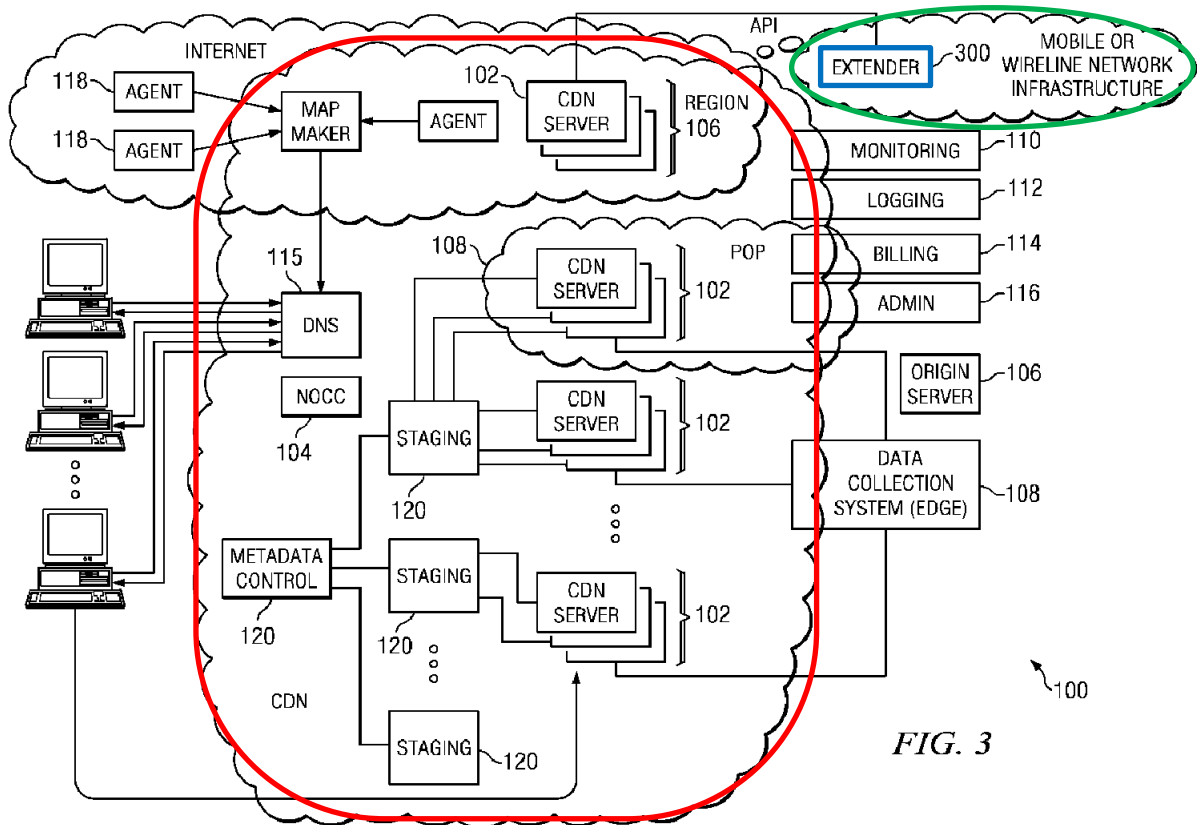


FIG. 3

Blumofe, Fig. 3.

When the Extender receives a request for content, it determines whether the requested content is already in the Extender's cache. Blumofe, [0033]. If not, the Extender sends an HTTP GET request to the CDN to fetch the content, based on which the Extender responds to the end-user. *Id.* The CDN's response to the Extender's request may also include cache control information, such as expiration time. *Id.*, [0037].

2. Motivation to Combine O'Rourke and Blumofe

A POSITA would have been motivated to modify O'Rourke's content delivery system to add an extender as Blumofe teaches. Lanning, ¶¶153-164.

O'Rourke's background explains that within a network, cache servers could be deployed between origin servers and end users to improve network performance. O'Rourke, 1:31-41. O'Rourke identified the problem that existing cache servers were difficult to customize. §VII(A)(1); Lanning, ¶154-155. O'Rourke disclosed a solution that allowed manufacturers to customize their cache servers. *Id.*

O'Rourke's solution may be used in various network environments. Lanning, ¶166. Specifically, O'Rourke's invention could advantageously be used “to distribute media content across a network” and “[c]ache server 300 may use any type of data communication interfaces necessary to communicate with other devices.” O'Rourke, 2:54-59, 5:10-15. Moreover, O'Rourke teaches that the cache policies may be defined based, *inter alia*, on the type of network and the other components in the system. *Id.*, 7:41-44. O'Rourke discloses preferred embodiments that could be used in LAN and WAN networks. *Id.*, 3:34-39, 9:39-53. However, O'Rourke is silent regarding how to extend the invention to other types of networks, e.g., wireless networks. Lanning, ¶156.

Blumofe teaches “extend[ing] [a CDN] into one or more mobile or wireline networks that cannot or do not otherwise support fully-managed CDN edge servers.”

Blumofe, Abstract; §VII(C)(1). Blumofe observes that in 3G wireless mobile networks, gateways at which CDN servers could be deployed may be thousands of miles and hundreds of milliseconds away from end-users. Blumofe, [0007]. To address this, Blumofe describes a CDN Extender. *Id.*, [0025]. The Extender is deployed on the network, e.g., “as a passive web caching proxy that is beyond the edge of the CDN but that serves CDN-provisioned content under the control of the CDN.” *Id.*, [0009]. The CDN controls the Extender, and the CDN supplies caching policies to the Extender, i.e., caching policies within the CDN related to caching content outside the CDN. *Id.*, [0036]. This enables serving “CDN-provisioned content under the control of the CDN” in situations where it was not “possible or practical to deploy fully-managed CDN servers.” *Id.*, [0007]-[0009].

A POSITA, aware of the benefits both of allowing customizable caching policies and of providing caching functionality close to wireless end users in various network environments, would have been motivated to improve upon O’Rourke (cache servers with customizable caching policies) to allow for further caching resources within a wireless network. Lanning, ¶¶154-160. And a POSITA would have recognized Blumofe would provide this benefit to O’Rourke. *Id.*

O’Rourke and Blumofe are directed to content delivery networks. O’Rourke, 1:19-21; Blumofe, [0005]. And both describe shortcomings with conventional CDN architectures and are directed to solving these problems by managing caches

deployed between origin servers and end users. O’Rourke, 4:10-22; Blumofe, [0009]. Both references describe methods for the content owner to customize the cache policies used for caching content. O’Rourke, 5:41-47; Blumofe, [0030].

Moreover, Blumofe teaches an Extender designed to work in combination with an existing CDN, such as O’Rourke’s CDN. Lanning, ¶161. Blumofe explains that the Extender “is deployed in the mobile or wireline network, preferably as a passive web caching proxy that is beyond the edge of the CDN but that serves CDN-provisioned content” and the Extender “preferably delivers all forms of CDN content (i.e., content that has been off-loaded by content providers for delivery by the CDNSP [CDN service provider, i.e., the entity managing the CDN] over the CDN).” Blumofe, [0009], [0029]; Lanning, ¶161.

A POSITA would have found it obvious to combine O’Rourke, which describes a method of managing cache servers within a CDN (*see generally supra*), with Blumofe, which describes managing caches both within and outside of the CDN (e.g., within the Extender). Lanning, ¶162. That is, a POSITA would have found it obvious to utilize an extender, as taught by Blumofe, with the CDN taught by O’Rourke. *Id.* A POSITA would have recognized the benefits this combination provides. *Id.* For example, using an extender as taught by Blumofe would have allowed using O’Rourke’s cache server in conjunction with other network environments (e.g., wireless networks) where it otherwise could not have been

deployed. *Id.* And this combined system would likewise have allowed caching content closer to wireless users, providing improved network performance. *Id.* Moreover, using an extender (as taught in Blumofe) with O'Rourke's cache server would enable consistently enforcing cache policies both inside and outside the CDN. *Id.* This would have involved applying a known technique (caching content in wireless networks) to improve similar devices (CDNs) in the same way (deploying caches in other network environments and/or closer to the user). *Id.* Further, this would have involved combining prior art elements (an extender as taught by Blumofe and O'Rourke's CDN) according to known methods (providing cache servers between origin servers and end users) to yield predictable results (improved network performance). *Id.*

A POSITA would have had a reasonable expectation of success combining O'Rourke with Blumofe in this manner. *Id.*, ¶163. O'Rourke already taught a cache server that could be adapted to various networks. *Id.*; O'Rourke, 2:54-59, 5:10-15, 7:41-44. A POSITA would have recognized that an extender as taught in Blumofe could be readily incorporated into O'Rourke's system. Lanning, ¶163. This combination would be well within the capabilities of a POSITA because an extender as taught in Blumofe was designed to work with an existing CDN. *Id.*, *see also* Blumofe, [0009], [0029]. Moreover, O'Rourke and Blumofe describe similar CDN environments and communications between origin servers, caches, and clients, such

that a POSITA would have found it straightforward to modify O'Rourke's CDN to implement an extender as taught in Blumofe. Lanning, ¶163. Additionally, the resulting system's elements would perform functions performed prior to combination—O'Rourke's CDN would allow users to customize caching policies, and the extender would enable caching content in wireless networks and/or closer to the user. *Id.*

A POSITA would have understood that in such a combination, the CDN would send the cache control information to the extender so that the extender would perform according to the CDN customer's rules, and the cache policy that configures the cache server within the CDN (e.g., with respect to whether cached content has expired or is an acceptable version) would also be used to configure the extender outside the CDN. *Id.*, ¶164; *see also* Blumofe, [0030], [0036]-[0037]. The combination would have been well within a POSITA's capabilities because it would have merely required modifying O'Rourke's CDN to include an extender taught by Blumofe. Lanning, ¶164.

3. Claims 7, 14

O'Rourke-Blumofe teaches claims 7 and 14's additional limitations. Lanning, ¶¶165-169.

Blumofe describes a CDN Extender, such that “[t]he CDN is extended into one or more mobile or wireline networks that cannot or do not otherwise support

fully-managed CDN edge servers.” Blumofe, [0009]. The Extender “is deployed in the mobile or wireline network, preferably as a passive web caching proxy that is beyond the edge of the CDN but that serves CDN-provisioned content.” *Id.* The combined O’Rourke-Blumofe system’s extender is thus *outside the CDN*. Lanning, ¶166; *see also* Blumofe, [0033]. Although not part of the CDN, the extender “preferably delivers all forms of CDN content (i.e., content that has been off-loaded by content providers for delivery by the CDNSP over the CDN).” Blumofe, [0029]. Thus, “the Extender can cache and deliver cacheable content, including on-demand media.” *Id.* In other words, any CDN content cached by the extender in the combined O’Rourke-Blumofe system is similarly *cached content outside the CDN*. Lanning, ¶166; *see also* Blumofe, [0050].

Blumofe describes that when the Extender receives an end-user request for content, the Extender determines whether the requested content is already in the Extender’s cache. *Id.*, [0033]. When the Extender determines the content is not in its cache, it sends an HTTP GET request to the CDN to fetch the content. *Id.* Based on the response received from the CDN (e.g., with the content), the Extender “serv[es] to the requesting end-user an appropriate HTTP response code and/or the content request.” *Id.*; Lanning, ¶167.

Blumofe additionally explains that “[t]o deliver CDN content, the Extender acts under the control of the CDN,” which is necessary because “it is desired (by the

CDN customer) that the correct content be delivered according to customer metadata rules.” Blumofe, [0030]. Thus, “the Extender caches and serves the CDN customer content based on information it receives, in terms of content and cache-control instructions, from the CDN edge servers.” *Id.*, [0036]. For example, “[w]hen the CDN edge server responds to an Extender, preferably it communicates some or all available cache control information in one or more HTTP response headers.” *Id.*, [0037]. The cache control information can include, for example, “time-to-live (TTL) expiration times.” *Id.*

In the combination, the cache control information the CDN returns controls the expiration time of the cached content on the extender, and thus *relates to caching content outside the CDN*. Lanning, ¶¶168-169. Moreover, Blumofe describes that the CDN sends the cache control information to the Extender so that the Extender performs according to the CDN customer’s rules. Blumofe, [0030], [0036]; Lanning, ¶169. And it would have been obvious, in the combined O’Rourke-Blumofe system, that the *cache policy* that configures the cache server within the CDN (e.g., with respect to whether cached content has expired or is an acceptable version) would also be used to configure the extender outside the CDN. Lanning, ¶169. Otherwise, the extender would operate differently than the CDN’s cache server. *Id.*; *see also* Blumofe, [0054]. Thus, the *cache policy* (*see* §VII.A.2.e) *also relates to caching content outside the CDN*. Lanning, ¶169.

D. Ground 2A: Obviousness over Middleton

1. Middleton

Middleton teaches “an extensible content delivery platform” that “provide[s] a logical flow of actions [for]...the content delivery process of the content delivery network.” Middleton, Abstract. Middleton recognized that in the CDNs of the time, the cache servers, while “feature rich,” were challenging to configure and required significant code changes to add new features. *Id.*, [0005]-[0006].

To address this and reduce complexity of configuring cache servers, Middleton teaches an “extensible content delivery platform” in which “[r]ich and well-formed metadata and ‘metacode’ can be associated with content and/or requests in a consistent and programmatic manner.” *Id.*, [0007]. Middleton explains that this “extensible content delivery platform also provides for flexible implementation...without core code changes,” and is provided by an edge server with an “event-based application programming interface (API).” *Id.*, [0010], [0035].

The edge server is part of a CDN and “provide[s] requested content to a requester,” such as an end user computer or intermediate device. *Id.*, [0031]-[0032]. When the edge server receives a request for content, it “determine[s] whether the requested content is available locally...to be provided to the requester” (e.g., whether the content is stored locally in cache and not stale). *Id.*, [0033]. If the content

is available locally, the edge server provides the requested content to the requester. *Id.*

If, however, the edge server determines that the content is stored locally but stale, it attempts to revalidate the content. *Id.*, [0034]. To revalidate, the edge server sends a request to a media access server (MAS). *Id.* If the content is revalidated, the MAS sends the edge server a new time-to-live (TTL) value for the content, based on which the edge server updates its TTL value associated with the content. *Id.*, [0057]-[0059]. Thus, based on the MAS response, the edge server re-characterizes locally stored content (that had been considered stale) using the new TTL value. Lanning, ¶¶81-84.

If, on the other hand, the MAS indicates that the content is not revalidated, the edge server deletes the stale content from the cache. Middleton, [0059]. Thus, the ultimate determination of whether locally-stored content is truly stale depends on the results of a revalidation request made to the MAS. Lanning, ¶85.

Figure 4 illustrates the states of content in the described system. Middleton, [0069]. Content is stored in the cache and not stale (state C, 406) until the content's age exceeds the content's specified TTL value. *Id.*, [0076]. When the content's age exceeds the content's specified TTL value, the content transitions to being stale (state D, 408). *Id.* The content may, however, be recharacterized as not stale upon successful revalidation. *Id.*, [0077]-[0078].

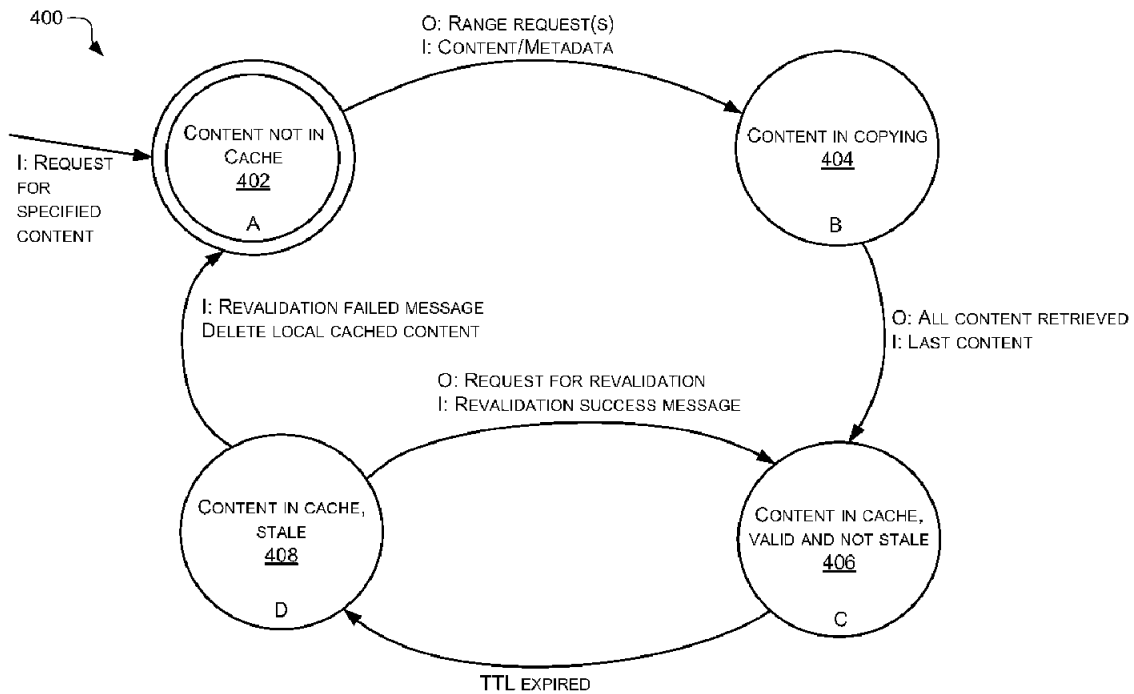


Fig. 4

Middleton, Fig. 4.

2. Claims 1, 8

a. 1[pre]

If limiting, Middleton discloses/suggests 1[pre]. Lanning, ¶170.

Middleton describes “*methods...for providing an extensible content delivery platform,*” which include “*identifying a plurality of discrete events of a content delivery process of a content delivery network.*” Middleton, Abstract. Middleton’s content delivery platform methods are *method[s], operable with a content delivery network (CDN).* Lanning, ¶171.

Middleton's extensible content delivery platform may be supported by a CDN with an edge server. Middleton, [0027], [0029], [0031]. CDN 102 (red) and edge server 108 (blue) are illustrated in Figure 1. *Id.* The edge server "comprises an extensible content delivery platform, such as an event-based application programming interface (API) that provides an extensible content delivery platform." *Id.*, [0035]. Middleton's extensible content delivery platform methods, including those performed by the edge server and including an API, are *computer-implemented method[s]*. Lanning, ¶¶172-73.

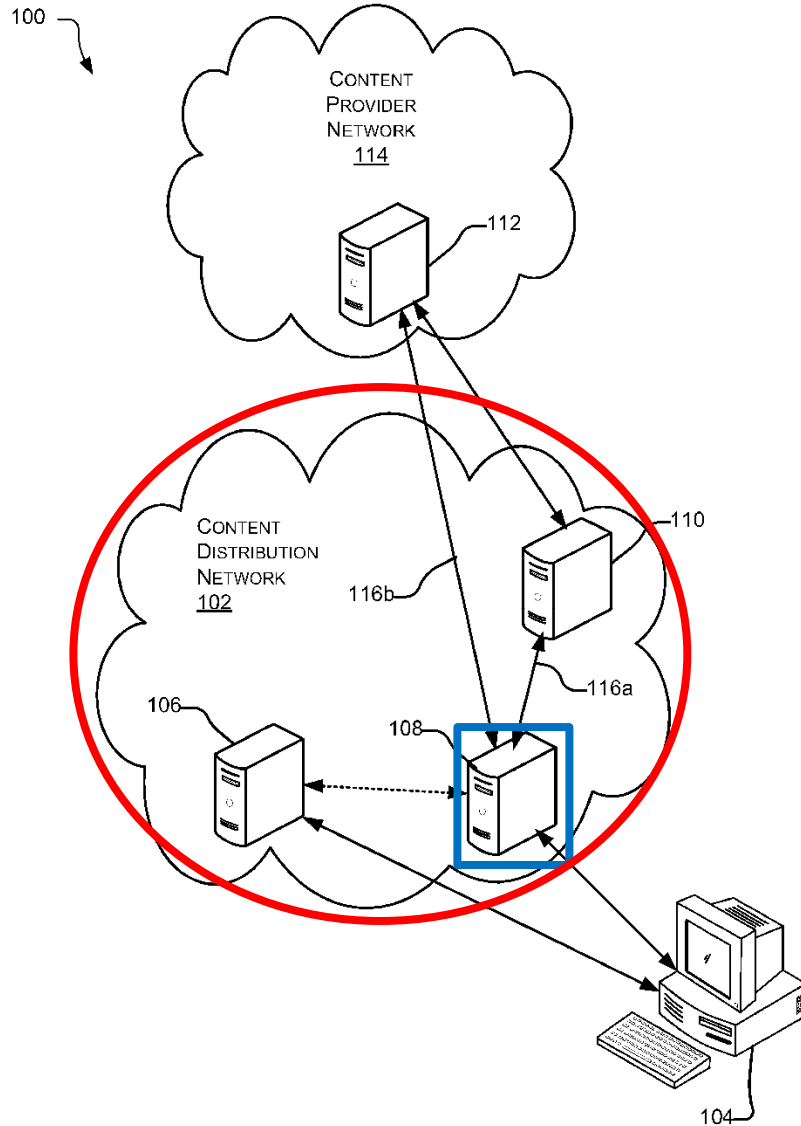


Fig. 1

Middleton, Fig. 1.

b. 8[pre]

If limiting, Middleton discloses/suggests 8[pre]. Lanning, ¶174.

Middleton describes “a computer system 1200 upon which embodiments of the present invention may be implemented and carried out. For example, one or more

computing devices 1200 may be used to support an extensible content delivery platform (e.g., for streamed content within a content distribution network).” Middleton, [0123]. The computer system (*computer program product*) includes a mass storage 1207 (*non-transitory computer readable media*), which “can be used to store information and instructions” (*computer readable instructions*). *Id.*, [0124]-[0125]. Middleton describes that these machine-executable instructions are used to perform operations of the extensible content delivery platform (*instructions for implementing a computer-implemented method, said method operable on a device comprising hardware including memory and at least one processor and running a service on said hardware, said method operable in a content delivery service (CDN)*). *Id.*, [0131], [0133]; Lanning, ¶¶175-77.

c. 1[a]/8[a]

Middleton discloses/suggests 1[a]/8[a]. Lanning, ¶¶178-79.

Middleton discloses “a streaming content delivery framework 200 adapted to supporting an extensible content delivery platform including an edge server computer 202 and a media access server computer 204.” Middleton, [0047]. The edge server (*node*) can retrieve content from the media access server (MAS) as well as stream content to a requesting entity. *Id.* Edge server includes various modules, such as a media streaming server, a media streaming broker, a stream caching module, and a content cache. *Id.*, [0048]. Figure 2 illustrates edge server 202 (blue)

and MAS 204 (green), and the modules therein. *Id.*, [0047]. Edge server is part of the content delivery framework 200 (red), and thus is a node *[in]/[of]* the CDN.

Lanning, ¶¶180-81.

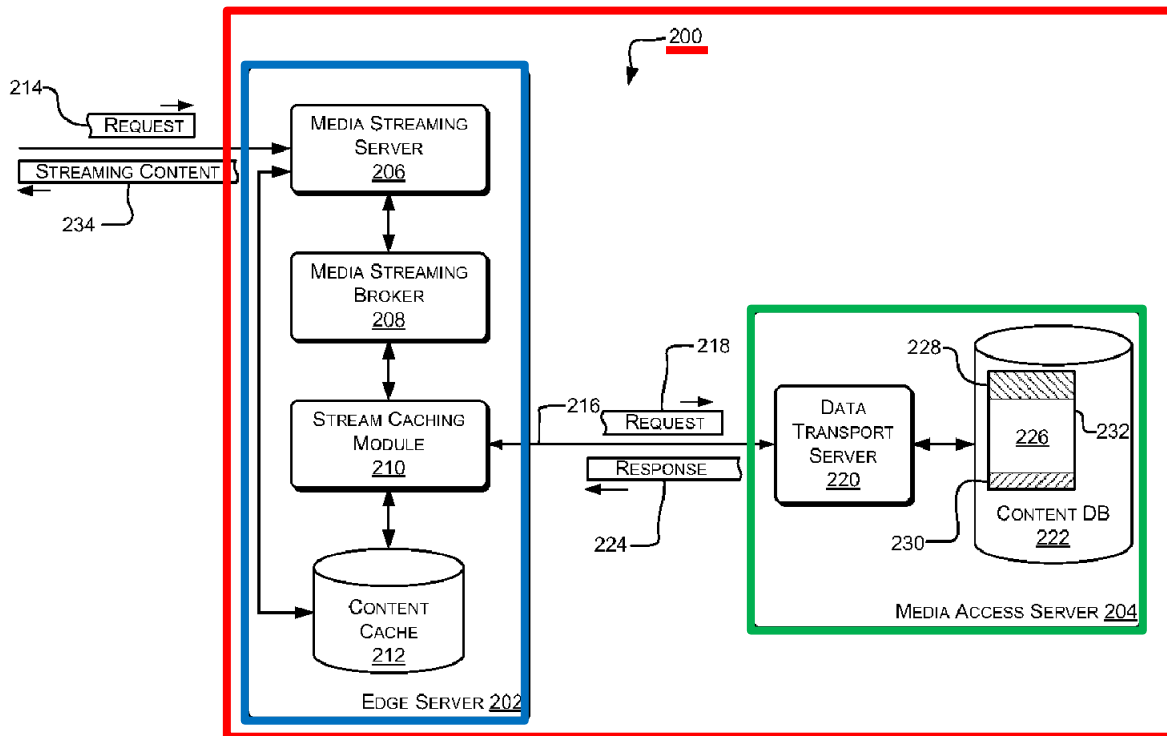


Fig. 2

Middleton, Fig. 2.

Figure 5 is a “flow chart[] illustrating [a] process[] for handling a request to deliver content,” and particularly illustrates a “content request handling operation 500” that supports the extensible content delivery platform. Middleton, [0079], [0080]. The operations “can be performed by functional modules such as one or more of the media streaming server 206, streaming media broker 208 and stream caching module 210 (FIG. 2), or other modules.” *Id.*, [0079]. Because Middleton’s

content request handling operation 500 can be performed by modules of the edge server (*node*), the steps are done *by a node in the CDN*. Lanning, ¶182.

Figure 5 illustrates that the content request handling operation 500 begins when “a request is initially received for specified content in receiving operation 502” (red). Middleton, [0080]. “The requested content is identified in the request.” *Id.* The operations that follow operation 502 are thus *in response to a request for content*. Lanning, ¶183.

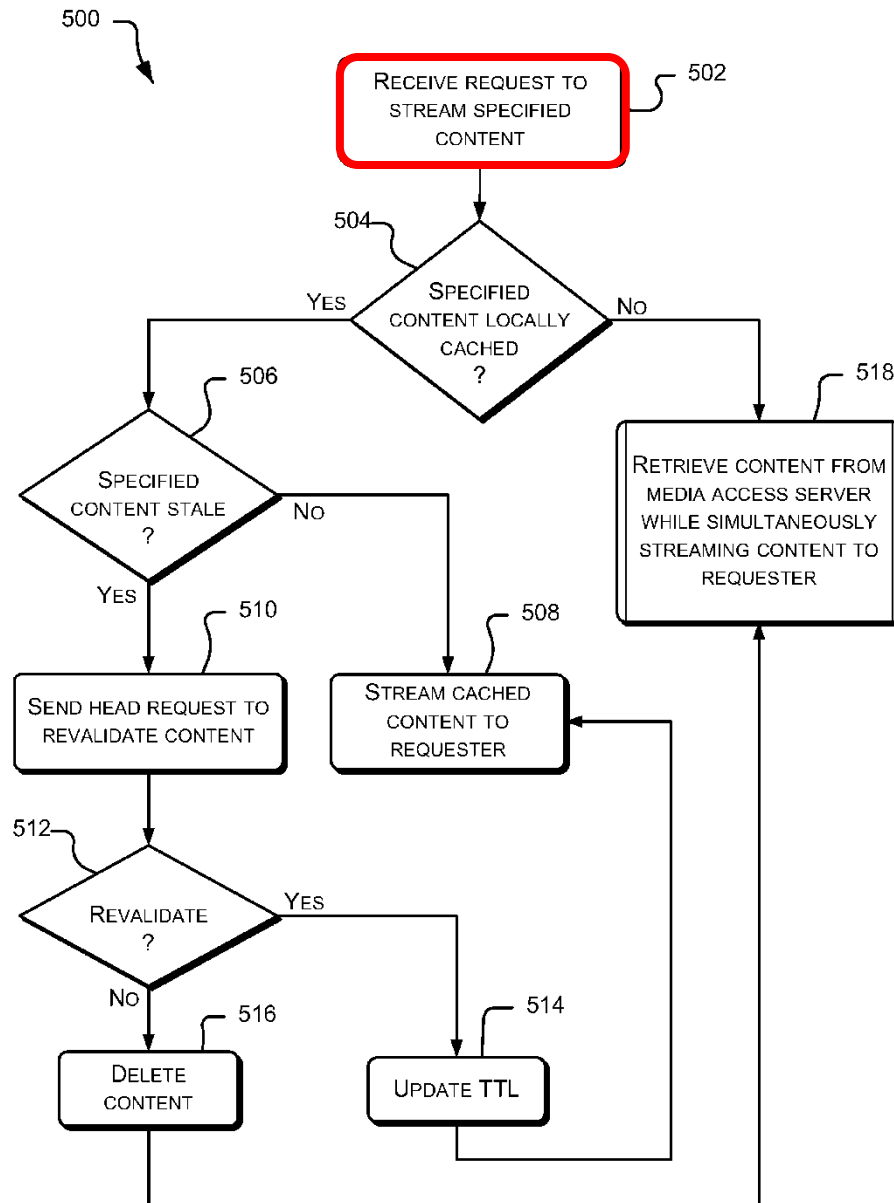


Fig. 5

Middleton, Fig. 5.

After receiving the content request, the content request handling operation 500 determines, at query operation 504, “if the requested content exists in local cache”

(*determining...if the content is cached locally at the node*). Middleton, [0080]; Lanning, ¶184. As Figure 5 shows below, whether the content exists in the local cache is determined at operation 504 (blue), after the content request is received at operation 502 (red), and is thus *in response to* the client request. Middleton, [0051], [0080]; Lanning, ¶184.

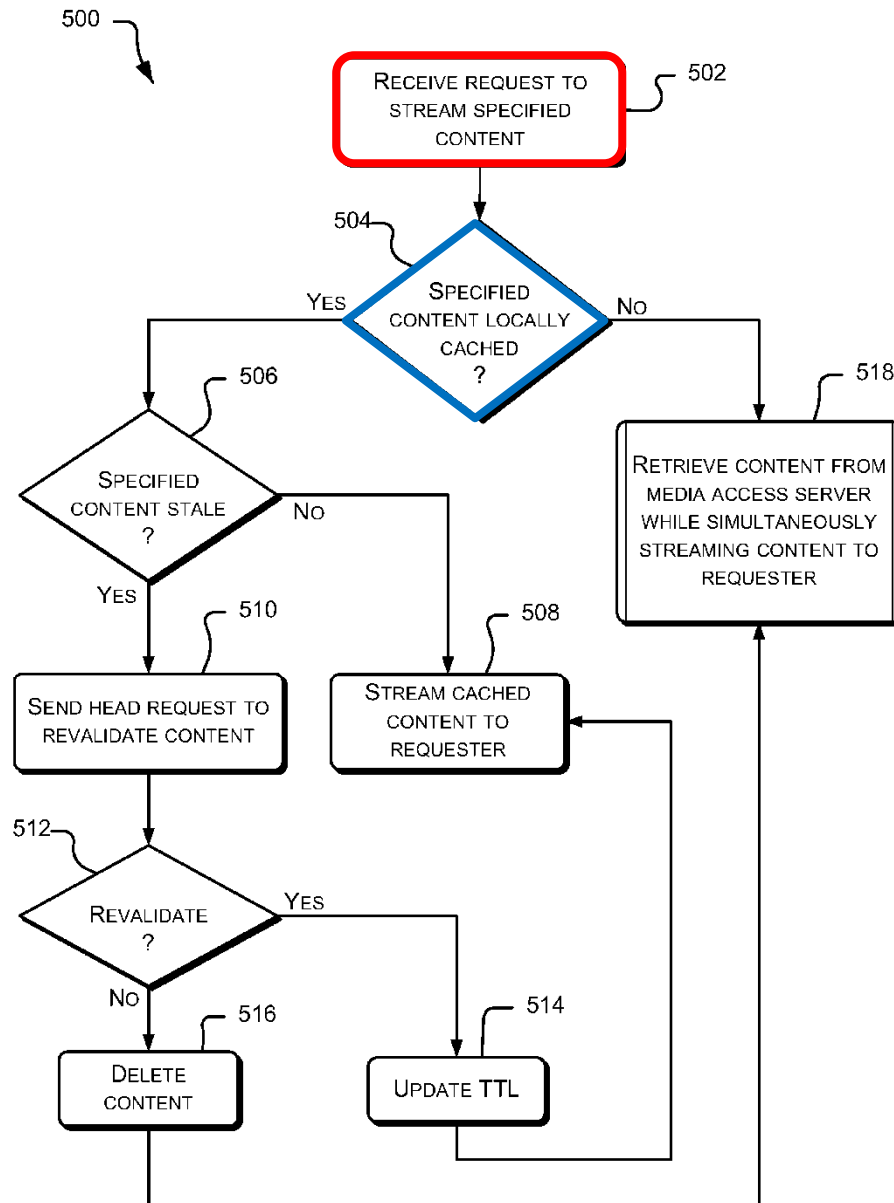


Fig. 5

Middleton, Fig. 5.

d. 1[b]/8[b]

Middleton discloses/suggests 1[b]/8[b]. Lanning, ¶¶185-86.

Middleton describes “[a] query operation 504 [that] determines if the requested content exists in local cache.” Middleton, [0080]; §VII.D.2.c. As described in detail herein, Middleton describes different operations that are performed depending on whether the requested content is in the cache. *See, e.g.*, Middleton, [0080] (describing proceeding to operation 506 if requested content is in local cache), [0083] (describing proceeding to operation 518 if requested content is not in local cache), [0052]-[0053]. Operations of Middleton’s content request handling operation 500 that are performed after operation 504 determines whether the requested content exists in the local cache—such as operation 506 (and other operations following the “YES” branch (green) of operation 504 (blue) in Figure 5)—are thus performed *when it is determined [by the node]/[] that said content is cached locally and after determining that the content is cached locally*. Lanning, ¶¶187-88.

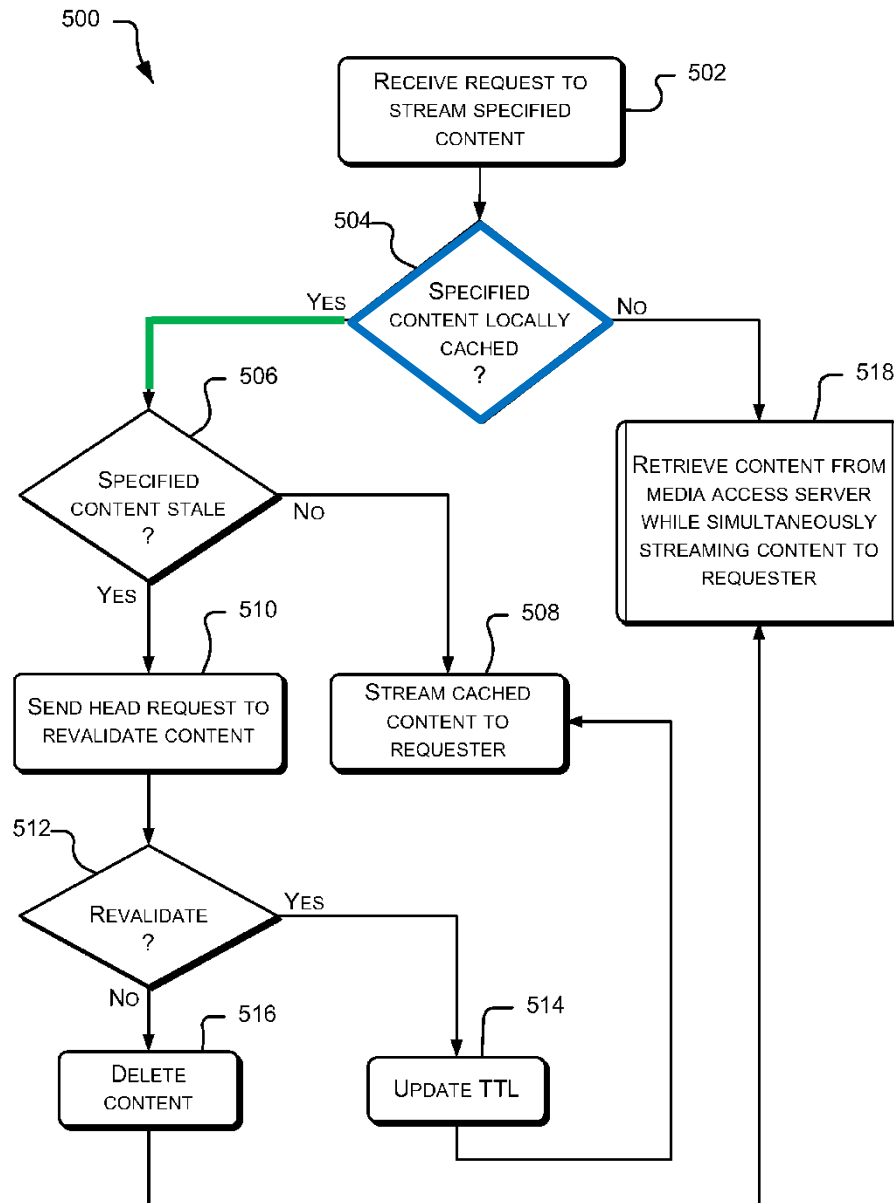


Fig. 5

Middleton, Fig. 5.

e. 1[b][i]/8[b][i]

Middleton discloses/suggests 1[b][i]/8[b][i]. Lanning, ¶¶189-90.

After determining the requested content exists in the local cache, content request handling operation 500 performs “query operation 506 [to] determin[e] if the content in local cache is stale.” Middleton, [0080]. The query operation “compares the age of the locally cached content to a TTL value associated with the content, and if the age is greater than the TTL value, the content is stale; otherwise the content is not stale.” *Id.* Because the content’s TTL value is compared to the age of the locally-cached content to determine if the content is stale, the content’s TTL value *determin[es] a time period for expiration of content in the cache*. Lanning, ¶191. And as described in detail herein, edge server’s response to the content request (e.g., whether to provide the locally-cached version of the content) is based on whether the locally-cached content is stale, which is determined based on the TTL value associated with the content, and thus the content’s TTL value is a *cache policy associated with the content*. *Id.*, ¶192.

“If the content is locally stored but is stale,” edge server takes a “remedial action [that] involves attempting to revalidate the content.” Middleton, [0034]. As described in detail below, when edge server (*node*) attempts to revalidate the content, it *determin[es]...a current cache policy associated with the content*. Lanning, ¶193. Specifically, in response to determining that locally-cached content is stale, Middleton performs a “sending operation 510, [in which] a HEAD request is sent to a media access server (MAS) to revalidate the locally cached content.” Middleton,

[0054], [0081]-[0082]. Edge server then performs a “query operation 512 [that] checks the response from the MAS to determine whether the locally cached content is revalidated.” *Id.*, [0082]. The response from MAS includes “information indicating whether the TTL of the content in cache 212 is revalidated or not.” *Id.*, [0057]. That is, MAS may indicate that it does not revalidate the content. *Id.*, [0077]. Alternatively, “[t]he MAS may send a message indicating successful revalidation and a new TTL value.” *Id.*, [0077]-[0078]. Based on an MAS response indicating successful revalidation (including a new TTL value), edge server “updates the TTL value associated with the locally cached content.” *Id.*, [0059], [0082]; Lanning, ¶194.

Figure 5 illustrates that the attempt to revalidate the locally-cached content (green) and determination of whether the revalidation was successful (blue) are performed after the content request is received (red), and thus the response from MAS—such as a revalidation failure or a new TTL value for the content with a successful revalidation—is *current*. Lanning, ¶195. Furthermore, edge server (*node*) performs the revalidation attempt, and responds accordingly (e.g., updating the content’s associated TTL value or deleting the locally-cached content). Middleton, [0079], [0082]-[0083]. Thus, the information used by the edge server in response to the revalidation attempt (e.g., an indication of revalidation failure or new TTL value) (*current cache policy associated with the content*) is *determin[ed]*, *by the node*. Lanning, ¶196.

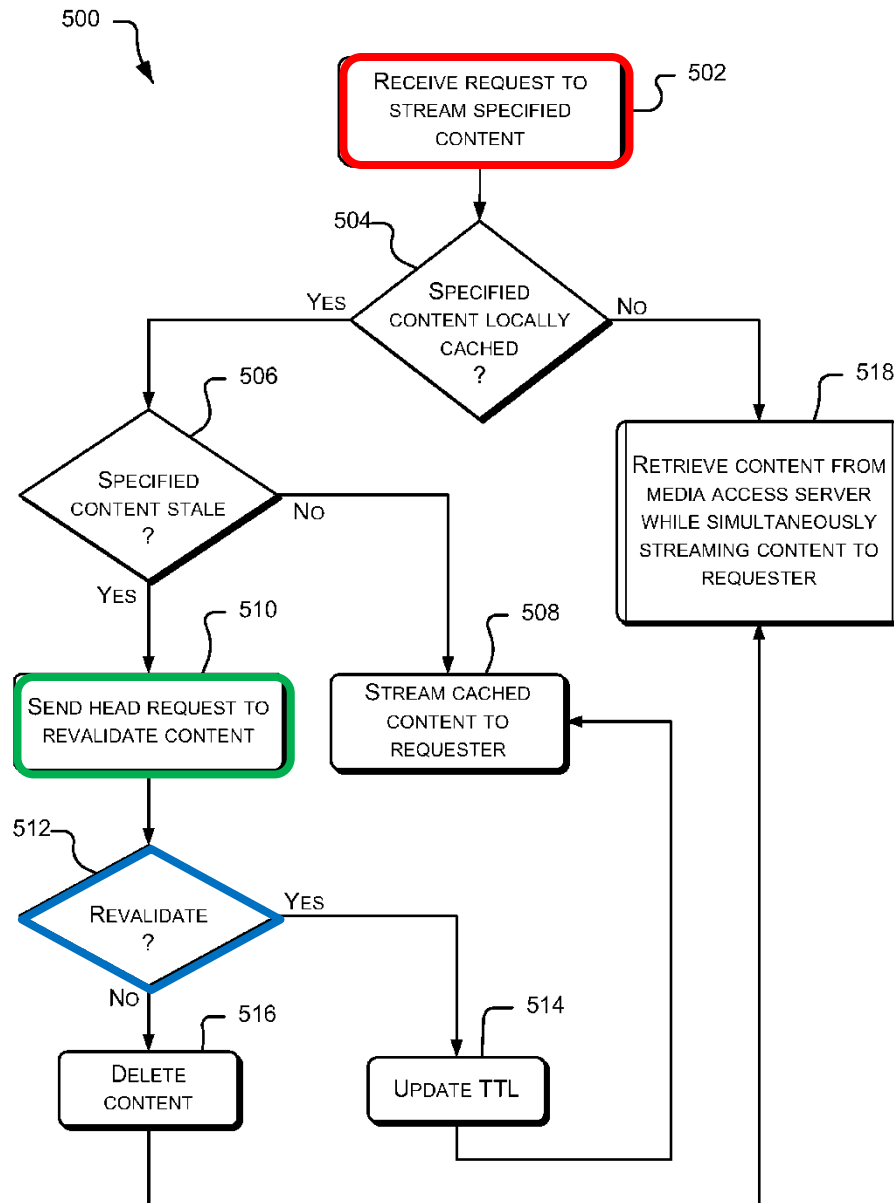


Fig. 5

Middleton, Fig. 5.

f. 1[b][ii]/8[b][ii]

Middleton discloses/suggests 1[b][ii]/8[b][ii]. Lanning, ¶¶197-98.

Middleton describes the edge server attempting to revalidate locally-cached content when it determines that the locally-cached content is stale (e.g., based on the age of the locally-cached content and the TTL value associated with the content). *See, e.g.*, Middleton, [0080]-[0082]; §VII.D.2.e. The response from MAS to edge server indicates whether the revalidation attempt was successful. *Id.*, [0082], [0083].

Figure 5 illustrates that the content request handling operation 500 performs different operations depending on whether revalidation attempt (blue) was successful (the “YES” branch, green) or unsuccessful (the “NO” branch, red). Lanning, ¶¶199-200. When the MAS response indicates that the locally-cached content is not revalidated, edge server deletes the locally-cached content. Middleton, [0083]. Alternatively, and as described in detail herein, when the locally-cached content is revalidated, the MAS response includes a new TTL value associated with the content, so that the locally-cached content is no longer stale and can be streamed. *Id.*, [0077], [0082]. The MAS response to the revalidation attempt (e.g., an indication of revalidation failure or new TTL value) that is used by the edge server is a *current cache policy associated with the content*. Lanning, ¶201; §VII.D.2.e. Thus, when edge server (*node*) determines whether the revalidation attempt was successful (*determining...whether it is acceptable to serve the content that is cached locally*) based on the MAS response, the determination is made *by the node and based on said current cache policy associated with the content*. Lanning, ¶201.

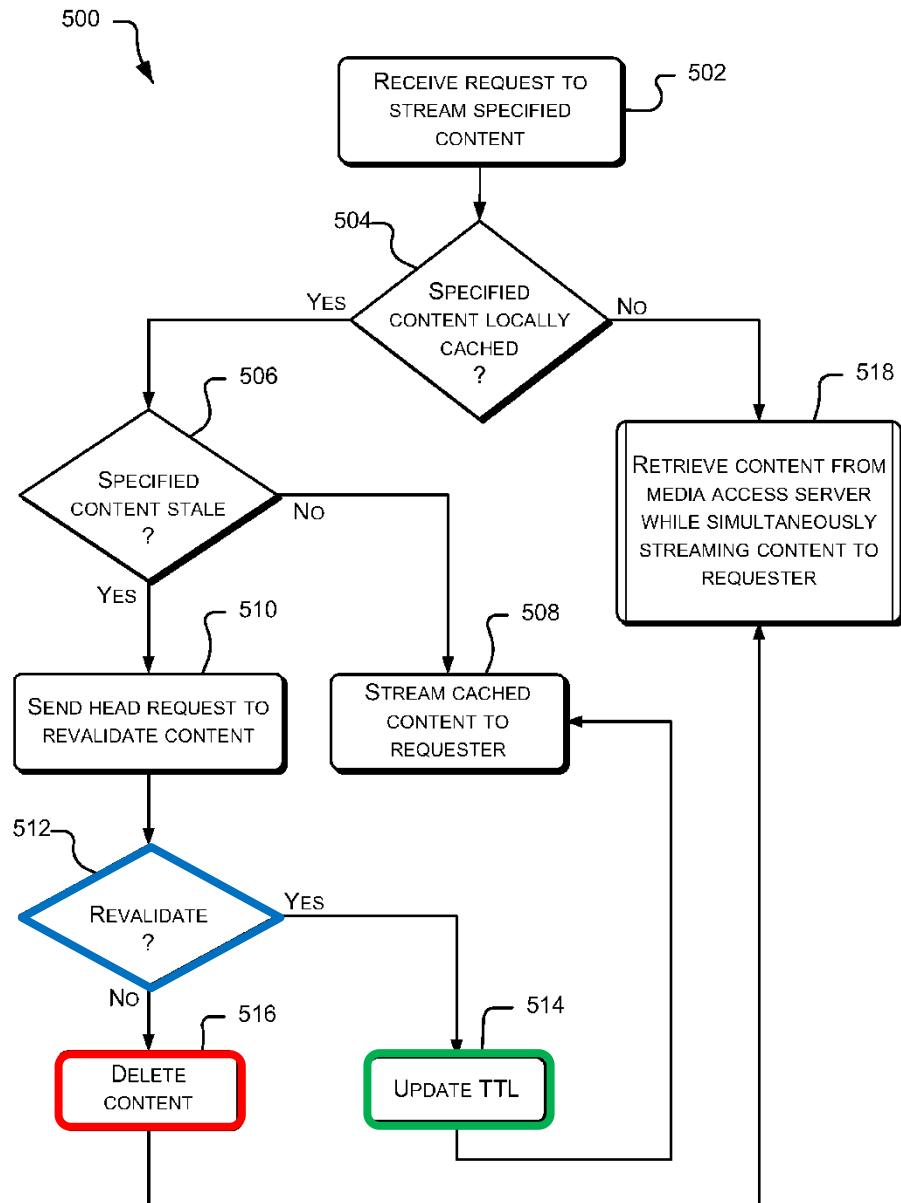


Fig. 5

Middleton, Fig. 5.

g. 1[b][iii]/8[b][iii]

Middleton discloses/suggests 1[b][iii]/8[b][iii]. Lanning, ¶¶202-03.

(B)(2), as recited in 1[b][iii]/8[b][iii], refers to 1[b][ii]/8[b][ii]. Lanning, ¶204.

Edge server's determination of whether a revalidation attempt was successful discloses 1[b][ii]/8[b][ii]. §VII.D.2.f; Lanning, ¶204.

Figure 5 illustrates that edge server determines whether the revalidation attempt was successful at operation 512 (blue), and content request handling operation 500 proceeds to operation 514 (green) if revalidation was successful, and operation 516 (red) otherwise. Middleton, [0082]-[0083]; Lanning, ¶205. Thus, operations of content request handling operation 500 performed after operation 512 are *based on said determining in (B)(2)*. Lanning, ¶206.

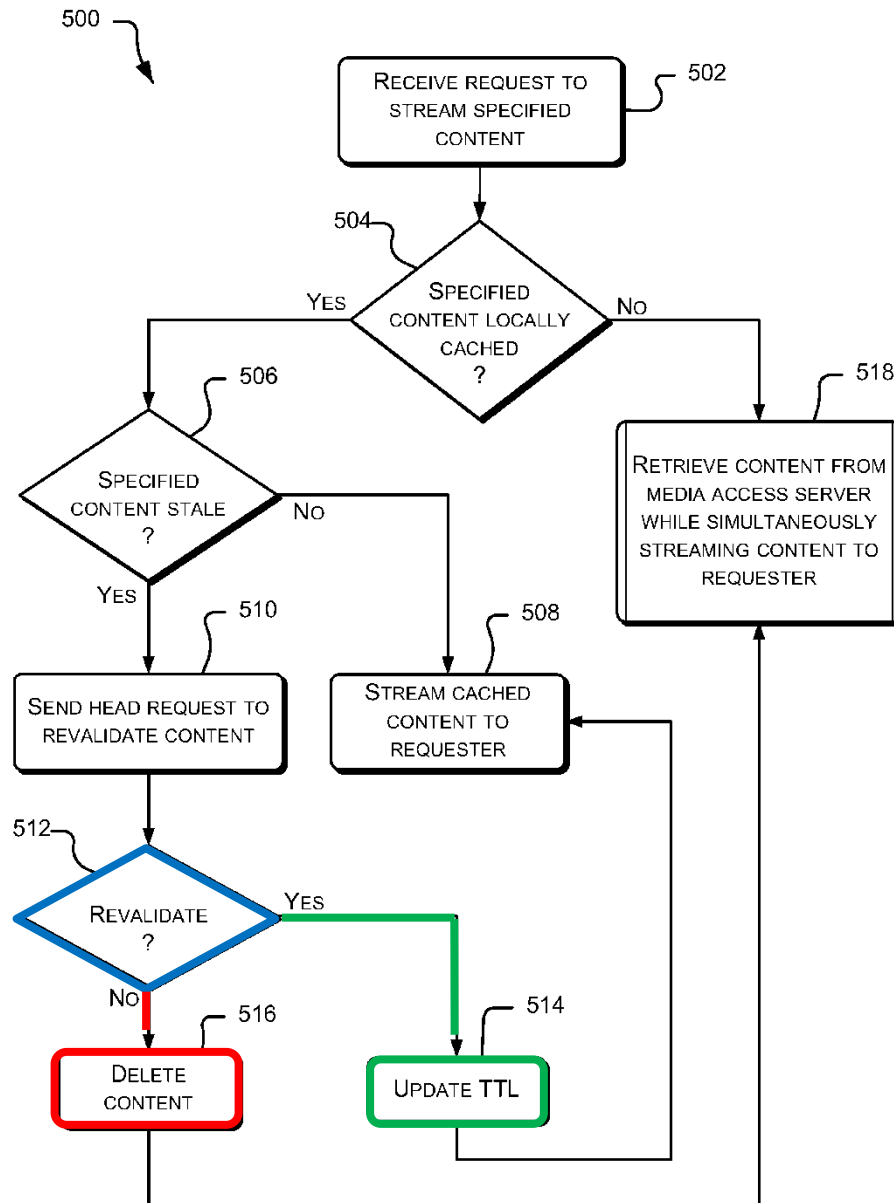


Fig. 5

Middleton, Fig. 5.

h. 1[b][iii][A]/8[b][iii][A]

Middleton discloses/suggests 1[b][iii][A]/8[b][iii][A]. Lanning, ¶¶207-08.

Whether the revalidation attempt was successful or not determines whether it is *acceptable to serve the content that is cached locally*. Lanning, ¶209; §VII.D.2.f. Middleton describes that “if the response from the MAS indicates that the locally cached content is not revalidated” (*when it is not acceptable to serve the content that is cached locally*), content request handling operation 500 proceeds to operation 516 to delete the locally-cached content. Middleton, [0083]; Lanning, ¶209. Then, “[i]n retrieving operation 518, the requested content is retrieved from the MAS while the content is simultaneously streamed to the requester.” Middleton, [0083]. For example, portions of the requested content may be received and cached by edge server (*obtaining, by the node, a new version of the content*), then sent to the requester (*and then serving, by the node, the new version of the content*), while the edge server retrieves other portions of the content. *Id.*, [0063]-[0064]; Lanning, ¶209.

Figure 5 illustrates retrieving content from MAS and streaming the content to the requester (operation 518, red) when the revalidation attempt was unsuccessful (operation 512, blue). Lanning, ¶210.

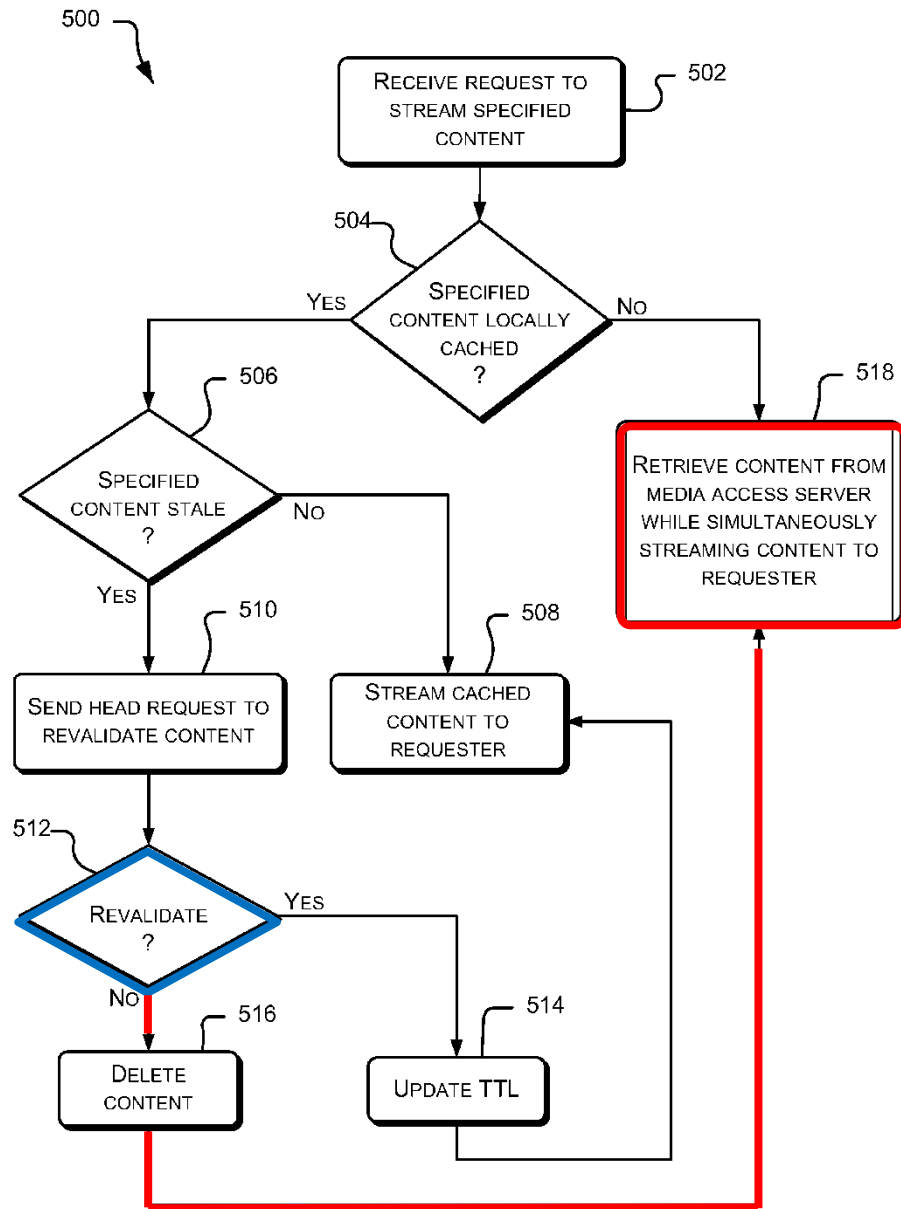


Fig. 5

Middleton, Fig. 5.

Middleton describes “a structured content delivery event model...[that] identifies discrete events within one or more content delivery process at which

various actions can be taken...within an extensible content delivery platform.” Middleton, [0112]. The “content delivery process is initiated when a connection with a user,” and upon a cache miss “the event model proceeds to operation 1112 at which a Cache Fill request is prepared to request the content resource from another source.” *Id.*, [0113]-[0115]. Middleton further describes that “[w]hen the Cache Fill is complete in operation 1122, the event model 1100 proceeds to operation 1110 indicating that a response to the user’s request for content is ready to be sent to the user.” *Id.*, [0115]-[0116].

Figure 11 illustrates (1) a user establishing a connection at operation 1102 and receiving a request for content at operation 1104 (red), (2) determining the requested content is not found in the cache at operation 1108 and making a cache fill request at operation 1112 (blue), and (3) determining the cache fill is complete at operation 1122, and responding to the user with the content at operations 1110 and 1124 (green). Lanning, ¶211.

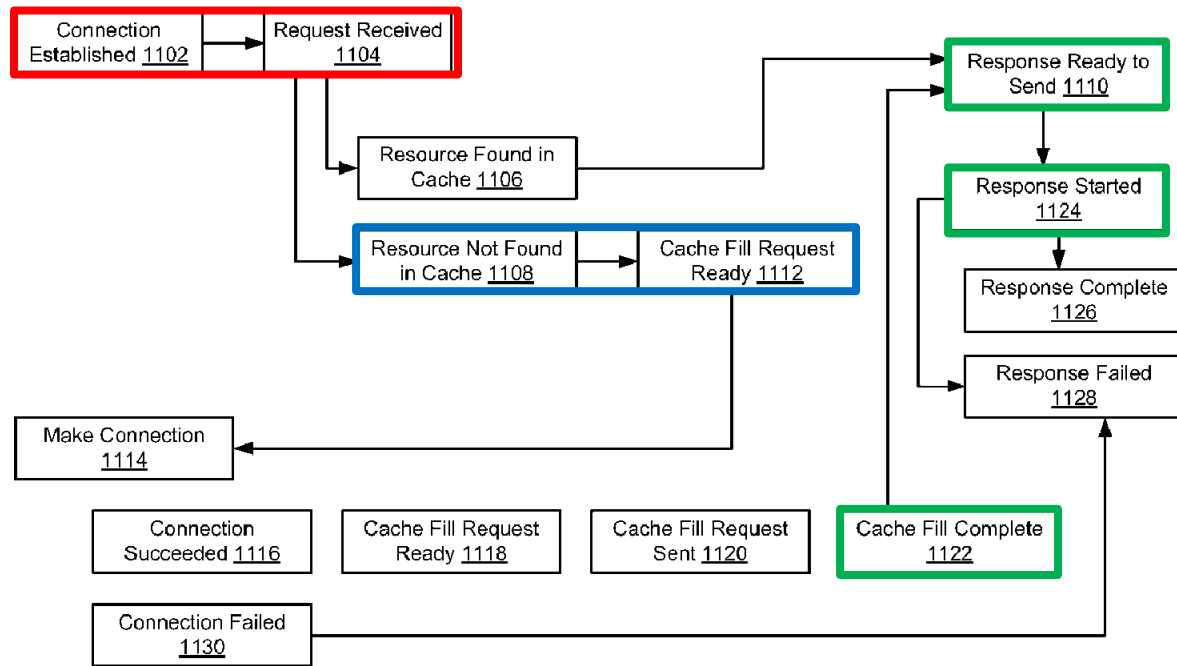


Fig. 11

Middleton, Fig. 11.

In other words, Middleton describes an event model in which requested content is sent to the user after completing the cache fill of the missing content. Lanning, ¶212. Thus, the event model performs “store-and-forward” where the cache stores requested content before it is served to the requesting user. *Id.* For this additional reason, Middleton discloses/suggests *obtaining, by the node, a new version of the content and then serving, by the node, the new version of the content.* *Id.*

i. 1[b][iii][B]/8[b][iii][B]

Middleton discloses/suggests 1[b][iii][B]/8[b][iii][B]. Lanning, ¶¶213-14.

Whether the revalidation attempt was successful or not determines whether it is *acceptable to serve the content that is cached locally*. Lanning, ¶215; §VII.D.2.f. Middleton describes if the content is revalidated (*when it is acceptable to serve the content that is cached locally*), content request handling operation 500 proceeds to operation 514 to update the TTL value associated with the locally-cached content so that it is no longer stale. Middleton, [0082]; Lanning, ¶¶215-17. “The locally cached content is then streamed in streaming operation 508” (*serving, by the node, the content that is cached locally*). *Id.*

Figure 5 illustrates streaming cached content to the requester (operation 508, green) when the revalidation attempt was successful (operation 512, blue) was successful. Lanning, ¶216.

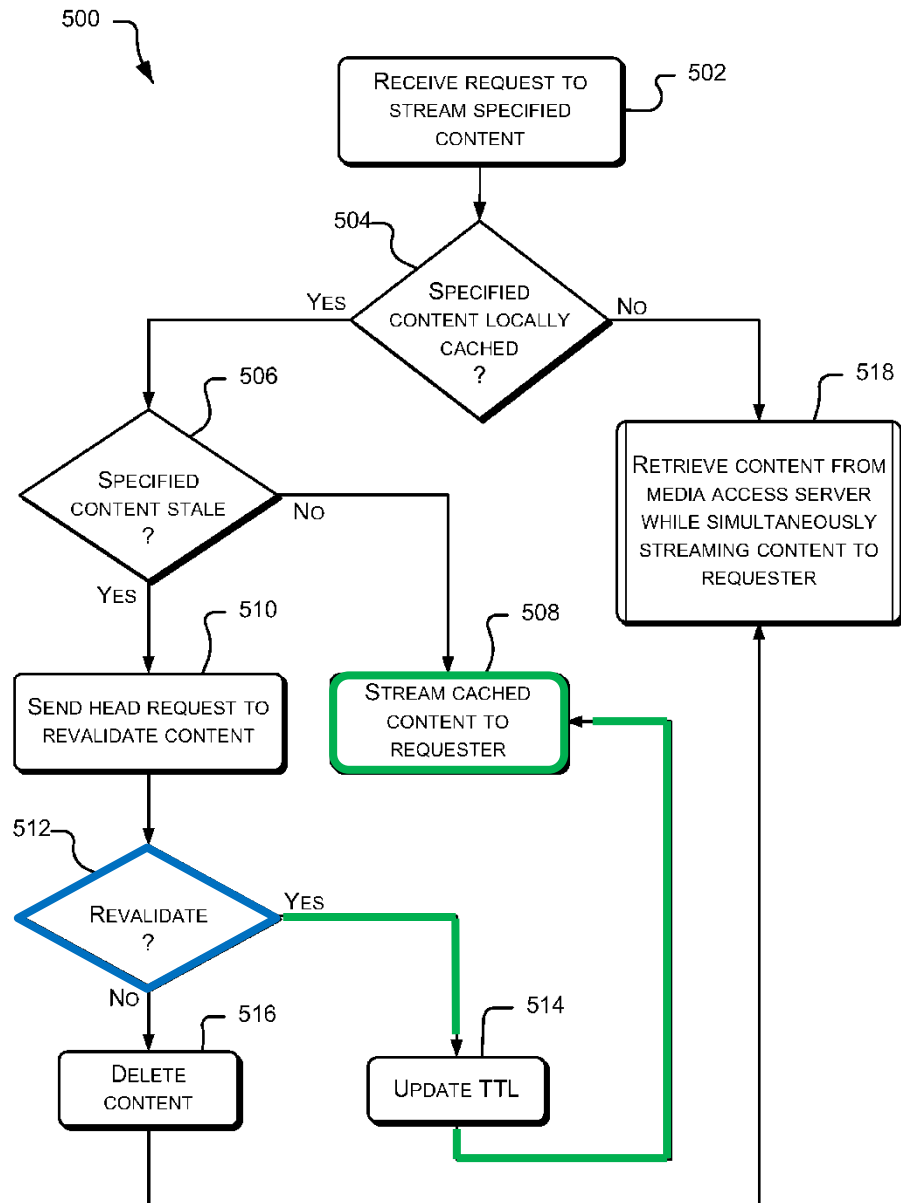


Fig. 5

Middleton, Fig. 5.

3. Claim 2, 9

a. 2[pre]

If limiting, Middleton discloses/suggests 2[pre]. Lanning, ¶218.

Middleton's content request handling operation 500 discloses/suggests *the method as recited in claim 1*. Lanning, ¶219; *see generally* §VII.D.2.

b. 9[pre]

If limiting, Middleton discloses/suggests 9[pre]. Lanning, ¶220.

Middleton's computer system is the *computer program product as recited in claim 8*, and Middleton's content request handling operation 500 discloses or suggests what *the method further comprises*. Lanning, ¶221; *see generally* §VII.D.2.

c. 2[a]/9[a]

Middleton discloses/suggests 2[a]/9[a]. Lanning, ¶222-23.

“[If], in query operation 504 it is determined that the requested content is not in the local cache” ((C) *when it is determined that said content is not cached locally*), “the operation 504 branches to retrieving operation 518.” Middleton, [0083]; Lanning, ¶224. Then, “[i]n retrieving operation 518, the requested content is retrieved from the MAS while the content is simultaneously streamed to the requester” ((C)(1) *obtaining the content and then (C)(2) serving the content obtained in (C)(1)*). Middleton, [0083]; Lanning, ¶224.

Figure 5 illustrates retrieving content from MAS and streaming the content to the requester (operation 518, red) when it is determined the requested content is not in the local cache (operation 504, blue). Lanning, ¶225.

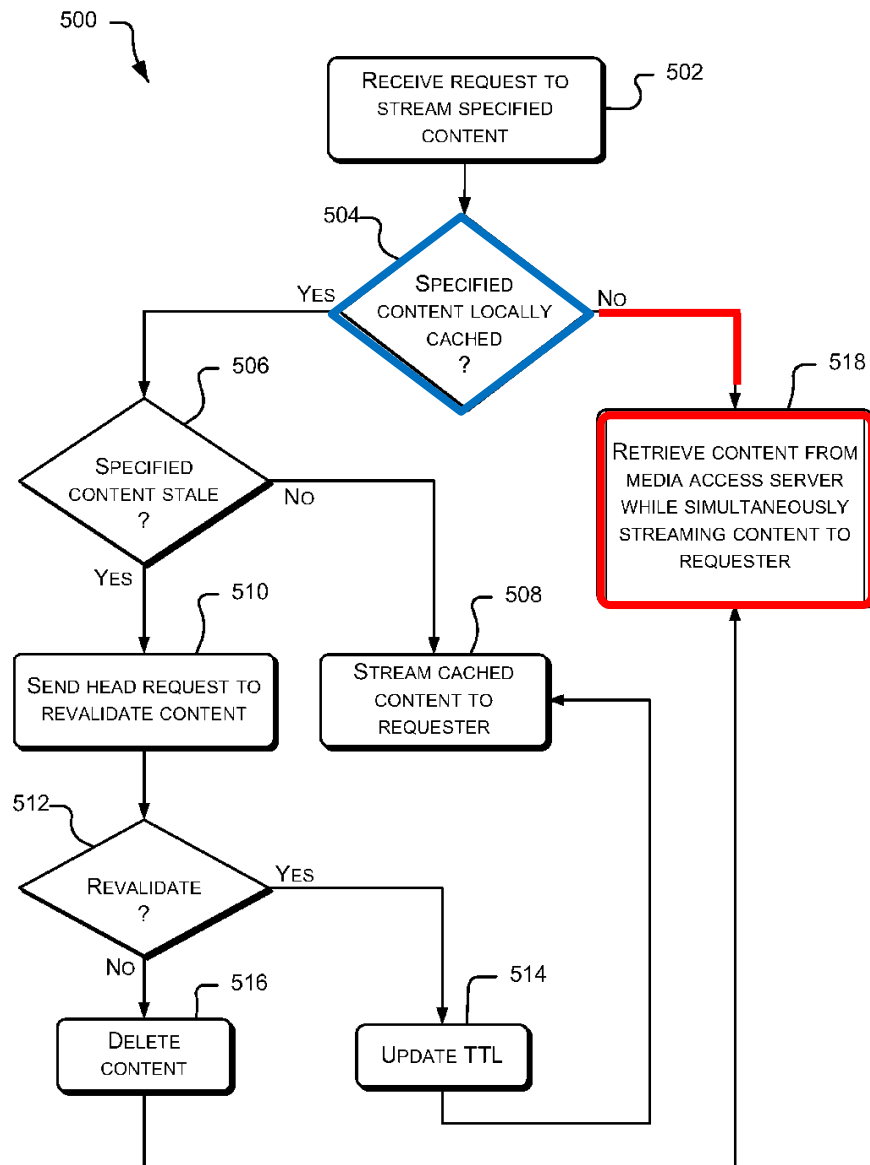


Fig. 5

Middleton, Fig. 5.

Middleton describes “a structured content delivery event model...[that] identifies discrete events within one or more content delivery process[es].” Middleton, [0112]. In the event model, requested content is sent to the user after completing the cache fill of the missing content. Lanning, ¶¶226-27; §VII.D.2.h. For this additional reason, Middleton discloses/suggests *(C)(1) obtaining the content and then (C)(2) serving the content obtained in (C)(1)*. Lanning, ¶227.

4. Claims 4, 11

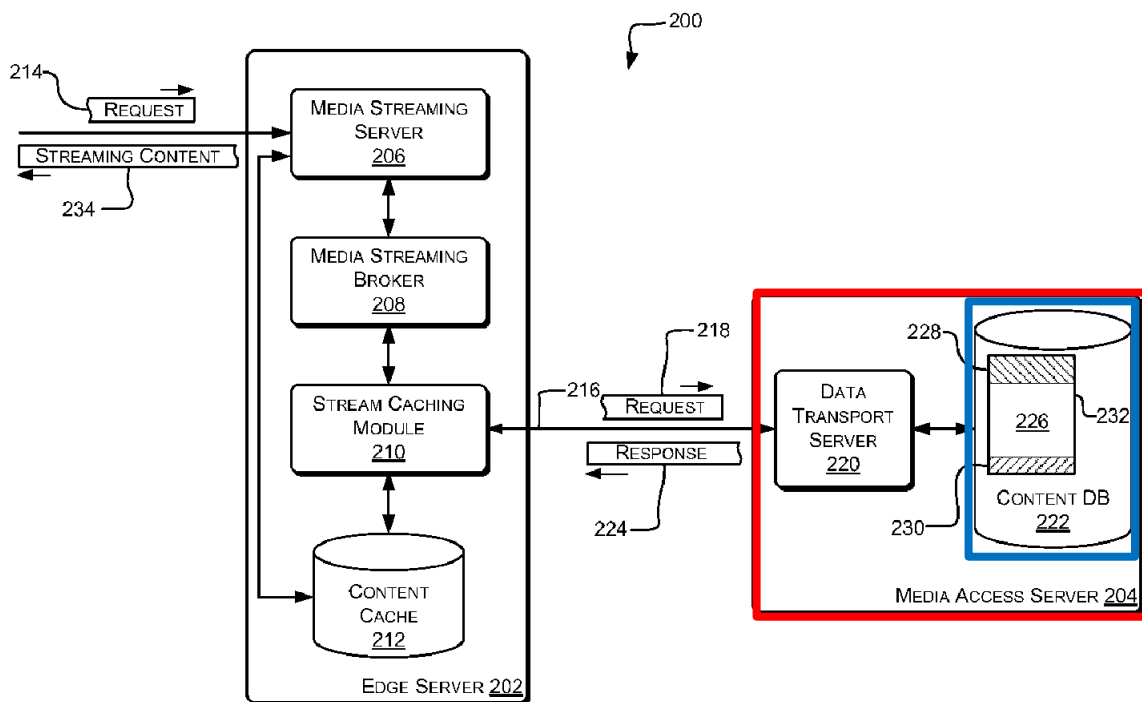
Middleton discloses/suggests claims 4 and 11’s additional limitations. Lanning, ¶228.

Middleton describes edge server (*node*) sending a request “to a media access server (MAS) to revalidate the locally cached content,” and determine a *current cache policy*. Middleton, [0054], [0082]; Lanning, ¶229; §VII.D.2.e. Media access server may be, e.g., a content distribution server computer or a content origin server, both of which are *distinct from* edge server (*node*). Middleton, [0040], Fig. 1; Lanning, ¶230. Thus, Middleton discloses/suggests that the *current cache policy is determined from a location distinct from the node*. Lanning, ¶230.

5. Claim 5, 12

Middleton discloses/suggests claims 5 and 12’s additional limitations. Lanning, ¶231.

Middleton describes “a content database 222 accessible to the MAS 204.” Middleton, [0055]. Figure 2 illustrates content database 222 (blue) within MAS 204 (red). Lanning, ¶232. MAS responses to edge server requests are based on the contents of content database. *See, e.g.*, Middleton, [0055]-[0057]. Content database is used when responding to a revalidation request from edge server. *Id.*, [0057]; Lanning, ¶233. Thus, the *location* the MAS uses when responding to a revalidation request *is a database*. Lanning, ¶233.



779. 2

Middleton, Fig. 2.

6. Claims 6, 13

Middleton discloses/suggests claims 6 and 13's additional limitations. Lanning, ¶234.

Middleton describes edge server “updat[ing] the TTL value associated with the locally cached content, so that the locally cached content is no longer stale[,]” based on the MAS response to the revalidation attempt (and new TTL value therein) (*cache policy*). Middleton, [0077], [0082]; Lanning, ¶235; §§VII.D.2.e-§VII.D.2.f. Edge server, which is *within the CDN*, uses the new TTL value to determine whether the locally-cached content is stale (e.g., in response to subsequent requests for the content). Middleton, [0080]; Lanning, ¶235. Thus, the TTL value *relates to caching content within the CDN*. Lanning, ¶235.

E. Ground 2B: Obviousness over Middleton and MS-MSSO

1. Motivation to Combine Middleton and MS-MSSO

A POSITA would have been motivated to modify Middleton's content delivery system to utilize the HTTP Streaming Protocol taught by MS-MSSO. Lanning, ¶¶236-41.

Middleton explains that while “[c]ache servers can be feature rich,” features were often “complex, sometimes interdependent, and often inconsistent in style and configuration.” Middleton, [0006]; §VII(D)(1). To solve these problems, Middleton discloses an “extensible content delivery platform” in which “[r]ich and well-formed

metadata and ‘metacode’ can be associated with content and/or requests in a consistent and programmatic manner.” Middleton, [0007].

Middleton’s cache server may be a Windows Media Server. Specifically, “[t]he edge computer server 108 is configured with media streaming server software, such as...Windows Media Server™ (WMS). As such, if the requested content is found to be locally stored on the edge computer server 108 and the cached content is not stale, the media streaming software can stream the requested content to the requester.” *Id.*, [0033]. Middleton further describes that during operation, a request “is initially received by the media streaming server. The media streaming server 206 could be a...Windows Media Server (WMS)...The media streaming server 206 is configured to communicate data with a content requester using a data streaming protocol.” *Id.*, [0048].

A POSITA, aware of the benefits of Middleton’s extensible content delivery platform, which could be used with existing cache servers including the Windows Media Server, would have looked to MS-MSSO for further details about how the Windows Media Server operated and the supported functions/protocols. Lanning, ¶240. A POSITA would have further recognized that the Windows Media HTTP Streaming Protocol, used by Windows Media Servers, provides additional advantages when, for example, “the multimedia data needs to pass through a proxy or a firewall.” MS-MSSO, 47; Lanning, ¶240. Accordingly, a POSITA, aware of the

benefits both of an extensible content delivery platform and of optimizing data transmission in various network environments, including those involving proxies and firewalls, would have been motivated to further improve upon Middleton (extensible content delivery platform including a Windows Media Server) to utilize the Windows Media HTTP Streaming Protocol, as taught by MS-MSSO. Lanning, ¶240.

This would have involved combining prior art elements (MS-MSSO's Windows Media HTTP Streaming Protocol, and Middleton's platform incorporating a Windows Media Server) according to known methods (deploying a Windows Media Server as an edge server as taught by Middleton, and utilizing the Windows Media HTTP Protocol as taught by MS-MSSO) to yield predictable results (improved caching and network performance). *Id.*

Furthermore, a POSITA would have had a reasonable expectation of success combining Middleton with MS-MSSO in this manner. *Id.*, ¶241. Middleton already taught an edge server that could be implemented as a Windows Media Server. *Id.*; Middleton, [0033], [0048]. A POSITA would have recognized that the Windows Media Server taught in Middleton could be readily modified to utilize the Windows Media HTTP Streaming Protocol. Lanning, ¶241. This combination would be well within the capabilities of a POSITA because Middleton describes implementing the invention using a Windows Media Server as its cache server, and MS-MSSO

describes using various protocols within such a Windows Media Server, including the Windows Media HTTP Streaming Protocol. *Id.* Additionally, the resulting system's elements would perform functions performed prior to combination—Middleton's platform, utilizing a Windows Media Server, would provide improved access existing to cache server features, and utilizing the Windows Media HTTP Streaming Protocol as described in MS-MSSO would improve network performance. *Id.*

2. Claims 3, 10

Middleton (with/without MS-MSSO) teaches claims 3 and 10's additional limitations. Lanning, ¶242.

Middleton describes that edge server uses a data transport protocol, such as HTTP, to communicate with other devices of the CDN. *See, e.g.,* Middleton, [0044], [0045], [0078]; Lanning, ¶243. And Middleton further describes that “[i]n one embodiment of a [CDN] servicing HTTP/S requests, for example, events in the process comprise a Receive Request (e.g., an HTTP request received by the content delivery network).” Middleton, [0118]. Thus, the *request* received by the CDN *comprises an HTTP request*. Lanning, ¶243.

Additionally, Middleton describes that the content provided by the CDN “may be any of numerous types of content, including video, audio, images, text, multimedia, or any other type of media.” Middleton, [0029]. In other words, the

requested content *comprises one or more resources* (e.g., audio and/or video data). Lanning, ¶244; *see also* MS-MSSO, 46 (“The Windows Media HTTP Streaming Protocol is used to transfer real-time multimedia data (that is, audio and video)”).

If PO argues Middleton alone does not teach *an HTTP request*, Middleton-MS-MSSO does. Lanning, ¶245.

Middleton describes edge server may be “configured with media streaming server software, such as...Windows Media Server.” Middleton, [0033]. Window Media Servers utilized the Windows Media HTTP Streaming Protocol, which “depends on HTTP for the transfer of all protocol messages, including the transfer of the multimedia data.” MS-MSSO, 47; Lanning, ¶246; §VII.B.1. Windows Media HTTP Streaming Protocol was advantageous when, for example, “the multimedia data needs to pass through a proxy or a firewall.” MS-MSSO, 47. It therefore would have been obvious in the combined Middleton-MS-MSSO system for the cache server configured as a Window Media Server (as taught by Middleton) to utilize the Windows Media HTTP Streaming Protocol (as taught by MS-MSSO), such that a request to the cache server *comprises an HTTP request* for this additional reason. Lanning, ¶246.

F. Ground 2C: Obviousness over Middleton and Blumofe

1. Motivation to Combine Middleton and Blumofe

A POSITA would have been motivated to modify Middleton's content delivery system to add an extender as Blumofe teaches. Lanning, ¶¶247-59.

Middleton explains that within a network, cache servers could be deployed between origin servers and end users to improve network performance. Middleton, [0031]. While “[c]ache servers can be feature rich,” the features were often “complex, sometimes interdependent, and often inconsistent in style and configuration.” *Id.*, [0006]; §VII(D)(1). To solve these problems, Middleton discloses an “extensible content delivery platform” in which “[r]ich and well-formed metadata and ‘metacode’ can be associated with content and/or requests in a consistent and programmatic manner.” Middleton, [0007].

Middleton explains that it was desirable to position cache servers so that they are close to users. Lanning, ¶250. Specifically, Middleton explains that it is advantageous for cached content to be “available in a more geographically or logically proximate location to the end user” because this “could reduce content download time to the user computer 104.” Middleton, [0031]. Further, the CDN could “include one or more networks” and “the user may use any of numerous different types of computing devices to access the network 102, including, but not limited to, a laptop computer, a handheld computer, a personal digital assistant

(PDA), or a cell phone.” *Id.*, [0027]-[0028]. However, Middleton is silent regarding how the invention could be used with wireless devices such as PDAs or cell phones, or how information could be cached in a location close to a user using a mobile network such as a 3G network. Lanning, ¶250.

Blumofe teaches “extend[ing] [a CDN] into one or more mobile or wireline networks that cannot or do not otherwise support fully-managed CDN edge servers.” Blumofe, Abstract; §VII(C)(1). Blumofe observes that in 3G wireless mobile networks, gateways at which CDN servers could be deployed, may be thousands of miles and hundreds of milliseconds away from end-users. Blumofe, [0007].

To address this, Blumofe describes a CDN Extender. *Id.*, [0025]. The Extender was deployed on the network, e.g., “as a passive web caching proxy that is beyond the edge of the CDN but that serves CDN-provisioned content under the control of the CDN.” *Id.*, [0009]. The CDN controls the Extender, and the CDN supplies caching policies to the Extender, i.e., caching policies within the CDN related to caching content outside the CDN. *Id.*, [0036]. This enables serving “CDN-provisioned content under the control of the CDN” in situations where it was not “possible or practical to deploy fully-managed CDN servers.” *Id.*, [0007]-[0009].

A POSITA, aware of the benefits both of Middleton’s extensible content delivery platform and of providing caching functionality close to wireless end users, would have been motivated to improve upon Middleton (an extensible content

delivery platform) to allow for further caching resources within a wireless network. Lanning, ¶253. And a POSITA would have recognized Blumofe would provide this benefit to Middleton. *Id.*

Middleton and Blumofe are directed to content delivery networks. Middleton, [0002]; Blumofe, [0005]. And both describe shortcomings with conventional CDN architectures and are directed to solving these problems by managing caches deployed between origin servers and end users. Middleton, [0031]; Blumofe, [0009]. Both references describe methods for the content owner to specify the methods used for caching content. Middleton, [0010], [0035]; Blumofe, [0030].

Moreover, Blumofe teaches an extender that is designed to work in combination with an existing CDN, such as Middleton's CDN. Lanning, ¶255. Blumofe explains that the Extender "is deployed in the mobile or wireline network, preferably as a passive web caching proxy that is beyond the edge of the CDN but that serves CDN-provisioned content" and the Extender "preferably delivers all forms of CDN content (i.e., content that has been off-loaded by content providers for delivery by the CDNSP [CDN service provider, i.e., the entity managing the CDN] over the CDN)." Blumofe, [0009], [0029]; Lanning, ¶255.

A POSITA would have found it obvious to combine Middleton, which describes a method of managing cache servers within a CDN (*see generally supra*), with Blumofe, which describes managing caches both within and outside the CDN

(e.g., within the Extender). Lanning, ¶256. That is, a POSITA would have found it obvious to utilize an extender as taught in Blumofe with the CDN taught by Middleton. *Id.* A POSITA would have recognized the benefits this combination provides. *Id.* For example, using an extender as taught in Blumofe would have allowed using Middleton's extensible content delivery platform in conjunction with other network environments (e.g., wireless networks) where it otherwise could not have been deployed. *Id.* And this combined system would likewise have allowed content to be cached closer to wireless users, providing improved network performance. *Id.* Moreover, using an extender (as taught in Blumofe) in combination with Middleton's extensible content delivery platform would enable consistently enforcing cache policies both inside and outside the CDN. *Id.*

This would have involved applying a known technique (caching content in wireless networks) to improve similar devices (CDNs) in the same way (deploying caches in other network environments and/or closer to the user). *Id.*, ¶257. Further, this would have involved combining prior art elements (an extender as taught in Blumofe and Middleton's CDN) according to known methods (providing cache servers between origin servers and end users) to yield predictable results (improved network performance). *Id.*

A POSITA would have had a reasonable expectation of success combining Middleton with Blumofe in this manner. *Id.*, ¶258. Middleton already taught a cache

server that could be used in conjunction with various networks, and that could be used to deliver content to wireless devices such as PDAs and cell phones. *Id.*; Middleton, [0027]-[0028]. A POSITA would have recognized that an extender as taught in Blumofe could be readily incorporated into Middleton's system. Lanning, ¶258. This combination would be well within the capabilities of a POSITA because an extender as taught in Blumofe was already designed to work with an existing CDN. *Id.*; *see also* Blumofe, [0009], [0029]. Moreover, Middleton and Blumofe describe similar CDN environments and communications between origin servers, caches, and clients, such that a POSITA would have found it straightforward to modify Middleton's CDN to implement an extender as taught in Blumofe. Lanning, ¶258. Additionally, the resulting system's elements would perform functions performed prior to combination—Middleton's CDN would allow users to customize caching policies, and the extender would enable caching content in wireless networks and/or closer to the user. *Id.*

A POSITA would have understood that in such a combination, the CDN would send the cache control information to the extender so that the extender would perform according to the CDN customer's rules, and the cache policy that configures the cache server within the CDN (e.g., with respect to whether cached content has expired or is an acceptable version) would also be used to configure the extender outside the CDN. *Id.*, ¶259; *see also* Blumofe, [0030], [0036], [0037]. The

combination would have been well within the capabilities of a POSITA because it would have merely required modifying Middleton's CDN to include an extender as taught by Blumofe. Lanning, ¶259.

2. Claims 7, 14

Middleton-Blumofe teaches claims 7 and 14's additional limitations. Lanning, ¶260.

Blumofe describes a CDN Extender operating as a cache beyond the CDN's edge. *See, e.g.*, Blumofe, [0027], [0050]; §VII.C.1. The combined Middleton-Blumofe system's extender is thus *outside the* CDN, and content cached by the extender in the combined system is *cached content outside the CDN*. Lanning, ¶261.

When the extender receives an end-user request for content that is not in its cache or is present in the cache but stale, the extender sends a GET request to the CDN to fetch content. Lanning, ¶262; Blumofe, [0033]; §VII.C.1. Similarly, if the requested content is already in the extender's cache, extender sends an if-modified-since (IMS) request to a CDN edge server. Blumofe, [0033]. In response, the CDN edge server provides cache control information, such as TTL expiration times. Blumofe, [0036], [0037]; Lanning, ¶262.

The cache control information returned by the CDN controls the expiration time of the cached content on the extender, and thus *relates to caching content outside the CDN*. Lanning, ¶263. Moreover, Blumofe describes that the CDN sends

the cache control information to the extender so that the extender would perform according to the CDN customer's rules. Blumofe, [0030], [0036]; Lanning, ¶264. And it would have been obvious, in the combined Middleton-Blumofe system, that the *cache policy* that configures edge server within the CDN (e.g., with respect to whether a revalidation attempt was successful, and if so, the new TTL value for the content) would have also been used to configure the extender outside the CDN. Lanning, ¶264. In other words, upon a GET or IMS request by the extender to the combined Middleton-Blumofe system's edge server, it would have been obvious when edge server performs a revalidation attempt (as taught by Middleton) to utilize the TTL value obtained via the revalidation attempt when responding to the extender with cache control information (as taught by Blumofe). *Id.* Otherwise, the extender would operate differently than the CDN's cache server (e.g., the edge server would have one TTL value for a piece of content, and the extender another), which a POSITA would have understood would be an undesirable result because it could result in serving stale content and would contradict Blumofe's teaching that "the Extender acts under the control of the CDN" so that "the correct content [is] delivered according to customer metadata rules." Blumofe, [0030], Lanning, ¶264; *see also* Blumofe, [0054]. Thus, the *cache policy* (*see* §VII.D.2.e) *also relates to caching content outside the CDN*. Lanning, ¶264.

VIII. CONCLUSION

IPR of the Challenged Claims is respectfully requested.

IX. MANDATORY NOTICES

A. Real Party in Interest

Petitioner identifies itself as the real party in interest.

B. Related Matters

To the best of Petitioner's knowledge, the '173 Patent has been involved in the following matters:

- *Sandpiper CDN, LLC v. Microsoft Corporation*, No. 2-25-cv-00664 (E.D. Tex. June 26, 2025).

C. Notice of Counsel and Service Information

LEAD COUNSEL
Jessica Kaiser (Reg. No. 58,937) kaiser-ptab@perkinscoie.com PERKINS COIE LLP 1900 Sixteenth Street, Suite 1400 Denver, CO 80202-5255 Telephone: (303) 291-2300
BACK-UP COUNSEL
Christopher Marando (Reg. No. 67,898) Marando-ptab@perkinscoie.com PERKINS COIE LLP 700 Thirteenth Street N.W., Suite 800 Washington, DC 20005-3960 Telephone: (202) 654-6200

Matthew A. Lembo (Reg. No. 75,633)
Lembo-ptab@perkinscoie.com
PERKINS COIE LLP
1155 Avenue of the Americas, 22nd Floor
New York, NY 10036-2711
Telephone: (212) 262-6900

Steven Beigelmacher (Reg. No. 75,050)
beigelmacher-ptab@perkinscoie.com
PERKINS COIE LLP
1301 Second Avenue, Suite 4200
Seattle, WA 98101-3804
Telephone: (206) 359-8000

Sarah Fowler (Reg. No. 54,854)
stahnke-ptab@perkinscoie.com
PERKINS COIE LLP
3150 Porter Drive
Palo Alto, CA 94304
Telephone: (650) 838-4300

Petitioner consents to electronic service. All services and communications to the attorneys listed above may be sent to:

Microsoft-Sandpiper-IPR@perkinscoie.com

D. Power of Attorney

A power of attorney is filed herewith according to 37 C.F.R. §42.10(b).

Respectfully submitted,

/ Jessica Kaiser /

Jessica Kaiser
Reg. No. 58,937
Attorney for Petitioner

PERKINS COIE LLP
1900 Sixteenth Street, Suite 1400
Denver, CO 80202-5255

Date: December 24, 2025

CERTIFICATE OF WORD COUNT UNDER 37 CFR §42.24(d)

Pursuant to 37 C.F.R. §42.24(a), Petitioner hereby certifies that portions of the above-captioned Petition for *inter partes* review of U.S. Patent No. 10,701,173, in accordance with and reliance on the word count provided by the word-processing system used to prepare this Petition, that the number of words in this paper is 13,981. Pursuant to 37 C.F.R. §42.24(a), this word count is in compliance and excludes the table of contents, table of authorities, mandatory notices under §42.8, certificate of service, certificate of word count, appendix of exhibits, and any claim listing. This word count was prepared using Microsoft Word.

Respectfully submitted,

/ Jessica Kaiser /

Jessica Kaiser
Reg. No. 58,937
Attorney for Petitioner

Date: December 24, 2025
PERKINS COIE LLP
1900 Sixteenth Street, Suite 1400
Denver, CO 80202-5255

CERTIFICATE OF SERVICE

The undersigned hereby certifies that true copies of the Petition for *inter partes* review of U.S. Patent No. 10,701,173 and supporting materials (Exhibits and Power of Attorney) were served via overnight delivery on the Patent Owner at the correspondence address of record as listed on PAIR:

STERNE, KESSLER, GOLDSTEIN & FOX P.L.L.C.
1101 K Street, NW
10th Floor
WASHINGTON, DC 20005

A courtesy copy was also sent via electronic mail to Patent Owner's litigation counsel listed below:

Ryan D. Dykal - rdykal@bsfllp.com
Mark D. Schafer - mschafer@bsfllp.com
Phillip Eckert - peckert@bsfllp.com

Respectfully submitted,

/ *Jessica Kaiser* /

Jessica Kaiser
Reg. No. 58,937
Attorney for Petitioner

Date: December 24, 2025